

MS11-L\*  
MS11-M\*

MS11-L/M MEMORY  
CZMSDBO

AH-F295B-MC  
FICHE 1 OF 2

FEB 1981  
COPYRIGHT © 79-80  
MADE IN USA





MS11-L\*  
MS11-M\*

MS11-L/M MEMORY  
CZMSDBO

AH-F295B-MC  
FICHE 2 OF 2

FEB 1981  
COPYRIGHT © 79-80  
MADE IN USA



Table with multiple columns and rows of data, including headers and numerical values. The text is faint and difficult to read.



IDENTIFICATION

PRODUCT CODE: AC-F294B-MC

PRODUCT NAME: CZMSDBO MS11-L/M MEMORY DIAGNOSTIC

DATE CREATED: OCTOBER 1980

MAINTAINER: BASE SYSTEM DIAGNOSTIC ENGINEERING

AUTHOR: MICHAEL D BIBEALT

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 DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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, 1980 BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

- 1.0 GENERAL PROGRAM INFORMATION
  - 1.1 Program Purpose (Abstract)
  - 1.2 System Requirements
  - 1.3 Related Documents And Standards
  - 1.4 Diagnostic Hierarchy Prerequisites
  - 1.5 Assumptions
- 2.0 OPERATING INSTRUCTIONS
  - 2.1 Loading and Starting Procedures
  - 2.2 Default Test Sequence
  - 2.3 Special Environments
  - 2.4 Program Options
  - 2.5 Execution Times
- 3.0 ERROR INFORMATION
  - 3.1 Error Reporting
  - 3.2 Error Abbreviations
  - 3.3 Error Halts
- 4.0 PROGRESS REPORTS
- 5.0 CSR INFORMATION TABLES
  - 5.1 Core/MOS Parity CSR
  - 5.2 MOS Bipolar CSR
  - 5.3 MF11S-K CSR
  - 5.4 MS11-L CSR
  - 5.5 MS11-M CSR
- 6.0 SUB-TEST SUMMARIES
  - 6.1 Tests
  - 6.2 Patterns
- 7.0 PROGRAM FEATURES
  - 7.1 Fast Data Access Rates
  - 7.2 Bank Zero Testing
  - 7.3 Memory Configuration Map
  - 7.4 Everything You've Always Wanted To Know About SUPERMAC ...
  - 7.5 Memory Management Mapping



## 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.

### 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 2.4.3.1)
2. Inhibit relocation (reference section 2.4.1)
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

#### NOTE

If on an 11/24 using MS11-L Memory BE SURE that the peripheral page jumper is in place; failure to do so sends the diagnostic to Never-Never Land.

#### 2.1.2 Stopping -

1. Set SW8, and/or
2. Type control "C" (Reference section 2.4.4.1).

#### 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 2.4.4.8).

### 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)
5	Rotating 1's Test	1
@ 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 extremely 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 conjunction 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 2.4.1.1, 2.4.1.2, & 2.4.1.3).

## 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 7.3

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.

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	PAFPAF	Patterns forward, banks forward
(12)	5	PAFPAW	Patterns forward, banks worst first
(14)	6	PARPAF	Patterns reverse, banks forward
(16)	7	PARPAW	Patterns reverse, banks worst first.

For more details reference section 2.4.1.1, 2.4.1.2 and 2.4.1.3.

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. 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 'PAFBAP'.

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 'MTO0NN' and 'MTP0NN' - where 'NN' is the Pattern number. MTO0NN refers to the routine that sets up for the test Pattern whereas MTP0NN 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 7.3).

#### NOTE

Another way to obtain this information is to type a CONTROL/T at the console (reference Section 2.4.4.5).

### 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 7.3).

## 2.4.4 Terminal Commands -

### 2.4.4.1 Control 'C'

This command will:

1. If Switch 8 (Halt Program) 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 to enter a modified version of ODT has been deleted.

### 2.4.4.3 Control 'E' (procEEd)

This command would allow you to exit ODT. It is has also been deleted.

### 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

By use of Field Service Command 17 "Trace" can be set so that it will automatically type out the bank and pattern numbers as each pattern is run. (Reference section 2.4.4.8.18).

#### 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" (Continue)

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

#### NOTE

CSR references are done in accordance with section 5.0.

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 'TIMEOUT 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 'TIMEOUT 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 physical 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.

#### NOTE

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 (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.

#### 2.4.4.8.9 Field Service Command 8 (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.

#### 2.4.4.8.10 Field Service Command 9 (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.

#### 2.4.4.8.11 Field Service Command 10 (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.

#### 2.4.4.8.12 Field Service Command 11 (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.

#### 2.4.4.8.13 Field Service Command 12 (Exit Kamikaze Mode)

Return to the default mode of testing (undo Command 12).



2.4.4.8.14 Field Service Command 13 (Turn Cache Off)

This changes the Cache constant to bypass cache (reference section 2.4.3.1).

2.4.4.8.15 Field Service Command 14 (Turn Cache On)

This changes the Cache constant to use cache (reference section 2.4.3.1).

2.4.4.8.16 Field Service Command 15 (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 inaccessible via command 5.

2.4.4.8.17 Field Service Command 16 (Resume Testing All Banks)

Return to the default mode of testing (undo Command 15).

2.4.4.8.18 Field Service Command 17 (Resume Testing All Banks)

Enable "Trace". After exiting field service mode, the program will type out the bank and pattern numbers as each pattern is run.

2.4.4.8.19 Field Service Command 18 (Resume Testing All Banks)

Disable "Trace". (undo Command 16).

## 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     18 Sec per BANK of Non-Intereaved MS11-M  
     Add     15 Sec per BANK of Interleaved MS11-M  
     Add     6 Sec per BANK of MS11-L

Quick Verify Pass  
     Add     8 Sec per BANK of Non-Interleaved MS11-M  
     Add     7 Sec per BANK of Interleaved MS11-M  
     Add     6 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 typenut 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 all 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

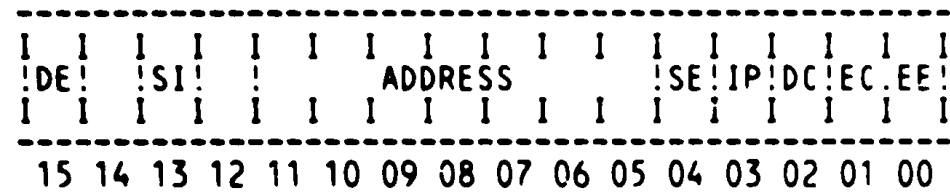
Reference Section 2.4.4.8.18 for more information on Tracing.







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

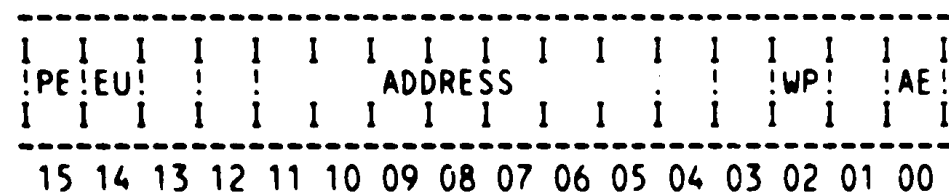
BIT03 INHIBIT MODE  
POINTER The Inhibit  
Mode Pointer works in  
conjunction with the  
Set Inhibit Mode bit.  
When BIT13 is set to a  
1, a 16K portion of  
memory is inhibited  
from operating in the  
ECC Disable mode or  
Diagnostic Check mode.  
the Inhibit Mode  
Pointer indicates  
which 16K is being in-  
hibited; e.g.-when  
BIT 3 =1, the second  
16K of memory is in-  
hibited. 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 correc-  
tion takes place and  
the error is not log-  
ged in the csr; cor-  
rect check bits are  
still written to the  
memory however.

BIT00 DOUBLE ERROR  
ENABLE When set  
enables trap to vector  
114 on double error.

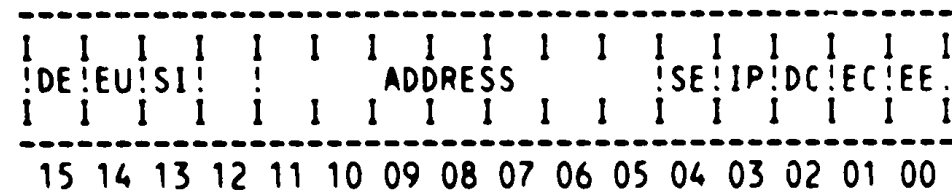
5.4 MS11-L CSR



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 inhibited  
from operating in the  
ECC Disable mode or  
Diagnostic check mode.  
the Inhibit Mode  
Pointer indicates  
which 16K is being in-  
hibited; e.g.-if BIT3  
=1, the second 16K of  
memory is inhibited.  
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 correcti-  
on 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 as-  
sert 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 'MTPXY' where X may be any sub program indicator (A,B,C,etc) or 0 and Y will be the number of the pattern.

Setup procedures for each pattern are identified by symbolic locations 'MTOOY' where Y 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, or (2) manually.

#### 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.  
:BITO SET WILL RUN THE FIRST ENTRY IN THE TABLE, BITO 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 checkbits) 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 MOV<sub>B</sub> 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 MOV<sub>B</sub> 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.

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 &amp; 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 API or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze 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 Traos.
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.

N 5

SEQ 0065

#### 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 ATT, and as part of a shutdown sequence directed by Switch 8 (Halt Program).

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...

```

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
1$:     BVS             2$           ;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 originally came from the PDP-11 Family Instruction Exerciser DZOKA-A. The first MOS memorys fell succceptable 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 API 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 the 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 occurs.
3. Make sure error address bits are set correctly.
4. Write good parity without trapping, and make sure no trap occurs 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 API 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 2.4.3.1).

## 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 overcome this because the UNIBUS Map can fool APT.

Because of the previous arguments this program does not relocate in the true sense 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 2.4.1). It can also be printed at any later time in Field Service Mode (reference section 2.4.4.8.7)

Example:

```

                MEMORY CONFIGURATION MAP
                16K BANKS
                1       2       3       4       5       6       7
0123456701234567012345670123456701234567012345670123
ERRORS              XX
CPU MAP 111111111111111111111111111111111111
INTRLV  -----3333333333333333
MEMTYPE LLLLLLLMMMMMMMMMMMMMMMMMMMMMMMM
CSR     00000001111111122222222222222222
PROTECT PP  I    I    P
          0      1      2      3      4      5      6
45670123456701234567012345670123456701234567
ERRORS
CPU MAP
INTRLV
MEMTYPE
CSR
PROTECT

```

Displayed are Banks 0-73 Octal (2 meg words). If the Fat Terminal Switch was set (reference section 2.4.1) then all Banks (0-167) would be shown. If this was an 18-Bit PDP-11 (eg - 11/34), only Banks 0-7 would be printed.

The fields:

**ERRORS:**

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

**CPU MAP:**

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

**INTRLV:**

There is interleaving on Banks 20-37, with CSR 2 (172104) controlling the Address Bit 1 Non-Asserted addresses, and CSR 3 (172106) controlling the Address Bit 1 Asserted addresses.

**ERRORS:**

**MEMTYPE:**

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

i1/45-type Bipolar memory.

SEQ 0076

## CSR:

Banks 0-7 are assigned to CSR 172100, 10-17 to CSR 172102, and 20-37 to interleaved CSR's 172104 and 172106.

## PROTECT:

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 occurs 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 statement 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 line 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 RO := 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,J
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 BITS SET.IN A
  MOV B,C
END ;OF IF BITS
IF BITS OFF.IN A
  MOV C,D
END ;OF IF BITS
;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 RO : = 0 TO 140 BY 4
    DEC    A(RO)
END ;OF FOR RO
FOR I :_ = 133 DOWNT0 3 BY 2
    ADD    A,B
END ;OF FOR I
;BEGIN EXAMPLES
BEGIN ALPHA
FOR RO :_ = 0 TO 167
    MOVB   A(RO),B
    IF B LT 0 THEN LEAVE ALPHA
END ;OF FOR RO
FOR RO : = 400 TO 567
    IF B GE 0 THEN LEAVE ALPHA
END ;OF FOR RO
END ALPHA
;$RETURN EXAMPLES
$RETURN
$RETURN ERROR
$RETURN NOERROR
;CASE EXAMPLES
MOV    A,RO
CASE RO
    A
    B
    C
    D
    E
    F
END ;OF CASE RO
.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	BITS_=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 RO := 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	MOV 23,D
28	000122			END ;OF IF A
29	000122			IF B IS FALSE
30	000130	012737	000034	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	MOV C,D
35	000172			ELSE
36	000174	013737	000010	MOV E,D
37	000202			END ;OF IF A
38	000202			IF A EQ B AND C NE D
39	000222	013737	000012	MOV F,G
40	000230			END ;OF IF A
41	000230			IF A EQ B OR C NE D
42	000250	013737	000012	MOV F,G
43	000256			END ;OF IF A
44	000256			IFB A EQ B AND C EQ 1
45	000276	013737	000016	MOV H,J
46	000304			ELSE
47	000306	013737	000010	MOV E,J
48	000314			END ;OF IFB A
49	000314			IFB A EQ B ANDB C EQ 1
50	000334	013737	000016	MOV H,J
51	000342			ELSE
52	000344	013737	000010	MOV E,J
53	000352			END ;OF IFB A
54	000352			IF RESULT IS EQ
55	000354	013737	000000	MOV A,B
56	000362			END ;OF IF RESULT
57	000362			IF BITS SET.IN A
58	000372	013737	000002	MOV B,C
59	000400			END ;OF IF BITS
60	000400			IF BITS OFF.IN A
61	000410	013737	000004	MOV C,D
62	000416			END ;OF IF BITS
63				:ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
64				:ON.ERROR EXAMPLES
65	000416			ON.ERROR
66	000420	013737	000000	MOV A,B
67	000426			ELSE
68	000430	013737	000004	MOV C,B
69	000436			END ;OF ON.ERROR
70	000436			ON.NOERROR
71	000440	013737	000004	MOV C,B

72 000446  
73 000450 013737 000000 000002  
74 000456

ELSE  
MOV A,B  
END ;OF ON.NCERROR

F 7

SEQ 0083

.MAIN. MACRO M1.111 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      005277 000000      INC A
79 000500      END ;OF FOR I
80 000514      FOR RO : = 0 TO 140 BY 4
81 000516      005360 000000      DEC A(RO)
82 000522      END ;OF FOR RO
83 000534      FOR I :_ = 133 DOWNT0 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 RO :_ = 0 TO 167
89 000570      116037 000000 000002      MOV B A(RO),B
90 000576      IF B LT 0 THEN LEAVE ALPHA
91 000604      END ;OF FOR RO
92 000614      FOR RO : = 400 TO 567
93 000620      IF B GE 0 THEN LEAVE ALPHA
94 000626      END ;OF FOR RO
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,RO
102 000654      CASE RO
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 RO
110
111            000001      --      .END

```

7.4.3 Sample Listing File (with Expanded Macros) - -  
.MAIN. MACRO M1111 C1-APR-79 16:10 PAGE 2

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

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

18					;LET EXAMPLES
19	000024				LET R0 := A
	000024	013700	000000		MOV A,R0
20	000030				LET B := C + D
	000030	013737	000004	000002	MOV C,B
	000036	063737	000006	000002	ADD D,B
21	000044				LET E := F + 1
	000044	013737	000012	000010	MOV F,E
	000052	005237	000010		INC E
22	000056				LET G := H + 2
	000056	013737	000016	000014	MOV H,G
	000064	062737	000002	000014	ADD 2,G
23	000072				LET J := J + 01
	000072	062737	000001	000022	ADD 01,J
24	000100				LET A := B = B
	000100	113737	000002	000000	MOVB B,A
25					;IF EXAMPLES
26	000106				IF A IS TRUE
	000106	005737	000000		TST A
	000112	001403			BEQ L0
27	000114	012737	000023	000006	MOV 23,D
28	000122				END ;OF IF A
	000122				L0:
29	000122				IF B IS FALSE
	000122	005737	000002		TST B
	000126	001003			BNE L1
30	000130	012737	000034	000010	MOV 34,E
31	000136				END ;OF IF B
	000136				L1:
32	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:
33	000154				IF A LT B
	000154	023737	000000	000002	CMP A,B
	000162	002004			BGE L3
34	000164	013737	000004	000006	MOV C,D
35	000172				ELSE
	000172	000403			BR L4
	000174				L3:
36	000174	013737	000010	000006	MOV E,D
37	000202				END ;OF IF A
	000202				L4:
38	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
39	000222	013737	000012	000014	MOV F,G
40	000230				END ;OF IF A
	000230				L5:
41	000230				IF A EQ B OR C NE D
	000230	023737	000000	000002	CMP A,B

000236 001404  
000240 023737 000004 000006  
000246 001403

BEQ L6  
CMP C,D  
BFQ L7

J 7

SEQ 0087



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

42	000250	013737	000012	000014	L6:	MOV F,G
43	000256					END ;OF IF A
44	000256				L7:	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
47	000306	013737	000010	000022	L10:	MOV E,J
48	000314					END ;OF IFB A
49	000314				L11:	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
52	000344	013737	000010	000022	L12:	MOV E,J
53	000352					END ;OF IFB A
54	000352				L13:	IF RESULT IS EQ
	000352	001003				BNE L14
55	000354	013737	000000	000002		MOV A,B
56	000362					END ;OF IF RESULT
57	000362				L14:	IF BITS SET.IN A
	000362	032737	000040	000000		BIT BITS,A
	000370	001403				BEQ L15
58	000372	013737	000002	000004		MOV B,C
59	000400					END ;OF IF BITS
60	000400				L15:	IF BITS OFF.IN A
	000400	032737	000040	000000		BIT BITS,A
	000406	001003				BNE L16
61	000410	013737	000004	000006		MOV C,D
62	000416					END ;OF IF BITS
63	000416				L16:	;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 013757 000004 000002

MOV C,B

L 7

SEQ 0089

69 000436

END ;OF ON.EPROR

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	103603			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				F0:
80	000514				FOR RO :_ = 0 TO 140 BY 4
	000514	005000			CLR RO
	000516				B1:
81	000516	005360	000000		DEC A(RO)
82	000522				END ;OF FOR RO
	000522	062700	000004		ADD 4,RO
	000526	020027	000140		CMP RO, 140
	000532	003771			BLE B1
	000534				E1:
83	000534				FOR I :_ = 133 DOWNT0 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 RO :_ = 0 TO 167
	000566	005000			CLR RO
	000570				B4:
89	000570	116037	000000	000002	MOVB A(RO),B
90	000576				IF B LT 0 THEN LEAVE ALPHA
	000576	005737	000002		TST B
	000602	002415			BLT E3
91	000604				END ;OF FOR RO
	000604	005200			INC RO
	000606	020027	000167		CMP RO, 167
	000612	003766			BLE B4

000614  
92 000614  
000614 012700 000400

E4: N 7  
FOR R0 := 400 TO 567  
MOV 400,R0

SEQ 0091

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

```

000620
93 000620
000620 005737 000002
000624 002004
94 000626
000626 005200
000630 020027 000567
000634 003771
000636
95 000636
000636
96
97 000636
000636 000207
98 000640
000640 000261
000642 000207
99 000644
000644 000241
000646 000207
100
101 000650 013700 000000
102 000654
000654 010046
000656 006316
000660 004737 000700
103 000664 000000
104 000666 000002
105 000670 000004
106 000672 000006
107 000674 000010
108 000676 000012
109 000700
000700
000700 062616
000702 013646
000704 004736
110
111 000001

```

```

B5:
      IF B GE 0 THEN LEAVE ALPHA
      TST B
      BGE E3
      END :OF FOR RO
      INC RO
      CMP RO, 567
      BLE B5
E5:
      END ALPHA
E3:
; $RETURN EXAMPLES
      $RETURN
      RTS PC
      $RETURN ERROR
      SEC
      RTS PC
      $RETURN NOFROR
      CLC
      RTS PC
; CASE EXAMPLES
      MOV A, RO
      CASE RO
      MOV RO, -(SP)
      ASL @SP
      JSR PC, L24
      A
      B
      C
      D
      E
      F
      END :OF CASE RO
L24:
      ADD (SP)+, @SP
      MOV @ (SP)+, -(SP)
      JSR PC, @ (SP)+
      .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 (eq 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 W/o 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

23-	57	OPERATIONAL SWITCH SETTINGS
23-	58	;SWITCH REGISTER DEFINITIONS
23-	59	.*
23-	60	.* SWITCH USE
23-	61	.* -----
23-	62	.* 15 HALT ON ERROR
23-	63	.* 14 LOOP ON TEST
23-	64	.* 13 INHIBIT ERROR TYPEOUTS
23-	65	.* 12 INHIBIT RELOCATION
23-	66	.* 11 QUICK VERIFY
23-	67	.* 10 BELL ON ERROR
23-	68	.* 9 LOOP ON ERROR
23-	69	.* 8 HALT PROGRAM (UNRELOCATED & RESTORE LOADERS)
23-	70	.* 7 DETAILED ERROR REPORTS
23-	71	.* 6 INHIBIT CONFIGURATION MAP
23-	72	.* 5 LIMIT MAX ERRORS PER BANK
23-	73	.* 4 FAT TERMINAL (132 COLUMNS OR BETTER)
23-	74	.* 3 TEST MODE - SEE DOCUMENT
23-	75	.* 2 TEST MODE - SEE DOCUMENT
23-	76	.* 1 TEST MODE - SEE DOCUMENT
23-	77	.* 0 DETECT SINGLE BIT ERRORS
27-	103	DEFINE TRAPS
28-	218	DEFINE BASIC PDP11 STUFF
28-	301	DEFINE CACHE REGISTERS
28-	308	DEFINE CPU REGISTERS
28-	311	DEFINE MEMORY MANAGEMENT REGISTERS
30-	441	DEFINE UNIBUS MAP REGISTERS
30-	509	DEFINE SOFTWARE SWITCH & DISPLAY REGISTERS
30-	513	DEFINE CONTROL STATUS REGISTERS
30-	516	DEFINE PARAMETERS
32-	526	MACRO FATAL
32-	546	MACRO TYPE
34-	564	MACRO NEWTST
36-	614	MACRO \$\$NEWTEST
36-	635	MACRO SUBTST
36-	654	MACRO \$\$SUBTST
38-	669	MACRO TYPOCT
40-	709	MACRO TYPOCS
42-	765	MACRO TYPDEC
43-	807	MACRO BMOV
45-	873	MACRO MAP
47-	912	MACRO SUPERVISOR
47-	933	MACRO USER
48-	955	MACRO TESTAREA
50-	977	MACRO SET4 & RES4
52-	1022	MACRO DLEFT
54-	1046	TRAP CATCHER
54-	1054	ACT11 HOOKS
54-	1073	APT11 HOOKS
56-	1098	VARIABLES INITIALIZED TO ZERO
58-	1301	VARIABLES INITIALIZED TO NON ZERO
60-	1347	CONFIGURATION TABLE
61-	1374	***** MAIN *****
61-	1375	INITIALIZE VARIABLES TO ZERO
61-	1391	CLEAR NON-PROGRAM SPACE
62-	1411	TYPE OF SYSTEM SIZER
64-	1457	INITIALIZE VARIABLES TO NON ZERO

64-	1471	INITIALIZE VECTORS	
66-	1495	INITIALIZE PATTERNS	
66-	1524	SUBR PLUG IN NULL PATTERNS	
68-	1535	CLEAR THE CONFIGURATION TABLE	
68-	1547	SIZE FOR A HARDWARE SWITCH REGISTER	
70-	1565	SETUP ACT, APT, & XXDP	
71-	1590	PROTECT PROGRAM & LOADERS	
71-	1597	CHECK SYSTEM FOR CACHE	
72-	1672	SETUP USER & SUPERVISOR STACK	
72-	1690	GET SOFTWARE SWITCH REGISTER IF NECESSARY	
72-	1703	GET MEMORY MANAGEMENT READY	
74-	1715	T1 BIT TEST OF ALL CSR'S	
78-	1963	CLEAR ALL MEMORY SPACE FROM BANK 2 ON	
80-	1991	MATCH ALL CSR'S WITH MEMORY	
81-	2227	T2 TEST BANK 0 ACCESSES	
81-	2256	ENABLE ECC FOR CORRECT TRAPS	
83-	2264	T3 TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES	
84-	2389	FIND SHADOW INHIBIT MODE POINTERS	
86-	2412	T4 ECC INHIBIT MODE POINTER TEST	
96-	2896	LEGAL CONFIGURATION CHECK	
98-	3047	PRINT CONFIGURATION DETAILS	
100-	3124	CHECK APT SIZING	
101-	3170	T5 DIAGNOSTIC MODE DISPATCH ROUTINE	
101-	3187	T6 UNIQUE BANK TEST	
101-	3201	FLUSH OUT DBE'S	
103-	3205	END OF PASS ROUTINE	
105-	3267	WRITE BACKGROUND PATTERNS	
107-	3281	MTEST MODES	
107-	3283	BANKS FORWARD, PATTERNS FORWARD	**RECURSIVE**
109-	3313	BANKS FORWARD, PATTERNS REVERSE	**RECURSIVE**
111-	3343	BANKS WORST FIRST, PATTERNS FORWARD	**RECURSIVE**
113-	3380	BANKS WORST FIRST, PATTERNS REVERSE	**RECURSIVE**
115-	3417	PATTERNS FORWARD, BANKS FORWARD	**RECURSIVE**
117-	3455	PATTERNS FORWARD, BANKS WORST FIRST	**RECURSIVE**
119-	3500	PATTERNS REVERSE, BANKS FORWARD	**RECURSIVE**
121-	3538	PATTERNS REVERSE, BANKS WORST FIRST	**RECURSIVE**
123-	3583	SUBR SETUP MEMORY TEST	
125-	3603	SUBR TEST ECC CSR LOGIC DISPATCH	
127-	3692	CHECK FOR SBE FREE LOCATIONS	
129-	3787	CSR PATTERN CASE STATEMENT	
131-	3821	SUBR ECC TEST DISPATCH	
133-	3879	SUBR PARITY TEST DISPATCH	
134-	3929	PATTERNS	
134-	3931	MEMORY TEST SETUP ROUTINES	
134-	3932	MT0000 SETUP DATA PATTERN TEST	
134-	3945	MT0001 SETUP ADDRESS TEST	
134-	3967	MT0002 SETUP COMPLEMENT ADDRESS TEST	
136-	3995	MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST	
136-	4031	MT0004 SETUP ROTATING ZEROS TEST	
136-	4049	MT0005 SETUP ROTATING ONES TEST	
138-	4071	MT0006 SETUP INITIAL DATA TEST	
138-	4078	MT0007 SETUP ADDRESS BIT TEST	
138-	4088	MT0010 SETUP BYTE ADDRESSING TEST	
140-	4097	MT0011 SETUP CREATE SINGLE BIT ERROR TEST	
140-	4105	MT0012 SETUP WRITE BYTE CLEARS SBE TEST	
140-	4119	MT0013 SETUP CREATE DOUBLE BIT ERROR TEST	
140-	4128	MT0014 SETUP WRITE INHIBIT DURING DATIP WITH DBE	



142- 4139	MT0015	SETUP WRITE INHIBIT OF BYTE WITH DBE
142- 4147	MT0016	SETUP WRITE INHIBIT OF WORD WITH DBE
142- 4155	MT0017	SETUP HOLDING 1'S & 0'S
144- 4162	MT0020	SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST
147- 4307	MT0021	SETUP MARCHING 0'S & 1'S TEST
148- 4354	MT0022	SETUP REFRESH & SHIFTING DIAGONAL TEST
148- 4362	MT0023	SHIFTING DIAGONAL TEST
149- 4372	MT0024	SETUP FAST GALLOPING PATTERN TEST
149- 4414	MT0025	SETUP INTERRUPT ENABLE TEST
151- 4424	MT0026	SETUP RANDOM DATA TEST
153- 4471	MT0027	UNIQUE BANK TEST
155- 4542	MT0030	SETUP FLUSH OUT DBE'S TEST
157- 4587	MT0031	SETUP SOB-A-LONG TEST
159- 4616	MT0032	SETUP WRITE RECOVERY TEST
161- 4680	MT0033	SETUP BRANCH GOBBLE TEST
161- 4710	MT0034	SOFT ERROR - BACKGROUND PATTERN TEST
161- 4740	MT0035	SETUP WORST CASE NOISE PARITY TEST
163- 4762	MT0999	SETUP NULL TEST
163- 4767		CHECK FOR KAMIKAZE MODE
165- 4775	SUBR	EXECUTE PATTERN IN SUPERVISOR
169- 4846	MEMORY	TEST PATTERN ROUTINES
169- 4856	MTP000	BASIC DATA TEST
169- 4867	MTP001	ADDRESS TEST
169- 4879	MTP002	COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
171- 4893	MTPA03	3 XOR 9 WORST CASE NOISE TEST (WRITE)
171- 4916	MTPB03	3 XOR 9 WORST CASE NOISE TEST (READ)
173- 4934	MTPC03	TEST DATA SUBPROGRAM
173- 4942	MTPD03	TEST DATA SUBSUBPROGRAM
175- 4952	MTPA04	ROTATING ZEROS TEST
175- 4965	MTPB04	SUBR ROTATING BIT
175- 4974	MTP005	ROTATION ONES TEST
177- 4988	MTP006	INITIAL DATA TEST
179- 5031	MTP007	ADDRESS BIT TEST
181- 5071	MTP010	BYTE ADDRESSING TEST
183- 5107	MTP011	SINGLE BIT ERROR TEST
185- 5248	MTP012	WRITE BYTE CLEARS SBE TEST
187- 5330	MTP013	CREATE DOUBLE BIT ERROR TEST
191- 5420	MTP014	WRITE INHIBIT DURING DATIP WITH DBE TEST
193- 5528	MTP015	WRITE INHIBIT OF BYTE WITH DBE
195- 5627	MTP016	WRITE INHIBIT OF WORD WITH DBE
199- 5729	MTP017	HOLDING 1'S & 0'S TEST
201- 5762	MTP020	MARCHING 1'S & 0'S IN CHECKBITS TEST
205- 5836	MTPA21	MARCHING 1'S & 0'S PATTERN TEST
209- 5906	MTP022	REFRESH & SHIFTING DIAGONAL TEST
210- 5978	SUBR	REFRESH DELAY
213- 6001	MTPA24	FAST GALLOPING PATTERN TEST
215- 6045	MTPB24	FAST GALLOP PART B
215- 6053	MTPC24	FAST GALLOP PART C
217- 6063	MTP025	INTERRUPT ENABLE TEST
221- 6157	MTPA26	RANDOM DATA (WRITE)
221- 6164	MTPB26	RANDOM DATA (READ)
221- 6182	RANDOM	NUMBER SUBPROGRAM
221- 6195	RANDOM	NUMBER SUBSUBPROGRAM
223- 6203	MT0030	FLUSH OUT DBE'S
223- 6209	MTP031	SOB-A-LONG TEST
225- 6260	MTP032	WRITE RECOVERY TEST
227- 6280	MTP033	BRANCH GOBBLE TEST

228- 6326	MTPO34	SOFT ERROR - BACKGROUND PATTERN TEST
229- 6338	MTPO35	WORST CASE NOISE PARITY TEST
230- 6370	MISC	SUBROUTINES
230- 6372	SUBR	COPY R0 TO R4, R1 TO R3, & R2 TO R5
230- 6378	FLIP	WARNING CONSTANTS IN WORST CASE NOISE TESTS
231- 6405	SUBR	WRITE BACKGROUND
233- 6425	SUBR	PRINT CONFIGURATION MAP
235- 6477	SUBR	TYPE CONFIGURATION
239- 6637	TRAP	PARITY ERROR HANDLER
241- 6669	TRAP	NON-EXISTANT MEMOPY (HOLES) HANDLER
241- 6689	TRAP	TIMEOUT (TRAP TO 4) HANDLER
241- 6693	TRAP	MEMORY MANAGEMENT (TRAP TO 250) HANDLER
241- 6696	TRAP	RESERVED INSTRUCTION HANDLER
241- 6706	FIND	BAD SP, PC, & PSW FROM STACK
243- 6714	TRAP	KERNEL TRAP HANDLER
243- 6722	TRAP	ENERGIZE TRAP HANDLER
243- 6726	TRAP	DEENERGIZE TRAP HANDLER
243- 6730	TRAP	CACHON TRAP HANDLER
243- 6737	TRAP	CACHOFF TRAP HANDLER
245- 6745	TRAP	LOAD CSR TRAP HANDLER
245- 6764	TRAP	READ CSR TRAP HANDLER
246- 6772	TRAP	TEST (R1) & READ CSR CAREFULLY
248- 6809	TRAP	ECC DISABLE ALL CSR'S TRAP HANDLER
248- 6813	TRAP	ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
248- 6817	TRAP	INITIALIZE ALL CSR'S TRAP HANDLER
248- 6821	TRAP	INITIALIZE 1 SELECTED CSR TRAP HANDLER
248- 6825	TRAP	ENABLE SBE PARITY TRAPS ON ALL CSR'S
248- 6829	TRAP	ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
248- 6833	TRAP	WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
248- 6838	TRAP	WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
250- 6845	TRAP	WAS THERE A SBE ON ANY CSR TRAP HANDLER
250- 6870	TRAP	WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
252- 6880	TRAP	WAS THERE A DBE ON ANY CSR TRAP HANDLER
252- 6905	TRAP	WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
254- 6916	TRAP	CLEAR ALL ECC CSR'S TRAP HANDLER
254- 6920	TRAP	CLEAR 1 SELECTED CSR TRAP HANDLER
254- 6924	TRAP	ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
254- 6929	TRAP	ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
256- 6936	SUBR	WRITE IN ALL CSR'S
256- 6951	TRAP	INVALIDATE BACKGROUND PATTERN
257- 6960	TRAP	GENERATE AND TEST ERROR ADDRESS
259- 7015	SUBR	GENERATE CHECK BITS
263- 7084	SUBR	MAPPER
263- 7169	TRAP	MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
265- 7192		RELOCATE PROGRAM
267- 7298		UNRELOCATE PROGRAM
267- 7343		SETUP LOWER 16K OF UNIBUS MAP
269- 7356		MOVE BANKS
271- 7404	SUBR	MAP USER TO NEW BANK
271- 7424	SUBR	SETUP KERNEL PAR'S FOR NEW BANK
271- 7437	SUBR	SETUP KERNEL PAR'S FOR NEW LOADER BANK
273- 7449	SUBR	EXAMINE BANK
275- 7540	SUBR	BANK OK?
275- 7551	SUBR	INCREMENT PATTERN TESTING
275- 7559	SUBR	SET HIGHEST PATTERN TESTING TYPE
275- 7563	SUBR	INCREMENT BANK & TEST
277- 7570		BOOTSTRAP ROUTINE

279- 7599	HALT PROGRAM
279- 7608	SHUTDOWN DIAGNOSTIC
279- 7635	APT SHUTDOWN SEQUENCE
281- 7645	BLOCK MOVE SUBROUTINE
282- 7672	FIELD SERVICE MODE
282- 7674	SUBR    FIELD SERVICE COMMAND MODE
284- 7724	COMMAND 0    EXIT
284- 7746	FS        COMMAND 1        READ CSR
286- 7761	FS        COMMAND 2        LOAD CSR
288- 7785	FS        COMMAND 3        EXAMINE MEMORY
290- 7827	FS        COMMAND 4        MODIFY MEMORY
292- 7879	FS        COMMAND 5        SELECT BANK & PATTERN
293- 7991	FS        COMMAND 6        TYPE CONFIGURATION MAP
295- 7997	FS        COMMAND 7        SOB-A-LONG TEST
297- 8038	FS        COMMAND 8        ERROR SUMMARY
299- 8068	FS        COMMAND 9        REFRESH TEST
301- 8109	FS        COMMAND 10       SET FILL COUNT
301- 8119	FS        COMMAND 11       ENTER KAMIKAZE MODE
301- 8124	FS        COMMAND 12       EXIT KAMIKAZE MODE
301- 8130	FS        COMMAND 13       TURN CACHE OFF
301- 8137	FS        COMMAND 14       TURN CACHE ON
302- 8156	FS        COMMAND 15       TEST ONLY SELECTED BANKS
302- 8176	FS        COMMAND 16       RESUME TESTING ALL BANKS
304- 8190	FS        COMMAND 17       ENABLE TRACE
306- 8196	FS        COMMAND 18       DISABLE TRACE
308- 8202	SUBR    DETERMINE CORRECT CSR
323- 8770	ERROR DATA (SUPERVISOR) SETUP STUFF
323- 8784	DATA WAS 3 WORDS
325- 8825	GET DATA FROM ABORTED AREA IF POSSIBLE
327- 8841	POWER FAIL AUTO RESTART
327- 8842	ROUTINE POWER DOWN AND UP
332- 9030	POWER FAIL WHILE RELOCATED
334- 9057	POWER UP FROM BANK 0 TO RELOCATION
336- 9097	IO SUBROUTINES
336- 9099	ROUTINE TYPE
351- 9892	ERROR DATA SETUP
356-10141	DATA WAS A WORD
356-10153	DATA WAS A BYTE
358-10166	DATA WAS A 7 BIT BYTE
358-10181	DETERMINE XOR OF GOOD & BAD
360-10190	LOG ERROR ON BAD BANK
364-10279	ROUTINE SCOPE HANDLER
365-10343	SUBR    DISPLAY
367-10360	ROUTINE ERROR HANDLER
370-10451	ROUTINE ERROR MESSAGE TYPEOUT
378-10666	SUBR    DETAILED ERROR REPORT
383-10808	ROUTINE BINARY TO OCTAL (ASCII) AND TYPE
384-10886	ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
385-10943	ROUTINE TTY INPUT
387-11038	CONTROL T
387-11063	CONTROL S & CONTROL Q
389-11181	ROUTINE READ AN OCTAL NUMBER FROM THE TTY
389-11230	ROUTINE READ A DECIMAL NUMBER FROM THE TTY
390-11289	ROUTINE SAVE AND RESTORE R0-R5
391-11325	ROUTINE RANDOM NUMBER GENERATOR
393-11355	ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
394-11397	TABLES

394-11399	APT MAILBOX-ETABLE
396-11481	ROUTINE TRAP DECODER
398-11508	TRAP TABLE
402-11611	TABLE    ERROR POINTER
412-11912	ERROR DATA TAGS (DT)
414-11951	ERROR DATA FORMATS (DF)
416-11969	ERROR MESSAGES (EM)
418-12039	ERROR DATA HEADERS (DH)
420-12073	MESSAGES

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33

.TITLE CZMSDBO MS11-L/M DIAGNOSTIC  
.REM &

IDENTIFICATION  
-----

PRODUCT CODE: AC-F294B-MC  
PRODUCT NAME: CZMSDBO 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

&

36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55

REVISION HISTORY  
=====

REVISION	DATE	AUTHOR	CHANGES
=====	=====	=====	=====
CZMSDA	01-DEC-79	MICHAEL D BIBEALT	NONE=NEW PROGRAM
CZMSDB	01-OCT-80	MICHAEL D BIBEALT	1) COMPATIBLE WITH 11/24 2) SIZNG ROUTINE WILL ACCEPT ALL LEGAL MEMORY CONFIGURATIONS 3) ALL FIELD SERVICE COMMANDS OPERATIVE

57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77

```
.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
```

```

80 000000      .ENABL  ABS
81             .ENABL  AMA
82             .DSABL  GBL
83             ;NOTE:  CZMSDB.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
84             ;THIS PROGRAM.  ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
85             .MCALL  SMACIT,..PUSH,..POP,..TAG,..BRAN,..EMIT,..EMITN,..EMITL,..EMITR
86             .MCALL  .IFOPR,..IS,..GENBR,..OPADD,..OPSUB,CLEAR,SET,CLEARB,SETB
87             .MCALL  RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
88             .MCALL  IF,..OR,..IFARI,..LEAVE,..GOTO,OR,AND,THEN,ELSE,WHILE,CASE
89             .MCALL  FOR,TO,DOWNT0,REPEAT,UNTIL,THRU,END,BEGIN
90             .MCALL  $$END,LEAVE,JUMPTO,GOTC,PUSH,POP,LET
91             .MCALL  .SIMPLE,..ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
92             .MCALL  $CALL,$RETURN
93
94             .NLIST  TTM                   ;I WANT-FAT PAPER.
95             .LIST  MC,SYM                ;LIST MACRO CALLS, SYMBOL TABLE
96             .NLIST MD,CND,ME            ;DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
97             ;LST$$= 0                     ;DEFINED TO LIST SUPERMAC EXPANSIONS
98             $$SWR= 163000               ;USE THESE SYSMAC SWITCHES
99             $TN= 1                       ;FIRST TEST NUMBER TO ONE(1)
100 000000    SMACIT

```

```

163000
000001

```



103  
 104  
 105  
 106  
 107  
 108  
 109  
 110  
 111  
 112  
 113  
 114  
 115  
 116  
 117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142  
 143  
 144  
 145  
 146  
 147  
 148  
 149  
 150  
 151  
 152  
 153  
 154  
 155  
 156  
 157  
 158  
 159

```

.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
TYPEIT= 104401      ;;ITY TYPEOUT ROUTINE
TYPOC=  104402      ;;TYPE OCTAL NUMBER (WITH LEADING ZEROS)
TYPOS=  104403      ;;TYPE OCTAL NUMBER (NO LEADING ZEROS)
:TYPON= 104404      ;;TYPE OCTAL NUMBER (AS PER LAST CALL)
TYPDS=  104405      ;;TYPE DECIMAL NUMBER (WITH SIGN)
:TYPBN= 104406      ;;TYPE BINARY (ASCII) NUMBER
:
GTSWR=  104407      ;;GET SOFT-SWR SETTING
CKSWR=  104410      ;;TEST FOR CHANGE IN SOFT-SWR
:
RDCHR=  104411      ;;TTY TYPEIN CHARACTER ROUTINE
RDLIN=  104412      ;;TTY TYPEIN STRING ROUTINE
RDOCT=  104413      ;;READ AN OCTAL NUMBER FROM TTY
RDDEC=  104414      ;;READ A DECIMAL NUMBER FROM TTY
:
SAVREG= 104415      ;;SAVE R0-R5 ROUTINE
RESREG= 104416      ;;RESTORE R0-R5 ROUTINE
:
KERNEL= 104417      ;ENTER KERNEL MODE
:
ENERGIZE=104420      ;TURN ON MEMORY MANAGEMENT & TRAPS
DEENERGIZE=104421   ;TURN OFF MEMORY MANAGEMENT & TRAPS
KMAP=    104422      ;MAP KERNEL 1 TO 1
:
CACHON= 104423      ;TURN ON CACHE
CACHOFF=104424     ;TURN OFF CACHE

```

160			
161	104425	LOADCSR=104425	;LOAD CORRECT CSR
162	104426	READCSR=104426	;READ CORRECT CSR
163			
164	104427	PERR01= 104427	;PROGRAM DETECTED ERROR
165	104430	PERR02= 104430	;PROGRAM DETECTED ERROR
166	104431	PERR03= 104431	;PROGRAM DETECTED ERROR
167	104432	PERR04= 104432	;PROGRAM DETECTED ERROR
168	104433	PERR07= 104433	;PROGRAM DETECTED ERROR
169	104434	PERR10= 104434	;PROGRAM DETECTED ERROR
170	104435	PERR11= 104435	;PROGRAM DETECTED ERROR
171	104436	PERR12= 104436	;PROGRAM DETECTED ERROR
172	104437	PERR13= 104437	;PROGRAM DETECTED ERROR
173	104440	PERR14= 104440	;PROGRAM DETECTED ERROR
174	104441	PERR15= 104441	;PROGRAM DETECTED ERROR
175	104442	PERR16= 104442	;PROGRAM DETECTED ERROR
176	104443	PERR17= 104443	;PROGRAM DETECTED ERROR
177	104444	PERR20= 104444	;PROGRAM DETECTED ERROR
178	104445	PERR21= 104445	;PROGRAM DETECTED ERROR
179	104446	PERR22= 104446	;PROGRAM DETECTED ERROR
180	104447	PERR23= 104447	;PROGRAM DETECTED ERROR
181	104450	PERR24= 104450	;PROGRAM DETECTED ERROR
182	104451	PERR25= 104451	;PROGRAM DETECTED ERROR
183	104452	PERR26= 104452	;PROGRAM DETECTED ERROR
184	104453	PERR27= 104453	;PROGRAM DETECTED ERROR
185	104454	PERR30= 104454	;PROGRAM DETECTED ERROR
186	104455	PERR31= 104455	;PROGRAM DETECTED ERROR
187	104456	PERR32= 104456	;PROGRAM DETECTED ERROR
188	104457	PERR33= 104457	;PROGRAM DETECTED ERROR
189	104460	PERR34= 104460	;PROGRAM DETECTED ERROR
190	104461	PERR35= 104461	;PROGRAM DETECTED ERROR
191	104462	PERR36= 104462	;PROGRAM DETECTED ERROR
192	104463	PERR37= 104463	;PROGRAM DETECTED ERROR
193	104464	PERR40= 104464	;PROGRAM DETECTED ERROR
194	104465	PERR41= 104465	;PROGRAM DETECTED ERROR
195	104466	PERR42= 104466	;PROGRAM DETECTED ERROR
196	104467	PERR43= 104467	;PROGRAM DETECTED ERROR
197			
198	104470	ECCDIS= 104470	;DISABLE ECC ON ALL CSR'S
199	104471	ECC1DIS=104471	;DISABLE ECC ON 1 SELECTED CSR
200	104472	ECCINIT=104472	;INITIALIZE ALL ECC CSR'S
201	104473	ECC1INIT=104473	;INITIALIZE 1 SELECTED ECC CSR
202	104474	CBCSR= 104474	;WRITE GENERATED CHECKBITS IN ALL CSR'S
203	104475	CB1CSR= 104475	;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
204	104476	WASSBE= 104476	;WAS THERE A SBE ON ANY CSR?
205	104477	WAS1SBE=104477	;WAS THERE A SBE ON 1 SELECTED CSR?
206	104500	WASDBE= 104500	;WAS THERE A DBE ON ANY CSR?
207	104501	WAS1DBE=104501	;WAS THERE A DBE ON 1 SELECTED CSR?
208	104502	CLRCSR= 104502	;CLEAR ALL CSR'S
209	104503	CLR1CSR=104503	;CLEAR 1 SELECTED CSR
210	104504	CHKDIS= 104504	;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
211	104505	CHK1DIS=104505	;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
212	104506	ENASBE= 104506	;ENABLE TRAPS ON SBE'S FROM ALL CSR'S
213	104507	ENA1SBE=104507	;ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
214	104510	TSTREAD=104510	;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
215	104511	INVALID=104511	;INVALIDATE BACKGROUND PATTERN ON 'BANK'
216	104512	ERRGEN =104512	;CHECK ERROR ADDRESS

```

218          .SBTTL DEFINE BASIC PDP11 STUFF
219
220          ;*INITIAL ADDRESS OF THE STACK POINTER
221          STACK= 2000          ;;FIRST ADDRESS OF THE STACK
222          KERSTK= STACK        ;;KERNEL STACK
223          SUPSTK= 740         ;;SUPERVISOR STACK
224          USESTK= 700         ;;USER STACK
225          ERROR=EMT          ;;BASIC DEFINITION OF ERROR CALL
226          SCOPE=IOT          ;;BASIC DEFINITION OF SCOPE CALL
227          PSW= 177776        ;;PROCESSOR STATUS WORD
228          ;STKLMT=177774      ;;STACK LIMIT REGISTER
229          ;PIRQ= 177772       ;;PROGRAM INTERRUPT REQUEST REGISTER
230          DSWR= 177570        ;;HARDWARE SWITCH REGISTER
231          DDISP= 177570       ;;HARDWARE DISPLAY REGISTER
232          LKS= 177546        ;;LINE CLOCK (KW11-L) STATUS REGISTER
233
234          ;*MISCELLANEOUS DEFINITIONS
235          HT= 11              ;;CODE FOR HORIZONTAL TAB
236          LF= 12              ;;CODE LINE FEED
237          CR= 15              ;;CODE CARRIAGE RETURN
238          CRLF= 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED
239          MFPT= 7             ;;CODE FOR PROCESSOR TYPE INSTRUCTION
240
241          ;*GENERAL PURPOSE REGISTER DEFINITIONS
242          ;SP=R6               ;;STACK POINTER
243          ;KSP=SP              ;;KERNEL STACK POINTER
244          SSP=SP               ;;SUPERVISOR STACK POINTER
245          USP=SP               ;;USER STACK POINTER
246          ;PC=R7              ;;PROGRAM COUNTER
247
248          ;*'SWITCH REGISTER' SWITCH DEFINITIONS
249          SW15= 100000
250          SW14= 40000
251          SW13= 20000
252          SW12= 10000
253          SW11= 4000
254          SW10= 2000
255          SW9= 1000
256          SW8= 400
257          SW7= 200
258          SW6= 100
259          SW5= 40
260          SW4= 20
261          SW3= 10
262          SW2= 4
263          SW1= 2
264          SW0= 1
265
266          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
267          BIT15= 100000
268          BIT14= 40000
269          BIT13= 20000
270          BIT12= 10000
271          BIT11= 4000
272          BIT10= 2000
273          BIT9= 1000
274          BIT8= 400
  
```

```

275      000200      BIT7= 200
276      000100      BIT6= 100
277      000040      BIT5= 40
278      000020      BIT4= 20
279      000010      BIT3= 10
280      000004      BIT2= 4
281      000002      BIT1= 2
282      000001      BIT0= 1
283
284      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
285      000004      ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
286      000010      RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
287      ;TBITVEC=14          ;;"T" BIT
288      ;TRTVEC= 14          ;;TRACE TRAP
289      ;BPTVEC= 14          ;;BREAKPOINT TRAP (BPT)
290      000020      IOTVEC= 20         ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
291      000024      PWRVEC= 24         ;;POWER FAIL
292      000030      EMTVEC= 30         ;;EMULATOR TRAP (EMT) **ERROR**
293      000034      TRAPVEC=34         ;;"TRAP" TRAP
294      000060      TKVEC= 60          ;;TTY KEYBOARD VECTOR
295      ;TPVEC= 64          ;;TTY PRINTER VECTOR
296      ;LKVEC= 100         ;;LINE CLOCK (KW11-L) VECTER
297      000114      CACHVEC=114        ;;CACHE ERROR INTERRUPT VECTOR
298      000114      PARVEC=CACHVEC
299      ;PIRQVEC=240        ;;PROGRAM INTERRUPT REQUEST VECTOR
300      000250      MMVEC= 250         ;;MEMORY MANAGEMENT VECTOR
301      .SBTTL DEFINE CACHE REGISTERS
302      ;MEMERR = 177744      ;;CACHE ERROR REGISTER
303      177746      CONTRL = 177746    ;;MEMORY CONTROL REGISTER
304      177750      MAINT = 177750    ;;MEMORY MAINTENENCE REGISTER
305      ;HITMIS = 177752    ;;HIT MISS REGISTER "1" IMPLIES HIT IN CACHE
306      177754      DATARG = 177754   ;;DATA REGISTER
307
308      .SBTTL DEFINE CPU REGISTERS
309      177766      CPUERR = 177766    ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
310
311      .SBTTL DEFINE MEMORY MANAGEMENT REGISTERS
312      ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
313      177572      MMR0= 177572
314      177574      MMR1= 177574
315      177576      MMR2= 177576
316      172516      MMR3= 172516
317
318      ;*USER "I" PAGE DESCRIPTOR REGISTERS
319      177600      UIPDR0= 177600
320      ;UIPDR1= 177602
321      ;UIPDR2= 177604
322      ;UIPDR3= 177606
323      ;UIPDR4= 177610
324      ;UIPDR5= 177612
325      ;UIPDR6= 177614
326      ;UIPDR7= 177616
327
328      ;*USER "D" PAGE DESCRIPTOR REGISTORS
329      ;UDPDR0= 177620
330      ;UDPDR1= 177622
331      ;UDPDR2= 177624
  
```

```

332      ;UDPDR3=      177626
333      ;UDPDR4=      177630
334      ;UDPDR5=      177632
335      ;UDPDR6=      177634
336      ;UDPDR7=      177636
337
338      ;*USER 'I' PAGE ADDRESS REGISTERS
339      177640 FASTCITY=UIPAR0
340      177640 UIPAR0= 177640      ;PATTERN PROGRAM SPACE
341      177642 UIPAR1= 177642      ;PATTERN PROGRAM SPACE
342      177644 UIPAR2= 177644      ;PATTERN PROGRAM SPACE
343      177646 UIPAR3= 177646      ;PATTERN PROGRAM SPACE
344      177650 UIPAR4= 177650      ;PATTERN PROGRAM SPACE
345      177652 UIPAR5= 177652      ;PATTERN PROGRAM SPACE
346      177654 UIPAR6= 177654      ;PATTERN PROGRAM SPACE
347      ;UIPAR7=      177656      ;PATTERN PROGRAM SPACE
348
349      ;*USER 'D' PAGE ADDRESS REGISTERS
350      177660 UDPAR0= 177660      ;PATTERN PROGRAM SPACE
351      ;UDPAR1=      177662      ;PATTERN PROGRAM SPACE
352      ;UDPAR2=      177664      ;PATTERN PROGRAM SPACE
353      ;UDPAR3=      177666      ;PATTERN PROGRAM SPACE
354      ;UDPAR4=      177670      ;PATTERN PROGRAM SPACE
355      ;UDPAR5=      177672      ;PATTERN PROGRAM SPACE
356      ;UDPAR6=      177674      ;PATTERN PROGRAM SPACE
357      177676 UDPAR7= 177676      ;PATTERN PROGRAM SPACE
358
359      ;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
360      172200 SIPDR0= 172200
361      ;SIPDR1=      172202
362      ;SIPDR2=      172204
363      ;SIPDR3=      172206
364      ;SIPDR4=      172210
365      ;SIPDR5=      172212
366      ;SIPDR6=      172214
367      ;SIPDR7=      172216
368
369      ;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
370      ;SDPDR0=      172220
371      ;SDPDR1=      172222
372      ;SDPDR2=      172224
373      ;SDPDR3=      172226
374      ;SDPDR4=      172230
375      ;SDPDR5=      172232
376      ;SDPDR6=      172234
377      ;SDPDR7=      172236
378
379      ;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
380      172240 SIPAR0= 172240
381      ;SIPAR1=      172242
382      ;SIPAR2=      172244
383      172246 SIPAR3= 172246      ;TEST AREA
384      ;SIPAR4=      172250      ;TEST AREA
385      172252 SIPAR5= 172252      ;TEST AREA
386      172254 SIPAR6= 172254      ;TEST AREA
387      ;SIPAR7=      172256
388
    
```

```
389                                     ;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
390      172260      SDPAR0= 172260
391                                     ;SDPAR1=      172262
392                                     ;SDPAR2=      172264
393                                     ;SDPAR3=      172266
394                                     ;SDPAR4=      172270
395      172272      SDPAR5= 172272
396      172274      SDPAR6= 172274
397      172276      SDPAR7= 172276
398
399                                     ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
400      172300      KIPDR0= 172300
401                                     ;KIPDR1=      172302
402                                     ;KIPDR2=      172304
403                                     ;KIPDR3=      172306
404                                     ;KIPDR4=      172310
405                                     ;KIPDR5=      172312
406                                     ;KIPDR6=      172314
407                                     ;KIPDR7=      172316
408
409                                     ;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
410      ;KDPDR0=      172320
411      ;KDPDR1=      172322
412      ;KDPDR2=      172324
413      ;KDPDR3=      172326
414      ;KDPDR4=      172330
415      ;KDPDR5=      172332
416      ;KDPDR6=      172334
417      ;KDPDR7=      172336
418
419                                     ;*KERNEL 'I' PAGE ADDRESS REGISTERS
420      172340      KIPAR0= 172340
421      ;KIPAR1=      172342
422      ;KIPAR2=      172344
423      ;KIPAR3=      172346
424      172350      KIPAR4= 172350
425      172352      KIPAR5= 172352
426      172354      KIPAR6= 172354
427      ;KIPAR7=      172356
428
429                                     ;*KERNEL 'D' PAGE ADDRESS REGISTERS
430      172360      KDPAR0= 172360
431      ;KDPAR1=      172362
432      ;KDPAR2=      172364
433      ;KDPAR3=      172366
434      ;KDPAR4=      172370
435      ;KDPAR5=      172372
436      172374      KDPAR6= 172374
437      172376      KDPAR7= 172376
438
```

```
441 .SBTTL DEFINE UNIBUS MAP REGISTERS
442 ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
443 ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
444 170200 MAPL0 = 170200
445 170202 MAPH0 = 170202
446 170204 MAPL1 = 170204
447 ;MAPH1 = 170206
448 ;MAPL2 = 170210
449 ;MAPH2 = 170212
450 ;MAPL3 = 170214
451 ;MAPH3 = 170216
452 ;MAPL4 = 170220
453 ;MAPH4 = 170222
454 ;MAPL5 = 170224
455 ;MAPH5 = 170226
456 ;MAPL6 = 170230
457 ;MAPH6 = 170232
458 ;MAPL7 = 170234
459 ;MAPH7 = 170236
460 ;MAPL10 = 170240
461 ;MAPH10 = 170242
462 ;MAPL11 = 170244
463 ;MAPH11 = 170246
464 ;MAPL12 = 170250
465 ;MAPH12 = 170252
466 ;MAPL13 = 170254
467 ;MAPH13 = 170256
468 ;MAPL14 = 170260
469 ;MAPH14 = 170262
470 ;MAPL15 = 170264
471 ;MAPH15 = 170266
472 ;MAPL16 = 170270
473 ;MAPH16 = 170272
474 ;MAPL17 = 170274
475 ;MAPH17 = 170276
476 ;MAPL20 = 170300
477 ;MAPH20 = 170302
478 ;MAPL21 = 170304
479 ;MAPH21 = 170306
480 ;MAPL22 = 170310
481 ;MAPH22 = 170312
482 ;MAPL23 = 170314
483 ;MAPH23 = 170316
484 ;MAPL24 = 170320
485 ;MAPH24 = 170320
486 ;MAPL25 = 170324
487 ;MAPH25 = 170326
488 ;MAPL26 = 170330
489 ;MAPH26 = 170332
490 ;MAPL27 = 170334
491 ;MAPH27 = 170336
492 ;MAPL30 = 170340
493 ;MAPH30 = 170342
494 ;MAPL31 = 170344
495 ;MAPH31 = 170346
496 ;MAPL32 = 170350
497 ;MAPH32 = 170352
```

```
498 ;MAPL33 = 170354
499 ;MAPH33 = 170356
500 ;MAPL34 = 170360
501 ;MAPH34 = 170362
502 ;MAPL35 = 170364
503 ;MAPH35 = 170366
504 ;MAPL36 = 170370
505 ;MAPH36 = 170372
506 ;MAPL37 = 170374
507 ;MAPH37 = 170376
508
509 .SBTTL DEFINE SOFTWARE SWITCH & DISPLAY REGISTERS
510 000174 DISPREG=174
511 000176 SWREG= 176
512
513 .SBTTL DEFINE CONTROL STATUS REGISTERS
514 172100 CSRADD=172100
515
516 .SBTTL DEFINE PARAMETERS
517 060000 FIRST=60000 ;START OF THE 16K TEST PATTERN AREA
518 157776 LAST=157776 ;END OF THE 16K TEST PATTERN AREA
519 040000 SIZE=40000 ;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)
```



525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561

```
.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$$ .LIST ME
.ENABL CRF
.LIST
INC FATALS ;SET FATAL INDICATOR
ERROR +ARG
.DSABL CRF
.IIF DF LST$$ .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
```

564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611

```
.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>
$$NEWTST \ $TN,<ASCII>,SCOPE
.IFF
$$NEWTST \ $TN,<ASCII>,<COMAND>
.ENDC
.NLIST
.LIST ME
.LIST
.IF NE 4000&$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
```

614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666

```
.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
```

669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706

```
.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
```

709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762

```
.SBTTL MACRO TYPCCS
***** TYPCCS *****
TYPCCS 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) TYPCCS #12345,<TYPES '5'>,1
2) TYPCCS #004,<TYPES '04'>,2,X
3) TYPCCS #004,<TYPES ' 4'>,2
*****

.MACRO TYPCCS 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 TYPCCS
```

765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805

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

807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863

```
.SBTTL MACRO BMOV
***** BMOV *****
: THIS MACRO MOVES A BLOCK OF DATA.
: ARGUMENTS:
: 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
```

864  
865  
866  
867  
868  
869  
870

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



873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909

```
.SBTTL MACRO MAP
***** MAP *****
: THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
: TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-15777)).
: ARGUMENTS:
: 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
```

912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953

```
.SBTTL MACRO SUPERVISOR
:***** SUPERVISOR *****
:
: THIS MACRO SWITCHES TO SUPERVISOR MODE.
:
: ARGUMENTS: 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.
:
: ARGUMENTS: 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
```

955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974

```
.SBTTL MACRO TESTAREA  
:***** TESTAREA *****  
: THIS MACRO SWITCHES TO THE SPECIFIED TEST MODE.  
: ARGUMENTS: 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
```

977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019

```
.SBTTL MACRO SET4 & RES4
***** SET4 & RES4 *****
: THESE MACROS SET & RESTORE VECTOR 4(TIMEOUT TRAP)
: IN IT'S RESTORED MODE TRAPS ARE REPORTED AS SUCH.
: ARGUMENTS: LOC ;THE LOCATION TO VECTOR TO (ONLY USED IN "SET4" NOT "RES4")
: ! 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$$ .LIST ME
.ENABL CRF
.LIST
MOV #TIMEOUT,4
CMP #1,PROTYP ;IS THIS AN 11/44?
BNE 101$ ;BRANCH IF NOT
CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
;THAT A EXPECTED TRAP COULD HAVE SET
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM RES4

101$:
```

1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043

```
.SBTTL MACRO DLEFT  
***** DLEFT *****  
: THIS MACRO DOES A DOUBLE WORD LEFT SHIFT  
: ARGUMENTS: 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
```

```

1046
1047
1048 000000 000000 000000
1049 000177
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069 000046 000046
1070 000046 014430
1071 000052 000052
1072 000052 000020
1073
1074 000024 000024
1075 000024 000200
1076 000042 000042
1077 000042 002000
1078 000044 000044
1079 000044 062566
1080 000200 000200
1081 000200 000437
1082 000202 000442
1086 000300 000300
1087 000300 005037 002566
1088 000304 000137 003630
1089 000310
1090 000316 000137 003630
1095 002000
    
```

```

.SBTTL TRAP CATCHER
.=0
.WORD 0,0
.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

.=46
$ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
.=52
.WORD BIT4 ;:2)SET LOC.52 TO INDICATE MEMORY SIZE DEPENDANT
.SBTTL APT11 HOOKS
.=24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
200 ;:FOR APT START UP
.=42
STACK ;SO RT11 CAN START WITH RUN COMMAND
.=44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR ;:POINT TO APT HEADER BLOCK
.=200
START3: BR START1 ;'NORMAL' START
BR START2 ;RESTART (SAVE ERROR ACCOUNTING)
.=300
START1: CLR RESTART
JMP START
START2: SET RESTART
JMP START
.=STACK
    
```

1098			.SBTTL VARIABLES	INITIALIZED TO ZERO
1099			:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS	
1100			:*USED IN THE PROGRAM.	
1101	002000		\$CMTAG:	::START OF COMMON TAGS
1102	002000	000000	SELONLY:0	:SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
1103	002002	000000	DIAGFLAG:0	:SET FOR SHIFTING DIAGONAL TEST
1104	002004	000000	KAMIKAZE:0	:SET FOR KAMIKAZE MODE TESTING
1105	002006	000000	SKIPKAMI:0	:USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
1106			:NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER	
1107	002010	000	\$PATMAR:.BYTE 0	:PATTERN NUMBER
1108	002011	000	\$BANK: .BYTE 0	:BANK & SIGN
1109	002012	000	\$ERFLG: .BYTE 0	::CONTAINS ERROR FLAG
1110	002013	000	\$ITEMB: .BYTE 0	::CONTAINS ITEM CONTROL BYTE
1111	002014	000000	LASTERROR:.WORD 0	:NUMBER OF ERRORS ON LAST PASS
1112	002016	000000	ERRPC: .WORD 0	:CONTAINS PC OF ERROR FOR TYPEOUT
1113	002020	000000	BADPC: .WORD 0	:CONTAINS PC OF ERROR
1114	002022	000000	ERRSP: .WORD 0	:CONTAINS SP OF ERROR FOR TYPEOUT
1115	002024	000000	BADSP: .WORD 0	:CONTAINS SP OF ERROR
1116	002026	000000	ERRPSW: .WORD 0	:CONTAINS PSW OF ERROR FOR TYPEOUT
1117	002030	000000	BADPSW: .WORD 0	:CONTAINS PSW OF ERROR
1118	002032	000000	ADDRESS:.WORD 0	::CONTAINS ADDRESS OF 'BAD' DATA
1119	002034	000000	PADDRESS:.WORD 0	:ADDRESS OF PARITY ERROR
1120	002036	000000	PHYADD: .WORD 0,0	:22 BIT PHYSICAL ADDRESS
1121	002042	000000	GOOD: .WORD 0	::CONTAINS 'GOOD' DATA
1122	002044	000000	GOOD2: .WORD 0	::CONTAINS 'GOOD2' DATA
1123	002046	000000	GOOD3: .WORD 0	::CONTAINS 'GOOD3' DATA
1124	002050	000000	BAD: .WORD 0	::CONTAINS 'BAD' DATA
1125	002052	000000	BAD2: .WORD 0	::CONTAINS 'BAD2' DATA
1126	002054	000000	BAD3: .WORD 0	::CONTAINS 'BAD3' DATA
1127	002056	000000	BAD XOR: .WORD 0	:XOR OF GOOD & BAD = BAD BITS
1128	002060	000000	\$AUTO: .WORD 0	:AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
1129	002062	000000	FATAL\$: .WORD 0	:FATAL ERROR INDICATOR
1130	002064	000000	SKPERR: .WORD 0	:USED TO SKIP ERROR MESSAGE IN '\$ERRGEN'
1131	002066	000000	NEMCNT: 0	:NON-EXISTANT MEMORY COUNTER (HOLES)
1132	002070	000000	PARCNT: 0	:PARITY ERROR COUNTER
1133	002072	000000	PATERR: 0	:PATTERN ERROR COUNTER
1134	002074	000000	NOPAR: 0	:NO PARITY ERROR MODE INDICATOR
1135	002076	000000	NONEM: 0	:NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
1136	002100	000000	BANK: 0	:MEMORY BANK UNDER TEST
1137	002102	000000	BANKINDEX:0	:USED TO INDEX INTO CONFIG TABLE
1138	002104	000000	CPUBIT: 0	:CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
1139	002106	000000	MUT: 0	:MEMORY UNDER TEST FLAG
1140	002110	000000	PATTERN:0	:PATTERN NUMBER UNDER TEST
1141	002112	000000	KPFLAG: .WORD 0	:BANK IS PROTECTED REGION OF ECC
1142	002114	000000	ACFLAG: .WORD 0	:BANK CAN BE ACCESSED BY THIS CPU
1143	002116	000000	MKFLAG: .WORD 0	:IF SET INDICATES MS11-M OR MF11S-K UNDER TEST
1144	002120	000000	PFLAG: .WORD 0	:BANK IS IN PROGRAM SPACE
1145	002122	000000	RRFLAG: .WORD 0	:BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
1146	002124	000000	RLFLAG: .WORD 0	:PROGRAM IS RELOCATED FLAG
1147	002126	000000	BMFLAG: .WORD 0	: 'BANK IS IDENTIFIED AS BAD MEMORY' FLAG
1148	002130	000000	EUFLAG: .WORD 0	: 'BANK HAS EUB MEMORY' FLAG
1149	002132	000000	TMFLAG: .WORD 0	: 'TYPE OF MEMORY TO TEST' FLAG; 0 = PARITY, 1 = ECC
1150	002134	000000	INTFLAG: .WORD 0	: 'BANK IS INTERLEAVED' FLAG
1151	002136	000000	INT64K: .WORD 0	: 'BANK IS 64K INTERLEAVED' FLAG
1152	002140	000000	ABORTFLAG: .WORD 0	: 'ABORT OCCURED' FLAG
1153	002142	000000	CTLKVEC: .WORD 0	: HOLDS OLD KERNAL STACK POINTER IN CASE OF CNTL/K
1154	002144	000000	CSR: .WORD 0	: DATA TO OR FROM CSR

C00000

```

1155 002146 000000 CSRNO: 0 ;CSR ADDRESS NUMBER (4 LSB'S)
1156 002150 000000 OLDCSR: .WORD 0 ;OLD CSR NUMBER(USED IN INH PTR TEST)
1157 ;THESE LOCATIONS STORE GPR'S DURING SUPERVISOR TESTS
1158 002152 000000 SUPDRO: 0
1159 002154 000000 SUPDR1: 0
1160 002156 000000 SUPDR2: 0
1161 002160 000000 SUPDR3: 0
1162 002162 000000 SUPDR4: 0
1163 002164 000000 SUPDR5: 0
1164 002166 000000 SUPDR6: 0
1165 002170 000000 DJMMY: 0 ;DUMMY LOCATION FOR ADDRESS PASSING
1166 ;THESE LOCATIONS STORE GPR'S & PSW DURING DETAILED ERROR PRINTOUTS
1167 002172 000000 DETRO: 0
1168 002174 000000 DETR1: 0
1169 002176 000000 DETR2: 0
1170 002200 000000 DETR3: 0
1171 002202 000000 DETR4: 0
1172 002204 000000 DETR5: 0
1173 002206 000000 DETSP: 0
1174 002210 000000 DETPSW: 0
1175 002212 000000 DETFLAG: 0 ;DETAILED REPORT FLAG
1176 002214 000000 CONTFLAG: 0 ;CSR'S HAVE BEEN TESTED FLAG
1177 002216 000000 TOTCSRS: .WORD 0 ;1 BIT PER EXISTING CSR, EG-
1178 ;CSR 0 REPRESENTED BY BIT 15, ETC.
1179 002220 000000 CSRFIRST: .WORD 0 ;FIRST ADDRESS UNDER CONTROL OF THIS CSR
1180 002222 000000 CSRLAST: .WORD 0
1181 002224 000000 CSRFBANK: .WORD 0
1182 002226 000000 CSRLBANK: .WORD 0
1183 002230 000000 CSRINT: .WORD 0
1184 002232 000000 SPLTCSR: .WORD 0
1185 002234 000000 000000 DATBUF: .WORD 0,0 ;TWO WORD DATA BUFFER
1186 002240 000000 000000 TSTDAT: .WORD 0,0 ;TWO WORD TEST DATA
1187 002244 000000 000000 SBEMSK: .WORD 0,0 ;TWO WORD SINGLE BIT ERROR MASK
1188 002250 000000 000000 DBEMSK: .WORD 0,0 ;TWO WORD DOUBLE BIT ERROR MASK
1189 002254 000000 SUPDOADD: .WORD 0 ;ADDRESS OF SUBROUTINE TO EXECUTE IN SUPERVISOR MODE
1190 002256 000 PASFLG: .BYTE 0 ;LOCAL LOOP PASS CONTROL
1191 002257 000 UPPFLG: .BYTE 0 ;LOCAL LOOP PASS CONTROL
1192 002260 000000 REALPAT: .WORD 0 ;RFAL PATTERN UNDER TEST
1193 002262 000000 OLDCACHE: .WORD 0 ;BACKED UP VALUE OF CACHE CONTROL REGISTER
1194 002264 000000 PARTHERE: .WORD 0 ;PARITY TRAPS SOMETIMES GO TO ADDRESS STORED HERE
1195 002266 000000 FSSTACK: .WORD 0 ;STACK SAVED HERE IF IN FIELD SERVICE MODE
1196 002270 000000 NEWBANK: .WORD 0 ;USED FOR RELOCATION TO A NEW BANK
1197 002272 000000 SOURCE: .WORD 0 ;SOURCE OF DATA WORDS FOR CHECKBIT GENERATION SUBROUTINE
1198 002274 000000 CHECK: .WORD 0 ;CHECKBITS TO BE LOADED INTO CSR
1199 002276 000000 PCBUMP: .WORD 0 ;VALUE TO BUMP THE PC BY TO RECOVER AFTER A PARITY TRAP
1200 002300 000000 CSRINC: .WORD 0 ;VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR
1201 002302 000000 CSRLOOP: .WORD 0 ;LOOP CONTROL FOR CSR TESTING
1202 002304 000000 SUCCESS: .WORD 0 ;FLAG SET BY SUCCESSFULL TASK OR SUBROUTINE
1203 002306 000000 ZEROS: .WORD 0 ;FOR AID IN 'MOV' INSTRUCTIONS
1204 002310 000000 TIME: .WORD 0 ;SECONDS THAT BATTERIES SHOULD LAST
1205 002312 000000 SKIPMK: .WORD 0 ;FLAG TO SKIP MKCONTROL SUBROUTINE
1206 002314 000000 NULLFLAG: .WORD 0 ;SET WHEN RUNNING NULL PATTERNS
1207 002316 000000 QVFLAG: 0 ;FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
1208 002320 000000 ACTFLAG: 0 ;FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
1209 002322 000000 APTFLAG: 0 ;FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
1210 002324 000000 XXDPCHAIN: 0 ;FLAGS XXDP CHAIN MODE PROGRAMMING RULES
1211 ;NOTE: THESE TWO BYTES MUST STAY TOGETHER

```



1212	002326	000			\$NULL: .BYTE 0	::CONTAINS NULL CHARACTER FOR FILLS
1213	002327	000			\$FILLS: .BYTE 0	::CONTAINS # OF FILL CHARACTERS
1214	002330	000			\$TPFLG: .BYTE 0	::'TERMINAL NOT AVAILABLE' FLAG
1215					.EVEN	
1216	002332	000000			\$ESCAPE:0	::ESCAPE ON ERROR ADDRESS
1217	002334	000000			EVEN: 0	::USED FOR ALTERNATE DATA PATTERNS
1218	002336	000000			STRIPES:0	::COUNTS DIAGONAL STRIPES
1219	002340	000000			COUNT: 0	::BACKED UP COPY OF STRIPES
1220	002342	000000			NOTAB: 0	::NO TABLE BEING PRINTED - NOW
1221	002344	000000			Bsize: 0	::SIZE OF 11/45 MOS MEMORY IN K WORDS
1222	002346	000000			Ksize: 0	::SIZE OF MF11S-K MEMORY IN K WORDS
1223	002350	000000			Lsize: 0	::SIZE OF MS11-L MEMORY IN K WORDS
1224	002352	000000			Msize: 0	::SIZE OF MS11-M MEMORY IN K WORDS
1225	002354	000000			Psize: 0	::SIZE OF UNIBUS PARITY MEMORY IN K WORDS
1226	002356	000000			TOOMANY:0	::FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
1227	002360	000000			READONLY:0	::FLAG TO PATTERNS TO READ ONLY
1228	002362	000000	000000		TESTADD:0,0	::THE ADDRESS TO RUN CSR TESTS ON
1229	002366	000000			UNITOP: 0	::HIGHEST ACCESSABLE BANK OF MEMORY THRU UNIBUS MAP
1230	002370	000000			STOPOK: 0	::FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
1231	002372	000000			APTPAR: .WORD 0	::AMOUNT OF PARITY MEMORY ACCORDING TO APT
1232	002374	000000			APTECC: .WORD 0	::AMOUNT OF ECC MEMORY ACCORDING TO APT
1233	002376	000000			NOFSMODE:0	::FLAG TO DISABLE FIELD SERVICE MODE
1234	002400	000000			NOERROR:0	::'THIS IS NOT AN ERROR' FLAG
1235	002402	000000			LOADBANK:0	::BANK LOADERS ARE RELOCATED TO
1236	002404	000000			TEMP: 0	::USED FOR JUNK
1237	002406	000000			QUICK: 0	::QUICK STOP FLAG FOR APT POWER FAIL
1238	002410	000000			NOSCOPE:0	::'NO SCOPE LOOP ALLOWED' FLAG
1239	002412	000000			FSINFLAG:0	::'FIELD SERVICE - NO INTERNAL INTERLEAVE' FLAG
1240	002414	000000			APTSIZE:0	::APT SIZING INFO FLAG
1241	002416	000000			FS7FLAG:0	::TRUE WHEN IN FIELD SERVICE COMMAND 7
1246	002420	000000			CONFGERRROR:0	::CONFIGURATION ERROR FLAG
1247	002422	000000			I: 0	::USED FOR GENERAL PURPOSE INDEXING
1248	002424	000000			NO22BIT:0	::NO 22-BIT MODE FLAG
1249	002426	000000			NOSUPER:0	::NO SUPERVISOR MODE FLAG
1250	002430	000000			ERRADD: .WORD 0	::HOLDS THE CSR'S ERROR ADDRESS
1251	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	000000		
1252	002452	000000	000000	000000	0,0,0,0,0,0,0,0	::POSSIBLE CSR'S
	002460	000000	000000	000000		
	002466	000000	000000			
1253	002472	000000			LINK1: 0	::USED TO HOLD LINKS TO PATTERNS WHICH
1254	002474	000000			LINK2: 0	::CAN EXECUTE IN THE PAR/PDR'S OR NOT
1255	002476	000000			CSRHOLD:0	::USED TO STORE CSR VALUES FOR CSR TESTS
1256	002500	000000			KFLAG: 0	::USED TO FLAG MF11S-K MEMORY TO TESTS
1257	002502	000000	000000		PGMCSR: .WORD 0,0	::POINTS TO PROGRAM CSR
1258	002506	000000			INH ECC: .WORD 0	::FLAGS INHIBIT ECC TESTS ON RELOCATION
1259	002510	000000			INHBANK: .WORD 0	::
1260	002512	000000			FULLREL: .WORD 0	::
1298	002514				\$CMTGE: ;*END OF COMMON TAGS	

Line	Address	Value	Variable	Description
1301			.SBTTL	VARIABLES INITIALIZED TO NON ZERO
1302	002514	000001	CACHKN: 1	:CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
1303	002520	001415	CACHKF: 1415	:CACHE CONSTANT (MOVED TO CONTRL TO TURN OFF CACHE)
1304	002522	040000	TESTMODE: 40000	:USED TO SELECT THE PROPER TEST MODE FOR A PATTERN RUN
1305	002524	000012	ERRMAX: 10	:MAX # OF ERRORS PER BANK WITH SW11
1306	002526	000167	LASTBANK: 167	:HIGHEST BANK OF MEMORY
1307	002530	170000	LASTBLOCK: 170000	:HIGHEST BANK OF MEMORY+1 (IN PAR FORMAT)
1308	002532	000031	SOBK: 25	:SOB CONSTANT
1309	002534	002000	KSTACK: STACK	:STACK BEGINNING
1310	002536	000001	LOADHOME: 1	:HOME BANK OF LOADERS
1311	002540	177777	WORST: 177777	:SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
1312	002542	176543	SEEDHI: 176543	:WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
1313	002544	123456	SEEDLO: 123456	:WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
1314	002546	176543	MSEEDH: 176543	:MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
1315	002550	123456	MSEEDL: 123456	:MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
1316	002552	177777	HEADER: 177777	:USED TO PRINT HEADINGS ONLY ONCE
1317	002554	177777	ONES: 177777	:FOR AID IN 'MOV' INSTRUCTIONS
1318	002556	000003	FLIPLOC: 3	:COUNTER FOR FLIPING DATA ON WORST CASE NOISE TEST
1319	002560	052525	SOFTPAT: 52525	:PATTERN FOR SOFT ERROR BACKGROUND TESTS
1320	002562	000000	\$LPADR: .WORD 0	::CONTAINS SCOPE LOOP ADDRESS
1321	002564	000000	\$LPERR: .WORD 0	::CONTAINS SCOPE RETURN FOR ERRORS
1322	002566	000000	RESTART: 0	:RESTART (START ADD 202) FLAG
1323	002570	000000	\$ERTTL: .WORD 0	::CONTAINS TOTAL ERRORS
1327				
1328				:***** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER *****
1329	002572	000377	BAKPAT: .WORD 377	:BACKGROUND PATTERN *
1330	002574	177400	SWAPAT: .WORD 177400	:SWAPPED BAKPAT *
1331				:*****
1332				
1333	002576	177570	SWR: .WORD DSWR	::ADDRESS OF SWITCH REGISTER
1334	002600	177570	DISPLAY: .WORD DDISP	::ADDRESS OF DISPLAY REGISTER
1335	002602	177560	\$TKS: 177560	::TTY KBD STATUS
1336	002604	177562	\$TKB: 177562	::TTY KBD BUFFER
1337	002606	177564	\$TPS: 177564	::TTY PRINTER STATUS REG. ADDRESS
1338	002610	177566	\$TPB: 177566	::TTY PRINTER BUFFER REG. ADDRESS
1339	002612	012	\$FILLC: .BYTE 12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
1340	002613	207	\$BELL: .ASCIZ <207><377><377>	::CODE FOR BELL
	002616	000		
1341	002617	077	\$QUES: .ASCII /?/	::QUESTION MARK
1342	002620	015	\$CRLF: .ASCII <15>	::CARRIAGE RETURN
1343	002621	012	\$LF: .ASCIZ <12>	::LINE FEED
1344			.EVEN	

1347  
 1348  
 1349  
 1350  
 1351  
 1352  
 1353  
 1354  
 1355  
 1356  
 1357  
 1358  
 1359  
 1360  
 1361  
 1362  
 1363  
 1364  
 1365  
 1366  
 1367  
 1368  
 1369 002624 000201  
 1372 003630

```

.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 SUCCESSFULLY 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
:CONFIG: .REPT 201
:CONFIEND:
  
```

1374  
 1375 003630

```

.SBTTL ***** MAIN *****
START: SUBTST <<INITIALIZE VARIABLES TO ZERO>>
:*****
:*SUBTEST INITIALIZE VARIABLES TO ZERO
:*****
RESET
MOV KSTACK,SP ;;SETUP THE STACK POINTER
MOV #SCMTAG,RO ;;FIRST LOCATION TO BE CLEARED
1$: CLR (RO)+ ;;CLEAR MEMORY LOCATION
CMP #SCMTGE,RO ;;DONE?
BNE 1$ ;LOOP BACK IF NO
MOV #167, LASTBANK ;RESTORE LASTBANK (THIS MUST BE DONE PRIOR TO SYSTEM SIZING)
SUBTST <<CLEAR NON-PROGRAM SPACE>>

```

1379 003630 000005  
 1380 003632 013706 002534  
 1386 003636 012700 002000  
 1387 003642 005020  
 1388 003644 022700 002514  
 1389 003650 001374  
 1390 003652 012737 000167 002526  
 1391 003660

```

:*****
:*SUBTEST CLEAR NON-PROGRAM SPACE
:*****

```

1392  
 1393  
 1394  
 1395 003660 012737 000001 002074  
 1396 003666 005000  
 1397 003670 000241  
 1398 003672 005520  
 1399 003674 020027 160000  
 1400 003700 103773  
 1401 003702 005037 002074  
 1402

```

;THIS ATTEMPS TO GET RID OF ANY PARITY ERRORS BY WRITING INTO
;EVERY LOCATION THAT IS NOT LOADED INTO BY THE PROGRAM OR ALLOCATED
;TO THE XXDP LOADERS
MOV #1,NOPAR ;PARITY ACTION = COUNT & IGNORE
CLR RO
2$: CLC
ADC (RO)+
CMP RO,#160000
BLO 2$
CLR NOPAR ;RESTORE DEFAULT PARITY ACTION

```

1411 003706

SUBTST <<TYPE OF SYSTEM SIZER>>

```
*****
:SUBTEST      TYPE OF SYSTEM SIZER
*****
1412 003706 000401          BR      SYSSIZ          ;SKIP OVER VARIABLE LOCATION
1413 003710 000000          PROTYP: .WORD 0
1414 003712          SYSSIZ: SET4 #4$
1415 003720 005737 177746  TST     CONTRL          ;SEE IF CACHE REGISTER RESPONDS
1416 003724          SET4 #9$          ;YES - DO WE HAVE 11/44 TYPE CACHE
1417 003732 005737 177750  TST     MAINT           ;OR 11/60 TYPE CACHE?
1418 003736 000411          BR      5$             ;BRANCH IF 11/44 TYPE CACHE
1419 003740 012737 000014 002520 9$:  MOV     #14,CACHKF      ;TURN OFF CONSTANT FOR 11/60 CACHE
1420 003746 000405          BR      5$
1421 003750 005037 002514 4$:  CLR     CACHKN          ;NO CACHE ON SYSTEM
1422 003754 012737 002306 064002 5$:  MOV     #ZEROS,DT14     ;DO NOT PRINT CONTRL ERROR MESSAGES
1423 003762          SET4 #6$
1424 003770 005737 172516  TST     MMR3           ;DO WE HAVE AN MMR3?
1425 003774 005037 172516  CLR     MMR3           ;YES WE DO
1426 004000 052737 000020 172516  BIS     #BIT4,MMR3      ;SEE IF THERE IS 22-BIT MODE
1427 004006 032737 000020 172516  BIT     #BIT4,MMR3
1428 004014 001024          BNE    10$            ;BRANCH IF 22-BIT RELOCATION
1429 004016 000411          BR      7$             ;BRANCH IF MMR3 BUT NO 22-BIT RELOC.
1430
1431 004020 012737 140000 002522 6$:  MOV     #140000,TESTMODE ;MAKE TEST MODE USER
1432 004026 005237 002426          INC     NOSUPER        ;NO SUPERVISOR MODE
1433 004032 005037 063652          CLR     DT5+10
1434 004036 005037 064012          CLR     DT14+10
1435
1436 004042 005237 002424 7$:  INC     NO22BIT         ;NO 22 BIT MODE
1437 004046 012737 000007 002526  MOV     #7,LASTBANK     ;124K MEMORY MAX. MEMORY SIZE
1438 004054 005037 063654          CLR     DT5+12         ;DO NOT TRY TO PRINT ERROR REGISTER
1439 004060 005037 064014          CLR     DT14+12        ;ERROR MESSAGES, BECAUSE THERE IS
1440
1441 004064 000417          BR      8$             ;IS NO ERROR REGISTER!
1442 004066 10$:  SET4 #8$
1443 004074 000007          MFPT
1444
1445
1446
1447
1448 004076 110037 003710          MOVB   R0,PROTYP        ;TYPE OF PROCESSOR TEST: THIS INSTRUCTION
1449 004102 022737 060003 003710  CMP     #3,PROTYP        ;(AVAILABLE ON NEWER PROCESSORS ONLY) PLACES
1450 004110 001005          BNE    8$             ;A CODE IN THE LOWER BYTE OF R0 THAT
1451 004112 065237 002426          INC     NOSUPER        ;INDICATES THE PROCESSOR TYPE. 1=11/44
1452 004116 012737 140000 002522  MOV     #140000,TESTMODE ;3=11/24
1453
1454 004124 8$:  RES4          ;MOV THE CODE TO PROTYP
                          ;IS THIS AN 11/24?
                          ;BRANCH IF NOT - WE HAVE AN 11/44
                          ;NO SUPERVISOR MODE
                          ;MAKE TEST MODE USER
```

1457 004146

SUBTST <<INITIALIZE VARIABLES TO NON ZERO>>

\*\*\*\*\*  
: \*SUBTEST INITIALIZE VARIABLES TO NON ZERO  
\*\*\*\*\*

1458 004146  
1459 004154 012737 000003 002556  
1460 004162  
1461 004170 012737 176543 002546  
1462 004176 012737 123456 002550  
1463 004204 013737 002546 002542  
1464 004212 013737 002550 002544  
1465 004220 012737 000377 002572  
1466 004226 012737 177400 002574  
1471 004234

SET WORST  
MOV #3,FLIPLOC  
SET HEADER  
MOV #176543,MSEEDH  
MOV #123456,MSEEDL  
MOV MSEEDH,SEEDHI ;PRIME THE RANDOM NUMBER GENERATOR  
MOV MSEEDL,SEEDLO ;BOTH HIGH AND LOW WORDS  
MOV #377,BAKPAT  
MOV #177400,SWAPAT  
SUBTST <<INITIALIZE VECTORS>>

\*\*\*\*\*  
: \*SUBTEST INITIALIZE VECTORS  
\*\*\*\*\*

1472 004234 012737 055334 000020  
1473 004242 012737 000340 000022  
1474 004250 012737 055642 000030  
1475 004256 012737 000340 000032  
1476 004264 012737 062602 000034  
1477 004272 012737 000340 000036  
1478 004300 012737 051524 000024  
1479 004306 012737 000340 000026  
1480 004314 012737 077720 000114  
1481 004322 012737 000340 000116  
1482 004330 012737 040114 000010  
1483 004336 012737 000340 000012  
1484 004344 012737 040070 000004  
1485 004352 012737 000340 000006  
1486 004360 012737 040102 000250  
1487 004366 012737 000340 000252  
1492 004374 104423

MOV #SCOPE,IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE  
MOV #340,IOTVEC+2 ;;LEVEL 7  
MOV #ERROR,EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE  
MOV #340,EMTVEC+2 ;;LEVEL 7  
MOV #STRAP,TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS  
MOV #340,TRAPVEC+2;LEVEL 7  
MOV #SPWRDN,PWRVEC ;;POWER FAILURE VECTOR  
MOV #340,PWRVEC+2 ;;LEVEL 7  
MOV #PARITY,PARVEC;GET READY FOR PARITY ERRORS  
MOV #340,PARVEC+2  
MOV #PDP1105,RESVEC;RESERVED INSTRUCTION TRAP  
MOV #340,RESVEC+2  
MOV #TIMEOUT,ERRVEC;SETUP TIMEOUT ERRORS  
MOV #340,ERRVEC+2 ;SET PRIORITY OF ERROR TRAPS  
MOV #MMTRAP,MMVEC ;VECTOR FOR MEMORY MANAGEMENT  
MOV #340,MMVEC+2  
CACHON ;TURN CACHE ON

1495 004376

SUBTST <<INITIALIZE PATTERNS>>

\*\*\*\*\*  
: \*SUBTEST INITIALIZE PATTERNS  
\*\*\*\*\*

1496  
1497  
1498  
1499  
1500 004376 012700 062552  
1501 004402 012001  
1502 004404 012703 017366  
1503 004410 012702 000020  
1504 004414 004737 004514  
1505 004420 012001  
1506 004422 012702 000010  
1507 004426 004737 004514  
1508 004432 012001  
1509 004434 012703 017616  
1510 004440 012702 000020  
1511 004444 004737 004514  
1512 004450 012001  
1513 004452 012702 000010  
1514 004456 004737 004514  
1515 004462 012001  
1516 004464 012703 020002  
1517 004470 012702 000020  
1518 004474 004737 004514  
1519 004500 012001  
1520 004502 012702 000010  
1521 004506 004737 004514  
1522 004512 000417  
1523  
1524 004514

;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.

MOV #SDDW0,R0  
MOV (R0)+,R1  
MOV #MKCSRT,R3  
MOV #16.,R2  
CALL PATPLUG  
MOV (R0)+,R1  
MOV #8.,R2  
CALL PATPLUG  
MOV (R0)+,R1  
MOV #MKPAT,R3  
MOV #16.,R2  
CALL PATPLUG  
MOV (R0)+,R1  
MOV #8.,R2  
CALL PATPLUG  
MOV (R0)+,R1  
MOV #MJPAT,R3  
MOV #16.,R2  
CALL PATPLUG  
MOV (R0)+,R1  
MOV #8.,R2  
CALL PATPLUG  
BR SUBAAA

PATPLUG:SUBTST <<SUBR PLUG IN NULL PATTERNS>>

\*\*\*\*\*  
: \*SUBTEST SUBR PLUG IN NULL PATTERNS  
\*\*\*\*\*

1525 004514  
1526 004522 006001  
1527 004524  
1528 004526 012713 026354  
1529 004532  
1530 004532 062703 000002  
1531 004536  
1532 004550 000207

FOR I := #1 TO R2  
ROR R1  
ON.NOERROR ;IF CARRY CLEAR  
MOV #MT0999,(R3)  
END ;OF ON.ERROR  
ADD #2,R3  
END ;OF FOR  
RETURN

1535 004552

1536

1537

1538

1539 004552

1540 004560 012700 002624

1541 004564 005020

1542 004566 022700 003630

1543 004572 001374

1544 004574

1545

1546 004574 012737 000002 002104

1547 004602

1548

1549

1550

1551 004602

1552 004610 012737 177570 002576

1553 004616 012737 177570 002600

1554 004624

1555 004634 000403

1556 004636 012716 004644

1557 004642 000002

1558 004644

1559 004666 012737 000176 002576

1560 004674 012737 000174 002600

1561 004702

1562

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 DESCRIBED AT LOCATION 'CONFIG'.

.ENABLE LSB  
IF RESTART IS FALSE

1\$: MOV #CONFIG,R0  
CLR (R0)+  
CMP #CONFIEND,R0  
BNE 1\$

END ;OF IF RESTART

.DSABL LSB

MOV #BIT1,CPUBIT ;SET ID BIT

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

SET4 #3\$ ;TRAPS TO 4 GOTO 3\$

MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWITCH REGISTER

MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER

IF #-1 EQ @SWR ;IF NO TRAP FROM REFERENCE TO @SWR AND @SWR = #-1

BR 2\$ ;:BRANCH IF NO TIMEOUT

3\$: MOV #2\$, (SP) ;:SET UP FOR TRAP RETURN

RTI

2\$: RES4 ;:RESET TRAPS TO 4 TO DEFAULT

MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR

MOV #DISPREG,DISPLAY

END ;OF IF #-1

.DSABL LSB



1565 004702

```
SUBAAB: SUBTST <<SETUP ACT, APT, & XXDP>>
;*****
;*SUBTEST     SETUP ACT, APT, & XXDP
;*****
;THIS SETS UP A BUNCH OF FLAGS TO TELL THE PROGRAM EVERYTHING
;IT CARES TO KNOW ABOUT APT, ACT, & XXDP.
CLR $PASS ;CLEAR PASS COUNT
IFB #BITS SET.IN $ENVM
SET $TPFLG ;INDICATE NO TERMINAL
END ;OF IFB #BITS
IFB #BIT7 SET.IN $ENVM
SET APTSIZE
END ;OF IFB #BIT7
IFB $ENV EQ #1
SET APTFLAG,QVFLAG,$AUTO,QUICK
MOV #APTDOWN,PWPVEC
MOV #SSWREG,SWR ;USE APT SWR
ELSE
IF 42 NE #STACK AND 42 NE #0
SET QVFLAG,$AUTO
IF 42 EQ #SENDAD
SET ACTFLAG
ELSE
SET XXDPCHAIN
END ;OF IF 42
END ;OF IF 42
END ;OF IFB $ENV
```

1566  
1567  
1568 004702 005037 062474  
1569 004706  
1570 004716  
1571 004724  
1572 004724  
1573 004734  
1574 004742  
1575 004742  
1576 004752  
1577 005002 012737 045210 000024  
1578 005010 012737 062510 002576  
1579 005016  
1580 005020  
1581 005036  
1582 005052  
1583 005062  
1584 005070  
1585 005072  
1586 005100  
1587 005100  
1588 005100

1590 005100

SUBTST <<PROTECT PROGRAM & LOADERS>>

```

*****
*SUBTEST      PROTECT PROGRAM & LOADERS
*****
  
```

1591 005100 052737 000200 002624  
 1592 005106 052737 000200 002630  
 1593 005114  
 1594 005124  
 1595 005130  
 1596  
 1597 005130

```

BIS      #BIT7,CONFIG      ;PROTECT PROGRAM SPACE (BANK 0)
BIS      #BIT7,CONFIG+4    ;PROTECT LOADER SPACE (BANK 1)
IF #SENDAD NE 42          ;NOT ACT-11?
  TYPE   MSG000           ;TYPE PROGRAM TITLE
END ;OF IF #SENDAD
  
```

SUBTST <<CHECK SYSTEM FOR CACHE>>

```

*****
*SUBTEST      CHECK SYSTEM FOR CACHE
*****
  
```

1598  
 1599  
 1600  
 1601 005130  
 1602 005136 005737 177746  
 1603 005142  
 1604 005150 005737 177750  
 1605 005154  
 1606 005162 005737 177754  
 1607 005166  
 1608 005172 000405  
 1609 005174  
 1610 005200 000402  
 1611 005202  
 1612 005206 052737 000014 177746  
 1613 005214 042737 000014 177746  
 1614 005222 032737 000004 177746  
 1615 005230 001004  
 1616 005232 032737 000010 177746  
 1617 005240 001413  
 1618 005242  
 1619 005246 104424  
 1620 005250 013737 002514 002516  
 1621 005256 005037 002514  
 1622 005262 000404  
 1623 005264  
 1624 005270  
 1625

```

;* THIS FIGURES OUT IF THERE IS A CACHE ON THE SYSTEM,
;* WHAT TYPE OF SYSTEM IT IS, AND WHETHER IT IS ENABLED
;* OR DISABLED.
SET4     #3$
TST      CONTRL           ;IS THERE A CONTROL REGISTER?
SET4     #2$
TST      MAINT            ;IS THERE A MAINTENANCE REGISTER?
SET4     #1$
TST      DATARG           ;IS THERE A DATA REGISTER?
TYPE    MSG117           ; 11/44
BR       4$
1$:      TYPE    MSG116   ; 11/34
BR       4$
2$:      TYPE    MSG118   ; 11/60
4$:      BIS     #BIT2!BIT3,CONTRL ;SET CACHE DISABLE BITS
        BIC     #BIT2!BIT3,CONTRL ;CLEAR CACHE DISABLE BITS
        BIT     #BIT2,CONTRL      ;IS THE BIT SET?
        BNE     7$              ;BRANCH IF THE BIT IS SET
        BIT     #BIT3,CONTRL      ;IS THE BIT SET?
        BEQ     6$              ;BRANCH IF THE BIT IS SET
7$:      TYPE    MSG121   ; CACHE BYPASSED
        CACHOFF
        MOV     CACHKN,CACHKN+2  ;SAVE INFO ABOUT CACHE
        CLR     CACHKN          ;CACHE CANNOT BE USED - IT'S BYPASSED
BR       8$
3$:      TYPE    MSG119   ; NO
6$:      TYPE    MSG120   ; CACHE AVAILABLE
  
```

1672 005274

```
      SUBTST <<SETUP USER & SUPERVISOR STACK>>  
;*****  
;*SUBTEST   SETUP USER & SUPERVISOR STACK  
;*****
```

1673 005274 104421  
1674 005276 005737 002426  
1675 005302 001011

```
8$:  DEENERGIZE      ;TURN OFF MEMORY MANAGEMENT  
     TST NOSUPER    ;IS THERE A SUPERVISOR MODE?  
     BNE 5$         ;NO-SKIP SUPERVISOR SETUP.
```

1676  
1677  
1678 005304 042737 030000 177776  
1679 005312 052737 010000 177776

```
     ;SET PREVIOUS MODE TO SUPERVISOR  
     BIC #BIT13!BIT12,PSW  
     BIS #BIT12,PSW
```

1680  
1681 005320  
1682 005324 006606

```
     PUSH #SUPSTK  
     MTP1 SSP
```

1683  
1684  
1685 005326 052737 030000 177776 5\$:

```
     ;SET PREVIOUS MODE TO USER  
     BIS #BIT13!BIT12,PSW
```

1686  
1687 005334  
1688 005340 006606

```
     PUSH #USESTK  
     MTP1 USP
```

1689  
1690 005342

```
      SUBTST <<GET SOFTWARE SWITCH REGISTER IF NECESSARY>>  
;*****  
;*SUBTEST   GET SOFTWARE SWITCH REGISTER IF NECESSARY  
;*****
```

1694 005342  
1695 005350  
1696 005360 104407

```
     IF $AUTO IS FALSE      ;IF NOT(APT OR ACT)  
     IF SWR EQ #SWREG      ;IF SOFTWARE SWITCH REG SELECTED  
     GTSWR                  ;:GET SOFT-SWR SETTINGS  
     END ;OF IF SWR  
     END ;OF IF $AUTO
```

1697 005362  
1698 005362  
1702  
1703 005362

```
      SUBTST <<GET MEMORY MANAGEMENT READY>>  
;*****  
;*SUBTEST   GET MEMORY MANAGEMENT READY  
;*****
```

1707 005362 104422  
1711 005364  
1712 005400 104420

```
     KMAP                ;MAP KERNEL SPACE 1 TO 1  
     MAP                 ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1  
     ENERGIZE           ;TURN ON MEMORY MANAGEMENT
```

1715 005402

005402 0C0004

1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738 005404 005005  
1739 005406 005000  
1740 005410 012703 172100  
1741 005414 012737 000001 002074  
1742 005422  
1743 005430 005713  
1744 005432 052705 000001  
1745 005436 005004  
1746 005440 052760 000007 002432  
1747 005446 052760 000030 002432  
1748 005454 004537 006010  
1749 005460 100001  
1750 005462 012713 040000  
1751 005466 032713 040000  
1752 005472 001403  
1753 005474 042760 000001 002432  
1754 005502 005013  
1755 005504 012713 020000  
1756 005510 032713 020000  
1757 005514 001417  
1758 005516 042760 000002 002432  
1759 005524 012713 020000  
1760 005530 004537 006010  
1761 005534 000010  
1762 005536 012713 020000  
1763 005542 004537 006010  
1764 005546 000022  
1765 005550 012713 020000

```

NEWST <<BIT TEST OF ALL CSR'S>>
*****
*TEST 1 BIT TEST OF ALL CSR'S
*****
TST1: SCOPE
* THE FIRST PART OF THE CONFIGURATION ANALYSIS DOES THE FOLLOWING:
* 1) FINDS WHICH CSR'S RESPOND, AND PUTS THEM INTO THE CSR INFORMATION
* TABLE, AND STORES ANOTHER BIT FOR 'TOTCSRS'.
* 2) TESTS THE CSR BITS COMMON TO ALL CSR'S.
* 3) FIGURES OUT IF THERE IS AN EUB BIT, AN ECC BIT, AND THE EXISTANCE
* OF THE ERROR ADDRESS BITS, AND MARKS THIS IN THE CSR
* INFORMATION TABLE.
* 4) TESTS THE BITS PARTICULAR TO THAT TYPE OF CSR.
* 5) IF ANY BITS TEST BAD IN THE CSR UNDER TEST, THE CSR OK BIT IN THE
* CSR INFORMATION TABLE IS CLEARED.
* THE INFORMATION BITS ONE THROUGH THREE FORM A CODE WHICH GIVES THE TYPE
* OF CSR:
*
* TYPE ERR. ADDR. PARITY NOT EUB CODE TOTALS
* UNIBUS PARITY BIT2 BIT1 BIT0
* MS11-L 1 1 0 7
* MF11S-K 1 0 1 6
* MS11-M 1 0 0 5
* 11/45 BIPOLAR 0 1 1 4
*
* THIS MEMORY CODE WILL BE USED IN THE SECOND PART OF THIS ANALYSIS
*
LLR R5 ;R5 IS THE TOTAL CSR NUMBER
CLR R0 ;R0 IS A TABLE INDEX
MOV #CSRADD,R3 ;R3 HAS THE CSR ADDRESS
MOV #1,NOPAR ;IGNORE PARITY ERRORS
SET4 #2$
1$: TST (R3) ;DOES THE CSR RESPOND?
BIS #1,R5
CLR R4 ;CLEAR THE LAST CSR INDICATOR
BIS #7,CSRINFO(R0) ;SET ALL THE MEMORY INFO BITS
BIS #BIT4!BIT3,CSRINFO(R0) ;YES-MARK IT IN CSR INFORMATION TABLE
JSR R5,TEST ;TEST BIT 0 AND 15
.WORD BIT15!BIT0
MOV #BIT14,(R3) ;IS THERE A BIT 14 RESPONDING
BIT #BIT14,(R3) ;(IT'S THE EUB BIT)
BEQ 3$ ;BRANCH IF NO EUB BIT
BIC #BIT0,CSRINFO(R0) ;CLEAR EUB INFO IN THE CSR TABLE
3$: CLR (R3) ;CLEAR THE CSR UNDER TEST
MOV #BIT13,(R3) ;DOES BIT 13 RESPOND
BIT #BIT13,(R3) ;(TO TEST FOR ECC CSR)
BEQ 4$ ;BRANCH IF NOT ECC CSR
BIC #BIT1,CSRINFO(R0) ;CLEAR PARITY INFO IN THE CSR TABLE
MOV #BIT13,(R3) ;SET THE INHIBIT MODE POINTER TO 1ST 16K
JSR R5,TEST ;TEST BIT 3
.WORD BIT3
MOV #BIT13,(R3)
JSR R5,TEST ;TEST BIT 1 AND 4
.WORD BIT4!BIT1
MOV #BIT13,(R3)

```

```

1767 005554 004537 006010 4$: JSR R5,TEST ;TEST BIT 2
1768 005560 000004 .WORD BIT2
1769 005562 005013 CLR (R3)
1770 005564 052713 007740 BIS #7740,(R3) ;ARE THERE ERROR ADDRESS BITS?
1771 005570 032713 007740 BIT #7740,(R3)
1772 005574 001404 BEQ 5$ ;BRANCH IF NO ERROR ADDR. BITS.
1773 005576 004537 006010 JSR R5,TEST ;TEST BITS 5->11
1774 005602 007740 .WORD 7740
1775 005604 000403 BR 6$ ;SKIP OVER THE INFORMATION REPORTING
1776 005606 042760 000004 002432 5$: BIC #BIT2,CSRINFO(R0) ;REPORT THAT THERE ARE NO ERROR ADDRESS BITS.
1777 005614 032760 000002 002432 6$: BIT #BIT1,CSRINFO(R0) ;IS THIS CSR AN ECC CSR?
1778 005622 001014 BNE 7$ ;BRANCH IF NOT
1779 005624 032760 000001 002432 BIT #BIT0,CSRINFO(R0) ;IS THE EUB BIT SET?
1780 005632 001410 BEQ 7$ ;BRANCH IF IT IS
1781 .WE MUST NOW TEST FOR MS11-M ON THE UNIBUS
1782 005634 012713 007760 MOV #7760,(R3) ;PUT PATTERN & SBE BIT INTO BITS 4->11
1783 005640 022713 007760 CMP #7760,(R3) ;ARE THEY STILL THERE?
1784 005644 001403 BEQ 7$ ;YES - BRANCH FOR MF11S-K CSR
1785 005646 042760 000001 002432 BIC #BIT0,CSRINFO(R0) ;NO - SET EUB BIT FOR MS11-M
1786 005654 005013 7$: CLR (R3) ;CLEAR CSR
1787 005656 022760 000040 002432 CMP #40,CSRINFO(R0) ;IS THIS A LEGAL CONFIGURATION?
1788 005664 100004 BPL 10$ ;BRANCH IF IT'S LEGAL
1789 005666 016037 002432 002050 MOV CSRINFO(R0),BAD
1790 005674 104021 ERROR +21 ;ILLEGAL TYPE ERROR
1791 005676 032760 000004 002432 10$: BIT #BIT2,CSRINFO(R0) ;DOES THIS CSR HAVE ERROR BITS
1792 005704 001016 BNE 2$ ;BRANCH IF TRUE
1793 005706 032760 000002 002432 BIT #BIT1,CSRINFO(R0) ;ARE THE OTHER 2 BITS SET?
1794 005714 001404 BEQ 11$ ;BRANCH IF NOT
1795 005716 032760 000001 002432 BIT #BIT0,CSRINFO(R0)
1796 005724 001006 BNE 2$
1797 005726 016037 002432 002050 11$: MOV CSRINFO(R0),BAD
1798 005734 104021 ERROR +21 ;ILLEGAL TYPE ERROR
1799 005736 005060 002432 CLR CSRINFO(R0) ;CLEAR THE CSR INFO-IT WILL NOT EXIST IN THE PROGRAM
1800 005742 062700 000002 2$: ADD #2,R0 ;INCREMENT TO NEXT CSR TABLE
1801 005746 062703 000002 ADD #2,R3 ;INCREMENT TO NEXT CSR
1802 005752 006305 ASL R5
1803 005754 103001 BCC 8$ ;IS THERE A CSR 0?
1804 005756 005204 INC R4 ;YES - SET CSR PRESENT FLAG
1805 005760 022700 000040 8$: CMP #40,R0 ;ARE WE DONE?
1806 005764 001221 BNE 1$ ;BRANCH IF MORE TO DO
1807 005766 000241 CLC
1808 005770 006005 ROR R5 ;RESYNC R5
1809 005772 005704 TST R4 ;WAS THERE A CSR 0?
1810 005774 001402 BEQ 9$ ;BRANCH IF NOT
1811 005776 052705 100000 BIS #BIT15,R5 ;YES - SET IN THE CSR TABLE
1812 006002 010537 002216 9$: MOV R5,TOTCSRS ;STORE R5 IN TOTCSRS
1813 006006 000437 BR CTEST ;JUMP OVER SUBROUTINE
1814 ;THIS SUBROUTINE TESTS THE CSR BITS
1815 006010 012501 TEST: MOV (R5)+,R1 ;GET THE BIT TO TEST
1816 006012 050113 BIS R1,(R3) ;SET THAT IN THE CSR UNDER TEST
1817 006014 030113 BIT R1,(R3) ;IS THE BIT STILL THERE?
1818 006016 001013 BNE 1$ ;BRANCH IF STILL THERE
1819 006020 011337 002144 MOV (R3),CSR ;READ CSR
1820 006024 010137 002042 MOV R1,GOOD
1821 006030 032713 100020 BIT #BIT15!BIT4,(R3) ;IS IT BECAUSE OF A SBE OR DBE?
1822 006034 001004 BNE 1$ ;BRANCH IF IT IS
1823 006036 104035 ERROR +35 ;BIT SET ERROR

```

CZMSDB0 MS11-L/M DIAGNOSTIC  
T1 BIT TEST OF ALL CSR'S

MACRO M1113 07-OCT-80 18:01 PAGE 75-1 SEQUENCE 134

SEQ 0141

```

1824 006040 042760 000010 002432      BIC    #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
1825 006046 040113          1$:    BIC    R1,(R3)          ;CLEAR THE SELECTED BIT
1826 006050 030113          BIT    R1,(R3)          ;IS IT CLEARED?
1827 006052 001413          BEQ    2$              ;BRANCH IF IT IS CLEARED
1828 006054 011337 002144      MOV    (R3),CSR       ;READ CSR
1829 006060 010137 002042      MOV    R1,GOOD
1830 006064 032713 100020      BIT    #BIT15:BIT4,(R3);IS IT BECAUSE OF A SBE OR DBE?
1831 006070 001004          BNE    2$              ;BRANCH IF TRUE
1832 006072 104010          ERROR  +10           ;BIT CLEAR ERROR
1833 006074 042760 000010 002432      BIC    #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
1834 006102 000205          2$:    RTS    R5
1835 006104 0C0000          TRACE: .WORD 0

```

```

1837 ;THE FOLLOWING ROUTINE DETERMINES WHICH CSR CONTROLS PROGRAM SPACE
1838
1839 006106 104424 ;TEST: CACHOFF
1840 006110 012737 177777 002502 MOV #177777,PGMCSR
1841 006116 012737 002000 172350 MOV #2000,KIPAR4 ;SET UP MAP REGISTER
1842 006124 012701 002362 MOV #TESTADD,R1
1843 006130 012737 100000 002362 MOV #100000,TESTADD
1844 006136 012737 100002 002364 MOV #100002,TESTADD+2
1845 006144 005000 CLR R0 ;CLEAR CSR COUNTER
1846 006146 005037 002146 CLR CSRNO
1847 006152 013703 002216 MOV TOTCSRS,R3 ;OBTAIN CSR MAP
1848 006156 000240 NOP ;DEBUG AID
1849 006160 006303 4$: ASL R3 ;PUT HIGH ORDER BIT INTO C BIT
1850 006162 103407 BCS 2$ ;BRANCH IF CSR EXISTS
1851 006164 062700 000002 1$: ADD #2,R0 ;UPDATE CSR COUNTER
1852 006170 010037 002146 MOV R0,CSRNO
1853 006174 005703 TST R3 ;IS MAP EMPTY?
1854 006176 001464 BEQ 3$ ;BRANCH IF SO
1855 006200 000767 BR 4$
1856 006202 000240 2$: NOP ;DEBUG AID
1857 006204 000241 CLC ;CLEAR CARRY
1858 006206 032760 000002 002432 BIT #BIT1,CSRINFO(R0) ;IS THIS PARITY MEMORY?
1859 006214 001414 BEQ 5$ ;BRACH IF NOT
1860 006216 052760 000004 172100 BIS #BIT2,CSRADD(R0) ;SET WRITE WRONG PARITY
1861 006224 012771 123456 000000 MOV #123456,@(R1) ;WRITE DATA
1862 006232 012771 123456 000002 MOV #123456,@2(R1)
1863 006240 005060 172100 CLR CSRADD(R0) ;RESTORE CSR
1864 006244 000414 BR 6$
1865 006246 012760 000000 172100 5$: MOV #0,CSRADD(R0) ;CLEAR THE CSR UNDER TEST
1866 006254 012771 123456 000000 MOV #123456,@(R1) ;WRITE DATA
1867 006262 012771 123456 000002 MOV #123456,@2(R1)
1868 006270 012760 020006 172100 MOV #20006,CSRADD(R0) ;SET DIAG CHECK MODE
1869 006276 005771 000000 6$: TST @(R1) ;WRITE CHECKBITS TO CSR
1870 006302 016004 172100 MOV CSRADD(R0),R4 ;WRITE CSR TO R4
1871 006306 032760 000002 002432 BIT #BIT1,CSRINFO(R0) ;PARITY MEMORY?
1872 006314 001403 BEQ 7$ ;BRACH IF NOT
1873 006316 005704 TST R4 ;PARITY ERROR?
1874 006320 100411 BMI 8$ ;BRACH IF SO
1875 006322 000720 BR 1$ ;TRY NEXT CSR
1876 006324 000240 7$: NOP ;DEBUG AID
1877 006326 072427 177773 ASH #-5,R4
1878 006332 042704 177600 BIC #^C177,R4
1879 006336 022704 000157 CMP #157,R4 ;CORRECT CHECKBITS?
1880 006342 001310 BNE 1$ ;BRANCH IF NOT
1881 006344 010037 002502 8$: MOV R0,PGMCSR
1882 006350 000240 3$: NOP ;DEBUG AID
1883 006352 104502 CLRCSR ;CLEAR ALL CSR'S
1884 006354 012771 000000 000000 MOV #0,@(R1) ;RESTORE TEST LOCATIONS
1885 006362 012771 000000 000002 MOV #0,@2(R1)
1886 006370 023727 002502 177777 CMP PGMCSR,#177777
1887 006376 001402 BEQ FINT ;IF PROGRAM CSR NOT FOUND GO TO FINT
1888 006400 000137 007004 JMP CLRMEM ;GO TO SIZING ROUTINE IF FOUND

```

```

1890
1891      ; IF PGMCSR WAS NOT FOUND BY THE PRECEEDING ROUTINE, THIS ROUTINE TRIES
1892      ; TO FIND IT FOR INTERLEAVED MEMORIES
1893
1894 006404      FINT:  SET4      #2$      ;NE MEMORY TRAPS GO TO 2$
1895 006412      012771 123456 000000 1$:  MOV      #123456,@(R1)  ;WRITE DATA AT FIRST LOCATION OF BANK 2 IN BOARD
1896 006420      012771 123456 000002      MOV      #123456,@2(R1)  ;WRITE DATA AT SECOND LOCATION OF BANK 2 IN BOARD
1897 006426      062737 010000 172350      ADD      #10000,KIPAR4  ;UPDATE PAR4 TO POINT TO UPPER BOARDS
1898 006434      000766      BR        1$          ;KEEP GOING TILL NO MORE MEMORY
1899 006436      012700 177776      2$:  MOV      #-2,R0
1900 006442      013703 002216      MOV      TOTCSRS,R3  ;PUT CSR MAP IN R3
1901 006446      062700 000002      3$:  ADD      #2,R0      ;UPDATE CSR COUNTER
1902 006452      010037 002146      MOV      R0,CSRNO    ;UPDATE CSRNO
1903 006456      006303      ASL      R3
1904 006460      103403      BCS      4$          ;BRANCH IF CSR EXISTS
1905 006462      005703      TST      R3          ;ANY CSR'S LEFT?
1906 006464      001405      BEQ      5$          ;BRANCH IF NOT
1907 006466      000767      BR        3$          ;LOOK FOR NEXT CSR
1908 006470      012760 020006 172100 4$:  MOV      #20006.CSRADD(R0) ;SET DIAGNOSTIC CHECK MODE IN CSR
1909 006476      000763      BR        3$          ;LOOK FOR NEXT CSR
1910 006500      5$:  SET4      #6$          ;NE MEMORY TRAPS NOW GO TO 6$
1911 006506      012700 177776      MOV      #-2,R0      ;RESET CSR POINTER
1912 006512      012737 002000 172350      MOV      #2000,KIPAR4  ;REMAP PAR4 TO POINT TO BANK 2
1913 006520      005771 000000      TST      @(R1)        ;TEST NONASSERTED LOCATIONS
1914 006524      062700 000002      6$:  ADD      #2,R0      ;UPDTAE CSR POINTER
1915 006530      010037 002146      MOV      R0,CSRNO
1916 006534      022700 000040      CMP      #40,R0      ;NOT FOUND?
1917 006540      001515      BEQ      10$         ;BRANCH IF NOT
1918 006542      016004 172100      MOV      CSRADD(R0),R4 ;GET CSR CONTENTS
1919 006546      072427 177773      ASH      #-5,R4
1920 006552      042704 177600      BIC      #^C177,R4    ;CLEAR ALL BUT CHECKBITS
1921 006556      022704 000157      CMP      #157,R4     ;PROPER CHECKBITS?
1922 006562      001401      BEQ      7$          ;BRANCH IF SO
1923 006564      000757      BR        6$          ;TRY NEXT CSR IF NOT
1924 006566      110037 002502      7$:  MOVVB   R0,PGMCSR    ;WRITE NON-ASSERTED CSR # IN PGMCSR
1925 006572      SET4      #8$          ;NE TRAPS GO TO 8$
1926 006600      012700 177776      MOV      #-2,R0
1927 006604      013703 002216      MOV      TOTCSRS,R3  ;PUT CSR MAP IN R3
1928 006610      062700 000002      23$:  ADD      #2,R0      ;UPDATE CSR COUNTER
1929 006614      010037 002146      MOV      R0,CSRNO    ;UPDATE CSRNO
1930 006620      006303      ASL      R3
1931 006622      103403      BCS      24$         ;BRANCH IF CSR EXISTS
1932 006624      005703      TST      R3          ;ANY CSR'S LEFT?
1933 006626      001405      BEQ      25$         ;BRANCH IF NOT
1934 006630      000767      BR        23$        ;LOOK FOR NEXT CSR
1935 006632      012760 020006 172100 24$:  MOV      #20006.CSRADD(R0) ;SET DIAGNOSTIC CHECK MODE IN CSR
1936 006640      000763      BR        23$        ;LOOK FOR NEXT CSR
1937 006642      012700 177776      25$:  MOV      #-2,R0
1938 006646      005771 000002      TST      @2(R1)      ;TEST ASSERTED LOCATIONS
1939 006652      062700 000002      8$:  ADD      #2,R0
1940 006656      010037 002146      MOV      R0,CSRNO
1941 006662      022700 000040      CMP      #40,R0
1942 006666      001442      BEQ      10$
1943 006670      016004 172100      MOV      CSRADD(R0),R4
1944 006674      072427 177773      ASH      #-5,R4
1945 006700      042704 177600      BIC      #^C177,R4
1946 006704      022704 000157      CMP      #157,R4    ;PROPER CHECKBITS?

```



1947	006710	001401				BEQ	9\$		:BRANCH IF SO
1948	006712	000757				BR	8\$		:TRY NEXT CSR IF NOT
1949	006714	110037	002503		9\$:	MCVB	R0,PGMCSR+1		:WRITE ASSERTED CSR # IN PGMCSR
1950	006720	052737	100000	002502		BIS	#BIT15,PGMCSR		:SET INTERLEAVED INDICATOR IN PGMCSR
1951	006726	104502				CLRCR			
1952	006730	012737	002000	172350		MOV	#2000,KIPAR4		
1953	006736					SET4	#12\$		:NE MEMORY TRAPS GO TO 12\$
1954	006744	012771	000000	000000	11\$:	MOV	#0,@(R1)		:WRITE DATA AT FIRST LOCATION OF BANK 2 IN BOARD
1955	006752	012771	000000	000002		MOV	#0,@2(R1)		:WRITE DATA AT SECOND LOCATION OF BANK 2 IN BOARD
1956	006760	062737	010000	172350		ADD	#10000,KIPAR4		:UPDATE PAR4 TO POINT TO UPPER BOARDS
1957	006766	000766				BR	11\$		
1958	006770	104423			12\$:	CACHON			
1959	006772	000404				BR	CLRMEM		
1960	006774				10\$:	TYPE	MSG'26		:ERROR - PROGRAM CSR NOT FOUND.
1961	007000	005037	002502			CLR	PGMCSR		:SET TO DEFAULT OF 0

1963 007004

SUBTST <<CLEAR ALL MEMORY SPACE FROM BANK 2 ON>>

\*\*\*\*\*  
:SUBTEST CLEAR ALL MEMORY SPACE FROM BANK 2 ON  
\*\*\*\*\*

1964  
1965  
1966  
1967  
1968  
1969

:THIS ROUTINE CLEARS ALL MEMORY SPACE BEGINNING AT ADDRESS 200,000 AND  
:CONTINUES UNTIL THERE IS NO MEMORY LEFT. IT SHOULD CLEAR ANY PARITY ERRORS  
:CREATED BY THE LAST ROUTINE, AND CLEAN UP ANY JUNK LEFT HANGING AROUND IN  
:HIGHER MEMORY.

1970 007004  
1971 007012 005037 006104  
1972 007016 012737 000001 002074  
1973 007024 012737 002000 172350  
1974 007032 012701 100000  
1975 007036 020127 177776  
1976 007042 001003  
1977 007044 012737 177777 006104  
1978 007052 005021  
1979 007054 005737 006104  
1980 007060 001001  
1981 007062 000765  
1982 007064 062737 001000 172350  
1983 007072 005037 006104  
1984 007076 012701 100000  
1985 007102 000755  
1986 007104 000240  
1987 007106 005037 006104  
1988 007112

```
CLRMEM: SET4      #CLREX      ;NONEM TRAPS GO TO CLREX
          CLR      TRACE
          MOV      #1,NOPAR    ;IGNORE PARITY ERRORS
          MOV      #2000,KIPAR4 ;SET UP MAP TO START AT BANK 2
          MOV      #100000,R1  ;R1 MAPS TO KIPAR4
1$:      CMP      R1,#177776  ;WHOLE 16K BANK DONE?
          BNE      2$         ;KEEP GOING IF NOT
          MOV      #-1,TRACE   ;USE TRACE FLAG TO FLAG END OF BANK
2$:      CLR      (R1)+       ;CLEAR CONTENTS & INCREMENT
          TST      TRACE      ;EOB FLAG SET?
          BNE      3$         ;GO TO NEXT BANK IF SO
          BR       1$
3$:      ADD      #1000,KIPAR4 ;SET MAP FOR NEXT BANK
          CLR      TRACE      ;RESET FLAG
          MOV      #100000,R1  ;RESET R1
          BR       1$         ;CLEAR NEXT BANK
CLREX:  NOP
          CLR      TRACE
          RES4
```

1991 007134

1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022 007134  
2023 007142  
2024 007146  
2025 007152  
2026 007156  
2027 007160  
2028 007162  
2029 007166  
2030 007170  
2031 007174  
2032 007176  
2033 007204  
2034 007206  
2035 007212  
2036 007214  
2037 007220  
2038 007224  
2039 007230  
2040 007236  
2041 007244  
2042 007246  
2043 007250  
2044 007254

005037 002274  
012701 002362  
013703 002216  
005000  
005005  
005737 002424  
001403  
005037 002530  
000413  
022737 000167 002526 7\$:  
001407  
013702 002526  
005202  
072227 000011  
010237 002530  
012702 000004  
012737 001000 172350  
012737 001000 172352  
006303  
103420  
000002  
010037 002146

```

ANA2:  SUBST <<MATCH ALL CSR'S WITH MEMORY>>
*****
*SUBTEST  MATCH ALL CSR'S WITH MEMORY
*****
* THE SECOND PART OF THE ANALYSIS MATCHES UP THE CSR'S WITH THE MEMORY, AND
* INSTALLS ALL THE INFORMATION FOUND IN THE CONFIGURATION TABLE.  FOR ECC,
* THIS IS DONE BY TAKING EACH CSR FOUND IN THE PREVIOUS SECTION SEQUENTIALLY
* AND CHECKING THROUGH ALL OF MEMORY, ONE BANK AT A TIME, TO SEE WHICH BANKS
* RESPOND TO THE CSR IN QUESTION.  THE FIRST DOUBLE WORD PAIR IN EACH BANK ARE
* WRITTEN WITH DATA AND DIAGNOSTIC CHECK MODE SET IN THE CSR IN ORDER TO AC-
* COMPLISH THIS.  ALL POSSIBLE CONFIGURATIONS OF DOUBLE WORD PAIRS (NON-INTER-
* LEAVED, 64K INTERLEAVED, OR 128K INTERLEAVED) ARE CHECKED FOR EACH BANK
* THROUGH USE OF TESTADD AND KERNEL INSTRUCTION PAGE ADDRESS REGISTERS 4 AND
* 5.  IF WE GET THE PROPER CHECKBITS BACK, WE HAVE A MATCH.  IF NOT, THE ROUT-
* INE CHECKS FOR SINGLE OR DOUBLE BIT ERRORS.
*   IF ONE OR THE OTHER IS FOUND, THE ERROR ADDRESS IS CHECKED
* TO SEE IF IT IS THAT BANK.  IF IT IS, WE HAVE A MATCH.  AT THE END OF EACH
* BANK PASS, FOR EACH CSR PASS, THE PROGRAM COMES UP WITH A NUMBER, STORED IN
* 'I', WHICH DENOTES THE FOLLOWING:
*
*   I      MEMORY DESCRIPTION
*   -      -----
*   0      NON-EXISTANT MEMORY
*   1      NON-INTERLEAVED MEMORY
*   2      64K INTERLEAVED, A1 NOT ASSERTED MEMORY
*   3      128K INTERLEAVED, A1 NOT ASSERTED MEMORY
*   4      64K INTERLEAVED, A1 ASSERTED MEMORY
*   5      128K INTERLEAVED, A1 ASSERTED MEMORY
*
* NOTE - I=2 THROUGH I=5 CAN ONLY OCCUR WITH MS11-M MEMORY.
*
* NOTE THAT PARITY MEMORY WRITES WRONG PARITY TO THE DOUBLE WORDS, THEN LOOKS
* FOR THE PARITY ERROR BIT TO BE SET.  IF THE BIT IS SET, WE HAVE A MATCH.
*
SFT4  #6$      ;NE MEMORY TRAPS GO TO 4
CLR   CHECK    ;CLEAR CHECK
MOV   #TESTADD,R1 ;SET UP THE VIRTJAL ADDR. POINTER
MOV   TOTCSRS,R3 ;MOVE CSR MAP INTO R3
CLR   R0       ;CLEAR THE CSR POINTER
CLP   R5       ;CLEAR THE PROGRAM CSR STATUS POINTER
TST   NO22BIT  ;IS THIS AN 11/44 OR 11/24?
BEQ   7$      ;BRANCH IF IT IS
CLR   LASTBLOCK ;ADJUST LASTBLOCK INDICATOR FOR 124K MACHINE
BR    1$      ;BRANCH OVER NEXT PIECE OF CODE
CMP   #167, LASTBANK ;IS THERE UNIBUS MEMORY ABOVE 17000000?
BEQ   1$      ;BRANCH IF NOT
MOV   LASTBANK,R2 ;SET UP A NEW LAST BLOCK INDICATOR
INC   R2
ASH   #9.,R2
MOV   R2, LASTBLOCK
MOV   #4,R2      ;R2 IS INDEX FOR CONFIG TABLE
MOV   #1000,KIPAR4 ;SET KIPAR4 FOR BANK 1
MOV   #1000,KIPAR5 ;SET KIPAR5 FOR BANK 1
ASL   R3        ;DOES THIS CSR EXIST?
BCS   3$        ;BRANCH IF IT DOES EXIST
ADD   #2,R0     ;INCREMENT THE CSR POINTER
MOV   R0,CSRNO ;STORE IT IN CSRNO ALSO

```

```

2045 007260 005703          TST      R3          ;ARE THERE ANY MORE CSR'S TO DO?
2046 007262 001370          BNE      2$          ;BRANCH IF ALL CSRS NOT DONE
2047 007264 012737 001000 172350  MOV      #1000,KIPAR4 ;RESTORE KIPAR4
2048 007272 012737 001200 172352  MOV      #1200,KIPAR5 ;RESTORE KIPAR5
2049 007300 013706 002534          MOV      KSTACK,SP  ;RESTORE STACK
2050 007304 000137 010474          JMP      SUBAAS      ;JUMP TO SUBAAS IF ALL CSR'S ARE DONE
2051 007310 010037 002146          MOV      RO,CSRNO    ;MAKE SURE CSRNO IS UPDATED
2052 007314 104424          3$:      MOV      RO,CSRNO
13$:      CACHOFF          ;TURN THE CACHE OFF
2053 007316 000240          NOP
2054 007320 012737 100000 002362 45$:      MOV      #100000,TESTADD ;SET UP VIRTUAL ADDRESS TO KIPAR4
2055 007326 012737 120002 002364  MOV      #120002,TESTADD+2 ;SET UP VIRTUAL ADDRESS TO KIPAR5
2056 007334 032762 000040 002624  BIT      #BIT5,CONFIG(R2) ;IS THIS A BANK TO SKIP?
2057 007342 001402          BEQ      43$         ;NO - BRANCH AROUND NEXT INSTRUCTION
2058 007344 000137 010420          JMP      6$          ;YES - GO TO END OF BANK
2059 007350 005037 002422          43$:      CLR      I          ;CLEAR THE MEMORY CONFIGURATION COUNTER
2060 007354 005771 000000          4$:      TST      @ (R1)      ;TEST TO SEE THAT THERE IS MEMORY PRESENT
2061 007360 005237 002422          INC      I
2062 007364          PUSH     @ (R1),@2 (R1) ;SAVE THE LOCATIONS UNDER TEST
2063 007374 032760 000002 002432  BIT      #BIT1,CSRINFO(RO) ;IS THIS PARITY MEMORY?
2064 007402 001414          BEQ      34$         ;NO - BRANCH
2065 007404 052760 000004 172100  BIS      #BIT2,CSRADD(RO) ;SET WRITE WRONG PARITY
2066 007412 012771 123456 000000  MOV      #123456,@ (R1) ;SET THE FIRST LOCATION UNDER TEST
2067 007420 012771 123456 000002  MOV      #123456,@2 (R1) ;SET THE SECOND LUT
2068 007426 005060 172100          CLR      CSRADD(RO) ;CLEAR THE CSR
2069 007432 000411          BR       41$         ;TEST LOCATIONS
2070 007434 012771 123456 000000 34$:      MOV      #123456,@ (R1) ;SET THE FIRST LOCATION UNDER TEST
2071 007442 012771 123456 000002  MOV      #123456,@2 (R1) ;SET THE SECOND LUT
2072 007450 104503          CLR1CSR ;RESET CSR
2073 007452 104475          CB1CSR  ;SET DIAG. CHECK MODE IN CSR UNDER TEST
2074 007454 000240          NOP      ;DEBUG AID
2075 007456 005771 000000          41$:      TST      @ (R1)      ;READ THE FIRST LUT TO WRITE CKBITS. INTO CSR
2076 007462 104426          READCSR ;READ THE CSR UNDER TEST
2077 007464 000240          NOP      ;DEBUG AID
2078 007466 013704 002144          MOV      CSR,R4      ;GET THE CHECKBITS FROM THE CSR
2079 007472 000240          NOP      ;DEBUG AID
2080 007474 010437 002404          MOV      R4,TEMP     ;SAVE IN TEMP FOR LATER
2081 007500 104503          CLR1CSR ;RESET CSR
2082 007502          POP      @2 (R1),@ (R1) ;RESTORE LOCATIONS UNDER TEST
2083 007512 032760 000002 002432  BIT      #BIT1,CSRINFO(RO) ;IS THIS PARITY MEMORY?
2084 007520 001404          BEQ      42$         ;NO - BRANCH
2085 007522 005704          TST      R4          ;DID WE GET A PARITY ERROR?
2086 007524 100414          BMI      25$         ;YES - FILL IN CONFIG TABLE
2087 007526 000137 010420          JMP      6$          ;NO - JUMP TO END OF BANK
2088 007532 072427 177773          42$:      ASH      #-5,R4      ;MANIPULATE THE CSR BITS
2089 007536 042704 177600          BIC      #^C177,R4   ;INTO A USABLE FORM.
2090 007542 000240          NOP      ;DEBUG AID
2091 007544 022704 000157          CMP      #157,R4     ;DO THE CHECKBITS COMPARE TO WHAT WAS WRITTEN?
2092 007550 001402          BEQ      25$         ;BRANCH IF THERE IS A MATCH
2093 007552 000137 010100          JMP      22$         ;ELSE BRANCH IF NOT THE SAME
2094          ;*
2095          ;* WE COME HERE IF THERE IS A MATCH
2096          ;*
2097 007556 010004          25$:      MOV      RO,R4      ;GET THE CSR NUMBER
2098 007560 006204          ASR      R4          ;SET IT UP FOR USE IN THE
2099 007562 000304          SWAB     R4          ;CONFIGURATION TABLE.
2100 007564 042704 170377          BIC      #170377,R4  ;CLEAR OFF EXTRANEIOUS BITS
2101 007570 032737 000004 002422  BIT      #BIT2,I     ;INTERLEAVED AT ASSERTED MEMORY FOUND?

```

2102	007576	001402				BEQ	15\$		;BRANCH IF NOT
2103	007600	072427	000004			ASH	#4,R4		;PUT CSR NUMBER IN INTERLEAVED CSR SLOT
2104	007604	050462	002624		15\$:	BIS	R4,CONFIG(R2)		;PUT CSR NUMBER IN CONFIG. TABLE
2105	007610	016004	002432			MOV	CSRINFO(R0),R4		;GET MEMORY TYPE
2106	007614	042704	177770			BIC	#^C7,R4		;CLEAR OFF THE EXTRANEIOUS BITS
2107	007620	000304				SWAB	R4		;MOVE INTO PROPER POSITION
2108	007622	050462	002626			BIS	R4,CONFIG+2(R2)		;SET IT INTO THE CONFIG TABLE
2109	007626	022737	000001	002422		CMP	#1,!		;WAS THIS NON-INTERLEAVED MEMORY?
2110	007634	001431				BEQ	24\$		;BRANCH IF IT WAS
2111	007636	052762	010000	002626		BIS	#BIT12,CONFIG+2(R2)		;SET THE INTERLEAVED BIT
2112	007644	010204				MOV	R2,R4		;SAVE THE CURRENT BANK INDEX
2113	007646	032737	000001	002422		BIT	#BIT0,I		;WAS THIS 128K INTERLEAVED?
2114	007654	001006				BNE	5\$		;BRANCH IF TRUE
2115	007656	052762	004000	002626		BIS	#BIT11,CONFIG+2(R2)		;SET 64K INTERLEAVED FLAG IN CONFIG
2116	007664	062704	000020			ADD	#20,R4		;SET NEW BANK POINTER TO 4 BANKS AHEAD
2117	007670	000402				BR	16\$		;JUMP OVER NEXT INSTRUCTION
2118	007672	062704	000040		5\$:	ADD	#40,R4		;SET NEW BANK POINTER 8 BANKS AHEAD
2119	007676	052764	000040	002624	16\$:	BIS	#BIT5,CONFIG(R4)		;SET SKIP ECC LOGIC TESTS FLAG
2120	007704	056264	002624	002624		BIS	CONFIG(R2),CONFIG(R4)		;SET OTHER INFO INTO THAT BANK
2121	007712	056264	002626	002626		BIS	CONFIG+2(R2),CONFIG+2(R4)		
2122						*			
2123						*			;* THIS SECTION IS EXECUTED ONLY WHEN THE BANK=1
2124						*			
2125	007720	022737	001000	172350	24\$:	CMP	#1000,KIPAR4		;IS THIS BANK 1 ?
2126	007726	001402				BEQ	30\$		;BRANCH IF TRUE
2127	007730	000137	010260			JMP	33\$		;ELSE JUMP TO END OF THIS BANK
2128	007734	032737	100020	002404	30\$:	BIT	#BIT15!BIT4,TEMP		;WAS THERE A SBE OR DBE?
2129	007742	001417				BEQ	10\$		;BRANCH IF NOT
2130	007744	013704	002404			MOV	TEMP,R4		;GET CSR CONTENTS
2131	007750	072427	177767			ASH	#-9,R4		;MAKE ERROR ADDRESS INTO BANK #
2132	007754	022704	000001			CMP	#1,R4		;ERROR IN BANKS 0 OR 1?
2133	007760	003010				BGT	10\$		;BRANCH IF NOT
2134	007762	052762	000001	002624		BIS	#BIT0,CONFIG(R2)		;SET ERROR FLAG IN CONFIG TABLE
2135	007770	105262	002626			INCB	CONFIG+2(R2)		;ADD ONE TO BANK ERROR COUNT
2136	007774					SET	CONFERROR		;PRINT CONFIG TABLE
2137	010002	053737	002630	002624	10\$:	BIS	CONFIG+4,CONFIG		;SET UP INFORMATION IN BANK ZERO
2138	010010	053737	002632	002626		BIS	CONFIG+6,CONFIG+2		
2139	010016	000240				NOP			;DEBUG AID
2140	010020	022737	000001	002422		CMP	#1,I		;WAS THIS NON-INTERLEAVED MEMORY
2141	010026	001002				BNE	46\$		;NO - BRANCH OVER NEXT STMT.
2142	010030	000137	010420			JMP	6\$		;YES - JUMP TO END OF THIS BANK
2143	010034	012704	000020		46\$:	MOV	#20,R4		;SET UP COUNTER FOR 64K INTERLEAVED
2144	010040	032737	000001	002422		BIT	#BIT0,I		;WAS IT 128K INTERLEAVED?
2145	010046	001402				BEQ	26\$		;BRANCH IF NOT
2146	010050	062704	000020			ADD	#20,R4		;SET UP COUNTER FOR 128K INTERLEAVED
2147	010054	053764	002624	002624	26\$:	BIS	CONFIG,CONFIG(R4)		;SET OTHER BANK WITH SAME INFORMATION
2148	010062	053764	002626	002626		BIS	CONFIG+2,CONFIG+2(R4)		;AS IN BANK 0
2149	010070	052764	000040	002624		BIS	#BIT5,CONFIG(R4)		;SET SKIP ECC LOGIC TESTS FLAG
2150	010076	000470				BR	33\$		;BRANCH
2151						*			
2152						*			;* IF CHECKBITS DID NOT MATCH, WE COME HERE
2153						*			
2154	010100	032737	100020	002144	22\$:	BIT	#BIT15!BIT4,CSR		;SBE OR DBE FLAGS SET?
2155	010106	001001				BNE	8\$		;BRANCH IF TRUE
2156	010110	000463				BR	33\$		;CHECK TO SEE IF IT IS MS11-M
2157	010112	013704	002146		8\$:	MOV	CSRNO,R4		;GET CSRNO
2158	010116	042764	000006	172100		BIC	#6,CSRADD(R4)		;TURN OFF DIAG CHECK & ECC DISABLE

```

2159 010124          PUSH      R0,R1          ;SAVE R0 & R1
2160 010130 016401 172100  MOV      CSRADD(R4),R1    ;GET CSR INFORMATION
2161 010134 072127 177773  ASH      #-5,R1          ;SET UP ERROR ADDRESS
2162 010140 042701 177600  BIC      #^C177,R1
2163 010144 005737 002424  TST      NO22BIT          ;IS THIS AN 11/44 OR 11/24?
2164 010150 001015          BNE      27$             ;BRANCH IF NOT
2165 010152 052764 040000 172100  BIS      #BIT14,CSRADD(R4) ;GET EXTENDED ERROR ADDRESS BITS
2166 010160 016400 172100  MOV      CSRADD(R4),R0    ;READ FROM CSR
2167 010164 042764 040000 172100  BIC      #BIT14,CSRADD(R4) ;TURN OFF EUB BIT
2168 010172 042700 177037  BIC      #^C740,R0        ;SET UP EXTENDED BITS
2169 010176 006300          ASL      R0
2170 010200 006300          ASL      R0
2171 010202 060001          ADD      R0,R1            ;SET UP TOTAL ERROR ADDRESS
2172 010204 010104          MOV      R1,R4            ;SAVE IN R4
2173 010206          POP      R1,R0            ;RESTORE R0 & R1
2174 010212 072427 000005          ASH      #5,R4            ;SET ERROR ADDRESS UP IN PAR NOTATION
2175 010216 020437 172350          CMP      R4,KIPAR4        ;DOES IT EQUAL KIPAR4?
2176 010222 001001          BNE      28$             ;BRANCH IF FALSE
2177 010224 000403          BR       35$             ;YES - MARK INFO IN CONFIG TABLE
2178 010226 020437 172352          CMP      R4,KIPAR5        ;DOES IT EQUAL KIPAR5?
2179 010232 001012          BNE      33$             ;BRANCH IF FALSE
2180 010234 052762 000001 002624 35$:  BIS      #BIT0,CONFIG(R2) ;SET BANK ERROR FLAG
2181 010242 105262 002626          INCB    CONFIG+2(P2)     ;INCREMENT BANK ERROR COUNTER
2182 010246          SET     CONFIGERROR      ;PRINT CONFIG TABLE
2183 010254 000137 007556          JMP     25$             ;YES - MARK INFO IN CONFIG TABLE
2184          ;*
2185          ;* THIS SECTION SETS UP ALL THE POSSIBLE CONFIGURATIONS OF
2186          ;* MS11-M MEMORY.
2187          ;*
2188 010260 032760 000003 002432 33$:  BIT     #BIT0!BIT1,CSRINFO(R0) ;IS THIS MS11-M MEMORY?
2189 010266 001054          BNE     6$              ;NO - GO TO END OF BANK
2190 010270 032760 000004 002432          BIT     #BIT2,CSRINFO(R0)
2191 010276 001450          BEQ     6$
2192 010300 022737 000001 002422          CMP     #1,I            ;IS THIS 1ST TIME THROUGH?
2193 010306 103410          BLO     18$             ;BRANCH IF NOT
2194 010310 162737 000002 002364          SUB     #2,TESTADD+2     ;TRY AS 64K INTERLEAVED
2195 010316 062737 004000 172352          ADD     #4000,KIPAR5     ;A1 NON-ASSERTED MEMORY
2196 010324 000137 007354          JMP     4$              ;TRY TO MATCH AGAIN
2197 010330 022737 000004 002422 18$:  CMP     #4,I            ;4TH TIME THROUGH?
2198 010336 001404          BEQ     20$             ;YES - BRANCH
2199 010340 022737 000002 002422          CMP     #2,I            ;2ND TIME THROUGH
2200 010346 103405          BLO     12$             ;NO - BRANCH
2201 010350 062737 004000 172352 20$:  ADD     #4000,KIPAR5     ;TRY AS 128K INTERLEAVED
2202 010356 000137 007354          JMP     4$              ;TRY TO MATCH AGAIN
2203 010362 022737 000003 002422 12$:  CMP     #3,I            ;THIRD TIME THROUGH?
2204 010370 103413          BLO     6$              ;NO - BRANCH
2205 010372 062737 000002 002362          ADD     #2,TESTADD       ;TRY TESTING THE BANK
2206 010400 062737 000002 002364          ADD     #2,TESTADD+2     ;AS A1 ASSERTED
2207 010406 162737 004000 172352          SUB     #4000,KIPAR5     ;64K INTERLEAVED MEMORY
2208 010414 000137 007354          JMP     4$              ;TRY TO MATCH AGAIN
2209          ;*
2210          ;*END OF BANK ROUTINE
2211          ;*
2212 010420 104503          CLR1CSR                    ;CLEAR THE CSR UNDER TEST
2213 010422 062702 000004          ADD     #4,R2            ;UPDATE CONFIGURATION POINTER
2214 010426 062737 001000 172350          ADD     #1000,KIPAR4     ;UPDATE KIPAR4 TO NEXT BANK
2215 010434 013737 172350 172352          MOV     KIPAR4,KIPAR5    ;AND UPDATE KIPAR5

```

2216	010442	000240			NOP		:DEBUG AID
2217	010444	023737	002530	172350	CMP	LASTBLOCK,KIPAR4	:HAVE WE DONE THE WHOLE MEMORY SPACE?
2218	010452	002002			BGE	19\$	:BRANCH IF DONE
2219	010454	000137	007320		JMP	45\$	:JUMP IF NOT DONE
2220	010460	062700	000002	19\$:	ADD	#2,R0	:INCREMENT CSR POINTER
2221	010464	000240			NOP		:DEBUG AID
2222	010466	104423			CACHON		:TURN ON THE CACHE
2223	010470	000137	007224		JMP	1\$	:JUMP TO TRY NEXT CSR

2225 010474 104423  
2226 010476 104472  
2227 010500

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  
\*\*\*\*\*

010500 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  
CLR PARCNT ;CLEAR PARITY ERROR COUNTER  
MOV #1,NOPAR ;SET THE NO PARITY ERROR FLAG  
CLR NEMCNT ;CLEAR NON-EXISTANT MEMORY ERROR COUNTER  
MOV #1,NONEM ;SET THE NON-EXISTANT MEMORY ERROR MODE TO COUNT  
SET4 #NONEXIST ;TRAPS TO 4 GOTO NONEXIST  
CLR R0  
MOV #SIZE,R1  
CACHOFF ;TURN CACHE OFF  
1\$: TST (R0)+ ;SEE IF I CAN DO A READ ACCESS WITHOUT A PARITY TRAP  
SOB R1,1\$  
CACHON ;TURN CACHE ON  
;SEE IF ANY FAILURES  
TST PARCNT ;ANY PARITY ERRORS?  
BEQ 2\$ ;NO - SKIP  
FATAL 3  
2\$: TST NEMCNT ;ANY NON-EXISTANT MEMORY (HOLES)?  
BEQ 3\$ ;SKIP IF EQUAL  
SUB #2,ADDRESS ;UPDATE 1ST ADDRESS FAILURE FROM AUTO INCREMENT #  
FATAL 4  
3\$: BIS CPUBIT,CONFIG ;SET CORRECT ACCESSED BIT ON BANK 0  
RES4 ;RESET TRAPS TO 4 TO DEFAULT

2228  
2229  
2230  
2231  
2232  
2233  
2234 010502 005037 002070  
2235 010506 012737 000001 002074  
2236 010514 005037 002066  
2237 010520 012737 000001 002076  
2238 010526  
2239 010534 005000  
2240 010536 012701 040000  
2241 010542 104424  
2242 010544 005720  
2243 010546 077102  
2244 010550 104423  
2245  
2246 010552 005737 002070  
2247 010556 001403  
2248 010560  
2249 010566 005737 002066  
2250 010572 001406  
2251 010574 162737 000002 002032  
2252 010602  
2253 010610 053737 002104 002624  
2254 010616  
2255  
2256 010640

SUBTST <<ENABLE ECC FOR CORRECT TRAPS>>  
\*\*\*\*\*  
;\*SUBTEST ENABLE ECC FOR CORRECT TRAPS  
\*\*\*\*\*

2257 010640  
2258 010656 104506  
2259 010660  
2260 010662 104472  
2261 010664

IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE  
ENASBE ;TRAP ON SINGLE BIT ERRORS  
ELSE  
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)  
END ;OF IF #SW0



2264 010664

NEWSTST <<TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES>>

\*\*\*\*\*  
: \*TEST 3 TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES  
\*\*\*\*\*

010664 000004

TST3: SCOPE  
: EACH BANK IS TESTED FOR EXISTANCE AND IF IT EXISTS  
: THEN IS IS TESTED FOR ZEROS & ONES.  
: EXCEPT -  
: PROTECTED BANKS (WHERE THE PROGRAM IS) ARE ONLY TESTED BY  
: "TST" INSTRUCTIONS LIKE BANK #0  
: ANY BAD BANKS ARE LOGGED IN THE CONFIGURATION TABLE.  
: THIS ROUTINE IS ONLY DOING A SMART SIZE - NOT ACTUAL TESTING!  
CLR BANK  
MOV #1,NOPAR ;SET NO PARITY ERROR FLAG  
MOV #2,NONEM ;SET NON-EXISTANT MEMORY MODE TO EXIT TEST LOOP  
SET4 #NONEXIST ;TRAPS TO 4 GOTO NONEXIST  
CMP #1,PROTYP ;IS THIS AN 11/44?  
BEQ 1\$ ;BRANCH IF TRUE  
MOV #MTST3+4,LINK1 ;SET UP LINKS  
MOV #MTST3+6,LINK2  
BR TAG9\$  
1\$: BMDV MTST3 ;PUT IN FAST MEMORY  
MOV #UIPAR2,LINK1 ;SET UP LINKS  
MOV #UIPAR3,LINK2  
TAG9\$: INC BANK  
CMP LASTBANK,BANK ;DONE?  
BLO TAG2\$ ;YES - SKIP TO NEXT TEST  
MOV BANK,R1  
ASL R1  
ASL R1 ;BANK \* 4  
MOV R1,BANKINDEX  
CLR PATERR ;CLEAR PATTERN ERROR COUNTER  
CLR PARCNT ;CLEAR PARITY ERROR COUNTER  
CLR NEMCNT ;CLEAR NON-EXISTANT MEMORY COUNTER (HOLES)  
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK  
TSTB CONFIG(R1) ;IS THIS BANK PROTECTED?  
BMI TSTBANK ;YES - GO TEST BANK SPECIAL  
2272 010666 005037 002100  
2273 010672 012737 000001 002074  
2274 010700 012737 000002 002076  
2275 010706  
2276 010714 022737 000001 003710  
2277 010722 001407  
2278 010724 012737 011522 002472  
2279 010732 012737 011524 002474  
2280 010740 000411  
2281 010742  
2282 010750 012737 177644 002472  
2283 010756 012737 177646 002474  
2284 010764 005237 002100  
2285 010770 023737 002526 002100  
2286 010776 103457  
2287 011000 013701 002100  
2288 011004 006301  
2289 011006 006301  
2290 011010 010137 002102  
2291 011014 005037 002072  
2292 011020 005037 002070  
2293 011024 005037 002066  
2294 011030  
2295 011044 105761 002624  
2296 011050 100555  
2297 011052 012777 000207 171412 WARN1:  
2298 011060 012700 060000  
2299 011064 010004  
2300 011066 012701 040000  
2301 011072 010103  
2302 011074 005002  
2303 011076 104424  
2304 011100  
2305 011106 022737 000001 003710  
2306 011114 001403  
2307 011116 004737 011516  
2308 011122 000402  
2309 011124 004737 177640 1\$:  
2310 011130 104417 2\$:  
2311 011132 104423  
2312 011134 000416  
2313 011136 005037 002100 TAG2\$:  
2314 011142  
2315 011164 005037 002074  
2316 011170 000564

```

2317 011172 005737 002066 TAG3$: TST NEMCNT ;ANY TRAPS?
2318 011176 001401 BEQ 1$ ;NO - SKIP
2319 011200 000671 BR TAG9$ ;NOW - TRY NEXT BANK
2320 011202 104424 1$: CACHOFF ;TURN CACHE OFF
2321 011204 TESTAREA ;ENTER SUPERVISOR MODE
2322 011212 004777 171256 CALL @LINK2 ;FINISH PATTERN
2323 011216 104417 KERNEL ;ENTER KERNEL MODE
2324 011220 104423 CACHON ;TURN CACHE ON
2325 011222 005737 002072 TST PATERR ;ANY PATTERN ERRORS
2326 011226 001040 BNE 2$ ;YES - SKIP
2327 011230 005737 002070 TST PARCNT ;ANY PARITY ERRORS
2328 011234 001035 BNE 2$ ;YES - SKIP
2329 011236 005737 002066 TST NEMCNT ;ANY NON EXISTANT MEMORY
2330 011242 001032 BNE 2$ ;YES - SKIP
2331 011244 012700 060000 MOV #FIRST,R0
2332 011250 010004 MOV R0,R4
2333 011252 012701 040000 MOV #SIZE,R1
2334 011256 010103 MOV R1,R3
2335 011260 013702 002554 MOV ONES,R2 ;DATA IS ONES
2336 011264 012777 000240 171200 MOV #000240,@LINK1 ;PUT 'NOP' INSTRUCTION BACK IN SUBROUTINE
2337 011272 104424 CACHOFF ;TURN CACHE OFF
2338 011274 TESTAREA ;ENTER TEST MODE
2339 011302 022737 000001 003710 CMP #1,PROTYP ;IS THIS AN 11/44?
2340 011310 001403 BEQ 5$ ;BRANCH IF IT IS
2341 011312 004737 011516 CALL MTST3 ;DO IN MEMORY IF NOT
2342 011316 000402 BR 6$ ;JUMP OVER NEXT INSTRUCTION
2343 011320 004737 177640 5$: CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
2344 011324 104417 6$: KERNEL ;ENTER KERNEL MODE
2345 011326 104423 CACHON ;TURN CACHE ON
2346 011330 013700 002102 2$: MOV BANKINDEX,R0
2347 011334 005737 002072 TST PATERR ;ANY PATTERN ERRORS?
2348 011340 001006 BNE 3$ ;YES - SKIP
2349 011342 005737 002070 TST PARCNT ;ANY PARITY ERRORS?
2350 011346 001003 BNE 3$ ;YES - SKIP
2351 011350 005737 002066 TST NEMCNT ;ANY HOLES?
2352 011354 001406 BEQ 4$ ;NONE - SKIP
2353 011356 052760 000001 002624 3$: BIS #BIT0,CONFIG(R0) ;SET ERROR BIT IN THIS BANK
2354 011364 SET CONFGERROR ;FORCE PRINTING OF CONFIGURATION TABLE
2355 011372 053760 002104 002624 4$: BIS CPUBIT,CONFIG(R0) ;SET ACCESSED BIT
2356 011400 000137 010764 JMP TAG9$
2357
2358 ;TEST A PROTECTED BANK
2359 011404 TSTBANK: PUSH R1
2360 011406 012737 000001 002076 MOV #1,NONEM ;SET NON-EXISTANT MEMORY TO COUNT
2361 011414 012700 060000 MOV #FIRST,R0
2362 011420 012701 020000 MOV #20000,R1
2363 011424 104424 CACHOFF ;TURN CACHE OFF
2364 011426 TESTAREA ;ENTER TEST MODE
2365 011434 005720 4$: TST (R0)+
2366 011436 077102 SOB R1,4$
2367 011440 104417 KERNEL ;ENTER KERNEL MODE
2368 011442 104423 CACHON ;TURN CACHE ON
2369 011444 012737 000002 002076 MOV #2,NONEM ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
2370 011452 POP R1
2371 011454 IF PARCNT NE #0
2372 011462 052761 000001 002624 BIS #BIT0,CONFIG(R1) ;ERROR BANK
2373 011470 SET CONFGERROR
    
```

2374	011476				END ;OF IF PARCNT	
2375	011476				IF NEMCNT EQ #0	
2376	011504	053761	002104	002624	BIS (PUBIT,CONFIG(R1))	;ACCESSED BANK
2377	011512				END ;OF IF NEMCNT	
2378	011512	000137	010764		JMP TAG9\$	
2379	011516	010220			MTST3: MOV R2,(R0)+	;V177640
2380	011520	077102			SOB R1,MTST3	;V177642
2381	011522	000240			NOP	;V177644
2382	011524	012401			2\$: MOV (R4)+,R1	;V177646
2383	011526	020102			CMP R1,R2	;V177650
2384	011530	001402			BEQ 3\$	;V177652
2385	011532	005237	002072		INC PATERR	;V177654
2386	011536	077306			3\$: SOB R3,2\$	;V177660
2387	011540	000207			RETURN	;V177662

2389 011542

```

SUBAAI: SUBST <<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.
CLR BANK ;RESET BANK TO ZERO
SHADL1: CALL EXBANK ;SET BANK PARAMETERS
MOV BANKINDEX,R0
IF ACFLAG IS TRUE AND INTFLAG IS TRUE
IF INT64K IS TRUE
ADD #20,R0 ;POINT TO BANKINDEX + 4
ADD #10,BANK ;POINT TO BANK + 8
ELSE
ADD #40,R2 ;POINT TO BANKINDEX + 8
ADD #20,BANK ;POINT TO BANK + 16
END; OF IF INT64K
BIS #BIT7,CONFIG(R0) ;MAKE NEW BANK PROGRAM SPACE
ELSE
INC BANK ;GO TO NEXT BANK
END; OF IF ACFLAG
CMP LASTBANK,BANK ;HAVE WE DONE ALL THE BANKS?
BGE SHADL1 ;BRANCH IF NOT
  
```

2390

2391

2392

2393 011542 005037 002100

2394 011546 004737 044240

2395 011552 013700 002102

2396 011556

2397 011572

2398 011600 062700 000020

2399 011604 062737 000010 002100

2400 011612

2401 011614 062702 000040

2402 011620 062737 000020 002100

2403 011626

2404 011626 052760 000200 002624

2405 011634

2406 011636 005237 002100

2407 011642

2408 011642 023737 002526 002100

2409 011650 002336

2412 011652

```

NEWST <<ECC INHIBIT MODE POINTER TEST>>
:*****
:*TEST 4      ECC INHIBIT MODE POINTER TEST
:*****
TST4:  SCOPE
:THE MS11-M OR MF11S-K INHIBIT ECC DISABLE AND DIAGNOSTIC CHECK MODE
:ON THE BOTTOM FIRST OR SECOND 16K WORDS CONTROLLED BY A CSR. THIS
:IS CONSIDERED TO BE A PROTECTED BANK BY THE PROGRAM. IT MAY BE
:QUITE COMPLEX TO DETERMINE ON A GIVEN SYSTEM CONFIGURATION WHICH
:BANKS CAN BE PROTECTED;
:SO
:THIS ROUTINE ATTEMPS TO CREATE A DOUBLE BIT ERROR IN ADDRESS 0 & 2
:OF EVERY ECC BANK. ECC HARDWARE WILL PREVENT THIS FROM HAPPENING
:IN PROTECTED BANKS WHICH SHOULD ALWAYS INCLUDE BANK ZERO - WHERE
:THE PROGRAM IS.
:
:WARNING:!!!!!!!!!!!!
: IN CASE OF HARDWARE FAILURE IT IS COMMON THAT A DOUBLE BIT ERROR
: WILL BE CREATED ON THE KERNEL STACK & "CRASH" THE DIAGNOSTIC
: DURING THIS ROUTINE. YOUR ONLY CLUE IS THAT YOU CAN GET AS FAR AS
: THIS ROUTINE BUT NOT PAST IT.
CACHOFF ;TURN CACHE OFF
MOV #-1,OLDCSR
FOR BANK := #0 TO LASTBANK
MOV #FIRST,R1 ;SET UP VIRT ADDR POINTER
CALL EXBANK
MOV BANKINDEX,R0
IF ACFLAG IS TRUE
IF MKFLAG IS TRUE
IF SKIPMK IS FALSE
IF INTFLAG IS TRUE
MOV #40000,R3 ;SET INDEX COUNTER
MOV #1,SPLTCSR ;MAP AS INTERLEAVED BANK
ELSE
MOV #2,R3 ;SET INDEX COUNTER
END; OF IF INTFLAG
MOVB CONFIG+1(R0),R2
ASL R2
BIC #^C36,R2
MOV R2,CSRNO
IF CSRNO NE OLDCSR
MOV CSRNO,OLDCSR
IF PFLAG IS FALSE
BIS #BIT6,CONFIG(R0)
END; OF IF PFLAG
CALL IMPTEST
    
```

011652 000004

2413  
 2414  
 2415  
 2416  
 2417  
 2418  
 2419  
 2420  
 2421  
 2422  
 2423  
 2424  
 2425  
 2426  
 2427  
 2428  
 2429  
 2430 011654 104424  
 2431 011656 012737 177777 002150  
 2432 011664  
 2433 011670 012701 060000  
 2434 011674 004737 044240  
 2435 011700 013700 002102  
 2436 011704  
 2437 011712  
 2438 011720  
 2439 011726  
 2440 011734 012703 040000  
 2441 011740 012737 000001 002232  
 2442 011746  
 2443 011750 012703 000002  
 2444 011754  
 2445 011754 116002 002625  
 2446 011760 006302  
 2447 011762 042702 177741  
 2448 011766 010237 002146  
 2449 011772  
 2450 012002 013737 002146 002150  
 2451 012010  
 2452 012016 052760 000100 002624  
 2453 012024  
 2454 012024 004737 012160

```

2456 012030
2457 012036 116002 002625
2458 012042 072227 177775
2459 012046 042702 177741
2460 012052 010237 002146
2461 012056 062701 000002
2462 012062 004737 012160
2463 C 0066 005037 002232
2464 01 072
2465 012072
2466 012072
2467 012072
2468 012072
2469 012072
2470 012106
2471 012122 005037 002100
2472 012126
2473 012144 104506
2474 012146
2475 012150 104472
2476 012152
2477 012152 104423
2478 012154 000137 012422

```

```

IF INTFLAG IS TRUE
MOVW CONFIG+1(R0),R2
ASH #-3,R2
BIC #^C36,R2
MOV R2,CSRNO
ADD #2,R1 ;FIX POINTER FOR A1 ASSERTED HALF
CALL IMPTST
CLR SPLTCSR
END; OF IF INTFLAG
END; OF IF CSRNO
END; OF IF SKIPMK
END; OF IF MKFLAG
END; OF IF ACFLAG
END; OF FOR BANK
MAP ;MAP TEST SPACE TO BANK 0
CLR BANK
IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
ENASBE ;TRAP ON SINGLE BIT ERRORS
ELSE
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
END; OF IF #SWO
CACHON ;TURN THE CACHE BACK ON
JMP SUBAAR ;JUMP OVER THE SUBROUTINE

```

```

2480 012160 005004          IMPTEST:CLR      R4
2481 012162                MAP BANK                ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
2482 012176 005005          CLR      R5
2483 012200 012737 020000 002144  MOV     #BIT13,CSR
2484 012206                TESTAREA                ;ENTER TEST MODE
2485 012214                PUSH     (R1)                ;SAVE TEST LOCATION
2486 012216 060301          ADD      R3,R1                ;INDEX TO NEXT LOCATION
2487 012220                PUSH     (R1)                ;SAVE TEST LOCATION
2488 012222 104505          CHK1DIS                ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2489 012224 010411          MOV     R4,(R1)                ;WRITE CHECKBITS (ALL ZEROS)
2490 012226 160301          SUB     R3,R1
2491 012230 010411          MOV     R4,(R1)
2492 012232 104503          CLR1CSR                ;CLEAR CSR
2493 012234 005711          TST     (R1)                ;READ CHECKBITS INTO REAL CSR
2494 012236 104501          WAS1DBE                ;WAS THERE A DOUBLE BIT ERROR
2495
2496
2497
2498 012240                ON.NOERROR ;1
2499 012242 012737 020000 002144  MOV     #BIT13,CSR
2500 012250 104505          CHK1DIS                ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2501 012252 013711 002554          MOV     ONES,(R1)
2502 012256 060301          ADD     R3,R1
2503 012260 013711 002554          MOV     ONES,(R1)
2504 012264 160301          SUB     R3,R1
2505 012266 104503          CLR1CSR                ;CLEAR CSR
2506 012270 005711          TST     (R1)
2507 012272 104501          WAS1DBE                ;WAS THERE A DOUBLE BIT ERROR
2508 012274                ON.NOERROR ;2
2509 012276 012737 027400 002144  MOV     #27400,CSR
2510 012304 104505          CHK1DIS                ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2511 012306 010411          MOV     R4,(R1)
2512 012310 060301          ADD     R3,R1                ;ADD INDEX TO GET TO SECOND WORD
2513 012312 010411          MOV     R4,(R1)
2514 012314 160301          SUB     R3,R1                ;SUBTRACT INDEX TO FIRST WORD
2515 012316 104503          CLR1CSR                ;CLEAR CSR
2516 012320 005711          TST     (R1)
2517 012322 104501          WAS1DBE                ;WAS THERE A DOUBLE BIT ERROR
2518 012324                ON.NOERROR ;3
2519 012326 012737 074000 002144  MOV     #74000,CSR
2520 012334 104505          CHK1DIS                ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2521 012336 010411          MOV     R4,(R1)
2522 012340 060301          ADD     R3,R1                ;INDEX TO SECOND WORD
2523 012342 010411          MOV     R4,(R1)
2524 012344 104503          CLR1CSR                ;CLEAR CSR
2525 012346 160301          SUB     R3,R1                ;GO BACK TO FIRST WORD
2526 012350 005711          TST     (R1)
2527 012352 104501          WAS1DBE                ;WAS THERE A DOUBLE BIT ERROR
2528 012354                END ;OF ON.NOERROR ;3
2529 012354                END ;OF ON.NOERROR ;2
2530 012354                END ;OF ON.NOERROR ;1
2531 012354                ON.ERROR
2532 012356 005205          INC     R5                ;IDENTIFY AS BAD BANK
2533 012360                END ;OF ON.ERROR
2534 012360 104471          ECC1DIS                ;DISABLE ERROR CORRECTION
2535 012362 010411          MOV     R4,(R1)                ;CLEAR OUT DOUBLE BIT ERROR'
2536 012364 060301          ADD     R3,R1                ;INDEX TO SECOND WORD
  
```

2537	012366	010411		MOV	R4,(R1)		;CLEAR OUT DOUBLE BIT ERROR!
2538	012370	104503		CLR1CSR			
2539	012372	005705		TST	R5		
2540	012374	001405		BEQ	1\$		
2541	012376	050560	002624	BIS	R5,CONFIG(R0)		
2542	012402	105260	002626	INCB	CONFIG+2(R0)		
2543	012406	104036		ERROR	+36		
2544	012410		1\$:	POP	(R1)		;RESTORE TEST LOCATION (2ND WORD)
2545	012412	160301		SUB	R3,R1		;GO BACK TO FIRST WORD
2546	012414			POP	(R1)		;RESTORE TEST LOCATION (1ST WORD)
2547	012416	104417		KERNEL			
2548	012420	000207		RETURN			



2892  
2893 012422

SUBAAR: SET STOPOK

;PROGRAM CAN NOW BE HALTED

2896 012430

2897 012430 012700 000020  
 2898 012434 012701 002432  
 2899 012440 005021  
 2900 012442 077002  
 2904 012444  
 2905 012450 004737 044240  
 2906 012454 013700 002102  
 2928  
 2929 012460  
 2930 012466 116003 002625  
 2931 012472 042703 177760  
 2932 012476 006303  
 2933 012500 005263 002432  
 2934 012504  
 2935  
 2936 012512  
 2937 012512  
 2938 012520 116003 002625  
 2939 012524 010304  
 2940 012526 042703 177760  
 2941 012532 072427 177774  
 2942 012536 042704 177760  
 2943 012542  
 2944 012546 042760 014000 002626  
 2945 012554 042760 170000 002624  
 2946 012562  
 2947 012564  
 2948 012564  
 2949 012572  
 2950 012574  
 2951 012574 006303  
 2952 012576 006304  
 2953 012600 005263 002432  
 2954 012604 005264 002432  
 2955 012610  
 2956 012612  
 2957 012614  
 2958 012614  
 2959 012620  
 2960 012630  
 2961 012636  
 2962 012636  
 2963 012636  
 2964 012636  
 2965 012652  
 2966 012656 005000  
 2967 012660 005001  
 2968 012662 005005  
 2969 012664 005037 013074  
 2970 012670 022761 000010 002432 2\$:  
 2971 012676 002043  
 2972 012700 022761 000020 002432  
 2973 012706 002003

```

SUBTST <<LEGAL CONFIGURATION CHECK>>
:*****
:*SUBTEST      LEGAL CONFIGURATION CHECK
:*****
1$:  MOV      #16.,R0
      MOV      #CSRINFO,R1
      CLR      (R1)+
      SOB      R0,1$
      FOR BANK := #0 TO LASTBANK
      CALL     EXBANK
      MOV      BANKINDEX,R0

      IF ACFLAG IS TRUE
      MOV      CONFIG+1(R),R3
      BIC      #^C17,R3
      ASL      R3
      INC      CSRINFO(R3)
      IF MKFLAG IS TRUE
      ;MAKE SURE THAT EACH BANK HAS NO MORE THAN 2 CSRS
      BEGIN   LEGALCSR
      IF INTFLAG IS TRUE
      MOV      CONFIG+1(R0),R3
      MOV      R3,R4
      BIC      #^C17,R3
      ASH      #-4,R4
      BIC      #^C17,R4
      IF R3 EQ R4
      BIC      #BIT11!BIT12,CONFIG+2(R0)
      BIC      #170000,CONFIG(R0)
      LEAVE   LEGALCSR
      END; OF IF R3
      IF KFLAG IS FALSE
      LEAVE   LEGALCSR
      END; OF IF KFLAG
      ASL      R3
      ASL      R4
      INC      CSRINFO(R3)
      INC      CSRINFO(R4)
      ELSE
      LEAVE   LEGALCSR
      END; OF IF INTFLAG
      TYPE     MSG124
      TYPOCS   BANK,<TYPE BANK #>,3
      SET      CONFGERROR
      END      LEGALCSR
      END; OF IF MKFLAG
      END; OF IF ACFLAG
      END; OF FOR BANK
      PUSH     R5,R0
      CLR      R0
      CLR      R1
      CLR      R5
      CLR      MBERR
      ;SAVE CONTENTS OF R5, R0
      ;CLEAR REGISTERS

      CMP      #10,CSRINFO(R1)
      BGE      5$
      ;CLEAR ERROR INDICATOR
      ;IS CURRENT CSR <= 10
      ;BRANCH IF SO
      CMP      #20,CSRINFO(R1)
      BGE      3$
      ;IS CURRENT CSR < 20
      ;BRANCH IF SO
  
```

2974	012710	004737	013216		CALL	ILLCSR			;CALL ERROR ROUTINE
2975	012714	000434			BR	5\$			;TRY NEXT CSR
2976	012716	016005	002624	3\$:	MOV	CONFIG(R0),R5			;MOVE LOW WORD TO R5
2977	012722	032705	000002		BIT	#BIT1,R5			;DOES MEMORY EXIST HERE?
2978	012726	001415			BEQ	4\$			;BRANCH IF NOT
2979	012730	042705	170377		BIC	#^C7400,R5			;ISOLATE CSR NUMBER IN
2980	012734	072527	177771		ASH	#-7,R5			;REGISTER 5
2981	012740	020501			CMP	R5,R1			;IS IT THE CURRENT CSR?
2982	012742	001007			BNE	4\$			;TRY NEXT WORD OF CONFIG IF NOT
2983	012744	032760	010000	002626	BIT	#BIT12,CONFIG+2(R0)			;IS IT INTERLEAVED?
2984	012752	001003			BNE	4\$			;BRANCH IF SO
2985	012754	012737	000001	013074	MOV	#1,MBERR			;SET ERROR INDICATOR
2986	012762	062700	000004	4\$:	ADD	#4,R0			;UPDATE CONFIG COUNTER
2987	012766	022700	000340		CMP	#340,R0			;CONFIG TABLE ALL DONE?
2988	012772	001351			BNE	3\$			;BRANCH IF NOT
2989	012774	005737	013074		TST	MBERR			;ERRORS FOUND?
2990	013000	001402			BEQ	5\$			;TRY NEXT CSR IF NOT
2991	013002	004737	013216		CALL	ILLCSR			;CALL ERROR ROUTINE
2992	013006	005000		5\$:	CLR	R0			;REINITIALIZE CONFIG COUNTER
2993	013010	005037	013074		CLR	MBERR			;CLEAR ERROR INDICATOR
2994	013014	062701	000002		ADD	#2,R1			;UPDATE CSR COUNTER
2995	013020	022701	000040		CMP	#40,R1			;ALL CSR'S DONE?
2996	013024	001321			BNE	2\$			;BRANCH IF NOT
2997	013026				POP	R0,R5			;RESTORE REGISTERS
2998	013032	005037	013074		CLR	MBERR			;RESET ERROR INDICATOR
3002	013036	005001			CLR	R1			;CLEAR
3003	013040	005000			CLR	R0			;COUNTERS
3004	013042	032760	000002	002624	6\$:	BIT	#BIT1,CONFIG(R0)		;MEMORY PRESENT?
3005	013050	001404			BEQ	7\$			;BRANCH IF NOT
3006	013052	005201			INC	R1			;ADD ONE BANK TO COUNT
3007	013054	062700	000004		ADD	#4,R0			;UPDATE COUNTER
3008	013060	000770			BR	6\$			;TRY NEXT BANK
3009	013062	062701	177777	7\$:	ADD	#-1,R1			;ADJUST COUNT
3010	013066	010137	002526		MOV	R1,LASTBANK			;STORE COUNT
3011	013072	000402			BR	SKUJ			
3012	013074	000000			MBERR:	.WORD 0			;SAVE SPACE FOR ERROR INDICATOR
3013	013076	000000			PHEBE:	.WORD 0			;SAVE SPACE FOR ODD BOUNDARY INTERLEAVED INDICATOR
3014	013100	005000			SKUJ:	CLR R0			;CLEAR CONFIG COUNTER
3015	013102	005037	013076		CLR	PHEBE			;CLEAR COUNTER
3016	013106	032760	000002	002624	1\$:	BIT	#BIT1,CONFIG(R0)		;IS THERE MEMORY PRESENT?
3017	013114	001431			BEQ	3\$			;BRANCH IF NOT
3018	013116	032760	010000	002626	BIT	#BIT12,CONFIG+2(R0)			;IS IT INTERLEAVED?
3019	013124	001005			BNE	2\$			;BRANCH IF SO
3020	013126	005237	013076		INC	PHEBE			;INCREMENT COUNTER
3021	013132	062700	000004		ADD	#4,R0			;INCREMENT CONFIG COUNTER
3022	013136	000763			BR	1\$			;TRY NEXT BANK
3023	013140	023727	013076	000010	2\$:	CMP	PHEBE,#10		;IS THE COUNTER EQUAL TO...
3024	013146	001417			BEQ	4\$			;ONE OF THE SPECIAL VALUES.
3025	013150	023727	013076	000030	CMP	PHEBE,#30			;IF IT IS...
3026	013156	001413			BEQ	4\$			;BRANCH TO 4\$
3027	013160	023727	013076	000050	CMP	PHEBE,#50			
3028	013166	001407			BEQ	4\$			
3029	013170	023727	013076	000070	CMP	PHEBE,#70			
3030	013176	001403			BEQ	4\$			
3031	013200	005037	013076	3\$:	CLR	PHEBE			;CLEAR INDICATOR
3032	013204	000403			BR	5\$			
3033	013206	012737	000001	013076	4\$:	MOV	#1,PHEBE		;SET INDICATOR

3034	013214	000421	
3035	013216	010102	
3036	013220	006202	
3037	013222	022702	000012
3038	013226	100002	
3039	013230	062702	000007
3040	013234	062702	000060
3041	013240	110237	074767
3042	013244		
3043	013250		
3044	013256	000207	

5\$:	BR	SUBAAP	:BRANCH TO NEXT SUBTEST
ILLCSR:	MOV	R1,R2	:R2 HAS CSR NUMBER
	ASR	R2	:MAKE ACCEPTABLE FOR PRINTING
	CMP	#10.,R2	
	RPL	1\$	
	ADD	#7,R2	
1\$:	ADD	#60,R2	
	MOVB	R2,MSGA122	:PUT NUMBER INTO ERROR MESSAGE
	TYPE	MSG122	
	SET	CONFGERROR	
	RETURN		

3047 013260

```
SUBAAP: SUBTST <<PRINT CONFIGURATION DETAILS>>
:*****
:*SUBTEST PRINT CONFIGURATION DETAILS
:*****
```

3048 013260  
3049 013304 013702 002526  
3050 013310 006302  
3051 013312 006302

```
CLEAR BSIZE,KSIZE,LSIZE,MSIZE,PSIZE
MOV LASTBANK,R2
ASL R2
ASL R2
FOR R1 := #0 TO R2 BY #4
IF CPUBIT SET.IN CONFIG(R1)
IF #BIT10 SET.IN CONFIG+2(R1)
IF #BIT8 SET.IN CONFIG+2(R1)
IF #BIT9 SET.IN CONFIG+2(R1)
LET PSIZE := PSIZE + #1
ELSE
LET KSIZE := KSIZE + #1
END;IF BIT9
ELSE
IF #BIT9 SET.IN CONFIG+2(R1)
LET LSIZE := LSIZE + #1
ELSE
LET MSIZE := MSIZE + #1
END; IF BIT9
END;IF BIT8
ELSE
IF #BIT9 SET.IN CONFIG+2(R1)
IF #BIT8 SET.IN CONFIG+2(R1)
LET BSIZE := BSIZE + #1
END; OF IF #BIT8
END; OF IF #BIT9
END;IF BIT10
END; OF IF CPUBIT
END ;OF FOR ALL BANKS IN TABLE
```

3052 013314  
3053 013316  
3054 013326  
3055 013336  
3056 013346  
3057 013356  
3058 013362  
3059 013364  
3060 013370  
3061 013370  
3062 013372  
3063 013402  
3064 013406  
3065 013410  
3066 013414  
3067 013414  
3068 013414  
3069 013416  
3070 013426  
3071 013436  
3072 013442  
3073 013442  
3074 013442  
3075 013442  
3076 013442  
3077

3078 013452 005037 002422  
3079 013456  
3080 013460 006361 002344  
3081 013464 006361 002344  
3082 013470 006361 002344  
3083 013474 006361 002344  
3084 013500 066137 002344 002422  
3085 013506  
3086 013520  
3087 013522  
3088 013532  
3089 013536

```
CLR I
FOR R1 := #0 TO #10 BY #2
ASL BSIZE(R1)
ASL BSIZE(R1)
ASL BSIZE(R1)
ASL BSIZE(R1) ;BSIZE(R1) := BSIZE(R1) * 16.
ADD BSIZE(R1),I ;i <- i + BSIZE(R1)
END; FOR R1
FOR R1 := #0 TO #200 BY #4
IF CPUBIT SET.IN CONFIG(R1)
LET UNITOP := UNITOP + #1
END; OF IF CPUBIT
END; OF FOR R1
```

3091 013550 006337 002366  
3092 013554 006337 002366  
3093 013560 006337 002366  
3094 013564 006337 002366  
3095 013570  
3096 013606  
3097 013612 005737 002344  
3098 013616 001405  
3099 013620  
3100 013626

```
ASL UNITOP
ASL UNITOP
ASL UNITOP
ASL UNITOP ;UNITOP := UNITOP * 16.
IF I LT UNITOP THEN LET I := UNITOP
TYPE $CRLF
TST BSIZE
BEQ 1$
TYPDEC BSIZE
TYPE MSG071
```

3101	013632	005737	002346	1\$:	TST	KSIZE
3102	013636	001405			BEQ	2\$
3103	013640				TYPDEC	KSIZE
3104	013646				TYPE	MSG072
3105	013652	005737	002350	2\$:	TST	LSIZE
3106	013656	001405			BEQ	3\$
3107	013660				TYPDEC	LSIZE
3108	013666				TYPE	MSG112
3109	013672	005737	002352	3\$:	TST	MSIZE
3110	013676	001405			BEQ	4\$
3111	013700				TYPDEC	MSIZE
3112	013706				TYPE	MSG113
3113	013712	005737	002354	4\$:	TST	PSIZE
3114	013716	001405			BEQ	5\$
3115	013720				TYPDEC	PSIZE
3116	013726				TYPE	MSG114
3117	013732			5\$:	TYPDEC	I
3118	013740				TYPE	MSG070
3119	013744				IF #SW6	OFF.IN @SWR
3120	013754	004737	036570		CALL	PCONFIG
3121	013760				END; OF	IF #SW6

3124 013760

3125 013760  
3126 013774 005037 002404  
3127 014000 012700 062516  
3128 014004  
3129 014006  
3130 014014 111001  
3131 014016 042701 177400  
3132 014022  
3133 014030 000261  
3134 014032  
3135 014034 000241  
3136 014036  
3137 014036 006101  
3138 014040 005201  
3139 014042 006301  
3140 014044 006301  
3141 014046 006301  
3142 014050 006301  
3143 014052 163701 002404  
3144 014056 010137 002404  
3145 014062  
3146 014072 060137 002372  
3147 014076  
3148 014076  
3149 014106 060137 002374  
3150 014112  
3151 014112 062700 000004  
3152 014116  
3153 014116  
3154 014126  
3155 014146 104046  
3156 014150  
3168 014150

```
SUBTST <<CHECK APT SIZING>>
:*****
:*SUBTEST CHECK APT SIZING
:*****
IF APTFLAG IS TRUE AND APTSIZE IS TRUE
  CLR TEMP
  MOV #SMAMS1,R0
  FOR R2 := #0 TO #4
    IFB 1(R0) NE #0
      MOVB (R0),R1
      BIC #177400,R1
      IF 2(R0) LT #0
        SEC
      ELSE
        CLC
      END ;OF IF 2(R0)
      ROL R1
      INC R1 ;TO COMPENSATE FOR 4 BANKS BEING (0-3)
      ASL R1
      ASL R1
      ASL R1
      ASL R1
      SUB TEMP,R1
      MOV R1,TEMP
      IFB 1(R0) EQ #3
        ADD R1,APTPAR
      END ;OF IFB 1(R0)
      IFB 1(R0) EQ #4
        ADD R1,APTECC
      END ;OF IFB 1(R0)
      ADD #4,R0
    END ;OF IFB 1(R0)
  END ;OF FOR R2
IF APTPAR NE LSIZE OR APTECC NE MSIZE
  ERROR +46
END ;OF IF APTPAR
END ;OF IF APTFLAG
```

3170 014150

```
LOOP: NEWTST <<DIAGNOSTIC MODE DISPATCH ROUTINE>>  
:*****  
:*TEST 5 DIAGNOSTIC MODE DISPATCH ROUTINE  
:*****
```

014150 000004  
3171 014152 005037 002214  
3172 014156 017700 166414  
3173 014162 042700 177761  
3174 014166 004770 014176  
3175 014172 000137 014216  
3176 014176 014650  
3177 014200 014756  
3178 014202 015064  
3179 014204 015214  
3180 014206 015344  
3181 014210 015474  
3182 014212 015646  
3183 014214 015776  
3184  
3185 014216 004737 014550  
3186  
3187 014222

```
TST5: SCOPE  
CLR CONTFLAG ;GET SWITCHES  
MOV @SWR,RO ;MASK TO ONLY MODE BITS  
BIC #^C16,RO ;DISPATCH TO ROUTINE THROUGH NEXT TABLE  
CALL @DISPTBL(RO) ;GO TO NEXT TEST  
JMP MEMDONE  
DISPTBL:BAFPAF ;MODE 0;BANKS FORWARD, PATTERNS FORWARD  
BAFPAR ;MODE 1;BANKS FORWARD, PATTERNS REVERSE  
BAWPAF ;MODE 2;BANKS WORST FIRST, PATTERNS FORWARD  
BAWPAR ;MODE 3;BANKS WORST FIRST, PATTERNS REVERSE  
PAFBAF ;MODE 4;PATTERNS FORWARD, BANKS FORWARD  
PAFBAW ;MODE 5;PATTERNS FORWARD, BANKS WORST FIRST  
PARBAF ;MODE 6;PATTERNS REVERSE, BANKS FORWARD  
PARBAW ;MODE 7;PATTERNS REVERSE, BANKS WORST FIRST  
MEMDONE:CALL DOBACK ;CHECK BACKGROUND PATTERN
```

```
NEWTST<<UNIQUE BANK TEST>>  
:*****  
:*TEST 6 UNIQUE BANK TEST  
:*****
```

014222 000004  
3188  
3189  
3190 014224  
3191 014232  
3192 014246 004737 024172  
3193 014252  
3194 014260 005037 002106  
3195 014264  
3196 014264 004737 014550  
3200  
3201 014270

```
TST6: SCOPE  
;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA  
;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)  
IF SELONLY IS FALSE  
SET HEADER,MUT  
CALL MT0027  
SET HEADER  
CLR MUT  
END ;OF IF SELONLY  
CALL DOBACK ;RESTORE BACKGROUND PATTERN
```

```
FLUSH: SUBTST <<FLUSH OUT DBE'S>>  
:*****  
:*SUBTEST FLUSH OUT DBE'S  
:*****  
CALL MT0030
```

3202 014270 004737 024656



```

3205
3206
3207
3208
3209
3210
3211
3212 014274 005037 002412
3213 014300 012700 002626
3214 014304 042710 020000
3215 014310 062700 000004
3216 014314 020027 003620
3217 014320 003771
3218 014322 013737 002570 002014
3219 014330 005237 062474
3220 014334 042737 100000 062474
3221 014342
3222 014346
3223 014364
3224 014370 005037 002316
3225 014374
3226 014374
3227 014402 013700 000042
3228 014406 001456
3229 014410 022700 002000
3230 014414 001453
3231
3232 014416
3233 014420 004737 045126
3234 014424
3235 014426 000005
3236 014430 004710
3237 014432 000240
3238 014434 000240
3239 014436 000240
3240 014440
3241
3242
3243
3244
3245 014440 013706 002534
3246 014444 005737 002424
3247 014450 001003
3248 014452 052737 000060 172516
3249 014460 104420
3250 014462 013700 002536
3251 014466 012701 000001
3252 014472 004737 043710
3253 014476
3254 014504
3255 014514 012701 000050
3256 014520 077001
3257 014522 062737 000001 062476
3258 014530 005537 062500
3259 014534 077107
3260 014536 005237 062474
3261 014542 000764

```

```

.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
$EOP: CLR FSINFLAG
MOV #CONFIG+2,RO ;MOVE 2ND WORD OF CONFIG TO RO
1$: BIC #BIT13,(RO) ;CLEAR BACKGROUND VALID BIT
ADD #4,RO ;INCREMENT TO NEXT BANK
CMP RO,#3620 ;DONE?
BLE 1$ ;NO - BRANCH
MOV $ERTTL,LASTERROR
INC $PASS ;INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
TYPE MSG077 ;TYPE 'END PASS #'
IF #SW11 SET IN @SWR OR QVFLAG IS TRUE
TYPE MSG035 ;QV
CLR QVFLAG
END ;OF IF SW11
TYPDEC $PASS
MOV 42,RO ;GET MONITOR ADDRESS
BEQ $DOAGAIN ;BRANCH IF NO MONITOR
$ZAP42: CMP #STACK,RO ;ARE WE UNDER RT11
BEQ $DOAGAIN ;YES - BRANCH
;WE ARE UNDER (HEAVEN HELP US) XXDP!
PUSH RO
CALL SHUTUP
POP RO
RESET ;CLEAR THE WORLD
$ENDAD: CALL (RO) ;GO TO MONITOR
NOP ;SAVE ROOM
NOP ;FOR
NOP ;ACT11
$DOAGN: ;UNDO SHUTUP STUFF
; RESTORE STACK
; ENERGIZE UNIBUS MAP & 22 BIT ADDRESSING
; ENERGIZE MEMORY MANAGEMENT
; PUT LOADERS BACK HOME
MOV KSTACK,SP
TST NO22BIT ;IS THIS AN 11/44 OR 11/24?
BNE 1$
BIS #BITS!BIT4,MMR3
1$: ENERGIZE ;TURN ON MEMORY MANAGEMENT
MOV LOADHOME,RO ;DESTINATION BANK
MOV #1,R1 ;SOURCE BANK
CALL BANKMOV
IF APTFLAG IS TRUE
IF $USWR EQ $PASS
APTHANG: MOV #50,R1
2$: SOB RO,2$
ADD #1,$DEVCT
ADC $UNIT
SOB R1,2$
INC $PASS
BR APTHANG

```

CZMSDBO MS11-L/M DIAGNOSTIC  
END OF PASS ROUTINE

MACRO M1113 07-OCT-80 18:01 PAGE 103-1 SEQUENCE 162

N 13

SEQ 0169

3262 014544  
3263 014544  
3264 014544 000137 014150

END ;OF IF \$USWR  
END ;OF IF APTFLAG  
\$DOAGAIN: JMP LOOP ;RETURN

3267 014550

DOBACK: SUBTST <<WRITE BACKGROUND PATTERNS>>

\*\*\*\*\*  
;SUBTEST WRITE BACKGROUND PATTERNS  
\*\*\*\*\*

3268 014550 005037 002110  
3269 014554  
3270 014560 004737 044240  
3271 014564  
3272 014600  
3273 014614 004737 017456  
3274 014620 005037 002106  
3275 014624  
3276 014632  
3277 014632  
3278 014646 000207

CLR PATTERN  
FOR BANK := #0 TO LASTBANK  
CALL EXBANK  
IF ACFLAG IS TRUE AND RRFLAG IS FALSE  
SET HEADER,MUT  
CALL MKTEST ;CALL MJTEST WOULD ALSO WORK  
CLR MUT  
SET HEADER  
END ;OF IF ACFLAG  
END ;OF FOR BANK  
RETURN

```

3281
3282
3283 014650

3284 014650 005037 002100
3285
3286 014654 004737 044240
3287 014660 005737 002114
3288 014664 001412
3289 014666 005737 002122
3290 014672 001007
3291 014674 005037 002110
3292
3293 014700 004737 016150
3294
3295 014704 004737 044726
3296 014710 001373
3297
3298 014712 005037 002214
3299 014716 004737 044752
3300 014722 002354
3301
3302 014724 005737 002124
3303 014730 001401
3304 014732 000207
3305 014734 004737 042466
3306 014740
3307
3308 014744 004737 014650
3309 014750 004737 043356
3310 014754 000207
    
```

```

.SBTTL MTEST MODES
BAFPAF: SUBTST <<BANKS FORWARD,PATTERNS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST BANKS FORWARD,PATTERNS FORWARD **RECURSIVE**
:*****
          CLR BANK ;SET BANK TO 0
          ;START OF BANK LOOP
1$: CALL EXBANK ;EXAMINE BANK
      TST ACFLAG ;CAN WE ACCESS THIS BANK?
      BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
      TST RRFLAG ;RELOCATION REQUIRED?
      BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
      CLR PATTERN ;SET PATTERN TO 0
          ;START OF PATTERN LOOP
2$: CALL MTEST ;GO TEST CORRECT MEMORY
      ;TERMINATION OF PATTERN LOOP
      CALL INCPAT ;GO SEE IF THIS IS THE LAST PATTERN
      BNE 2$ ;NO - LOOP ON THIS PATTERN
          ;TERMINATION OF BANK LOOP
4$: CLR CONTFLAG
      CALL INCBNK ;NEXT HIGHER BANK
      BGE 1$ ;IF NOT DONE - LOOP ON THIS BANK
          ;END OF LOOPS
      TST RLFLAG ;HAVE WE BEEN RELOCATED?
      BEQ 5$ ;NO - SKIP
      RETURN ;YES - RETURN
5$: CALL RELOCATE ;MOVE & MAP PROGRAM
      ON.ERROR THEN $RETURN
      ;**NOTE** RECURSIVE CALL
      CALL BAFPAF ;CALL SELF
      CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
      RETURN
    
```

3313 014756

```

BAFPAR: SUBTST <<BANKS FORWARD,PATTERNS REVERSE      **RECURSIVE**>>
:*****
:*SUBTEST      BANKS FORWARD,PATTERNS REVERSE      **RECURSIVE**
:*****
      CLR      BANK      ;SET BANK TO 0
      ;START OF BANK LOOP
1$:   CALL     EXBANK     ;EXAMINE BANK
      TST      ACFLAG    ;CAN WE ACCESS THIS BANK?
      BEQ      4$        ;NO - GO TO BANK LOOP TERMINATION
      TST      RRFLAG    ;RELOCATION REQUIRED?
      BNE      4$        ;YES - GO TO BANK LOOP TERMINATION
      CALL     SETPAT    ;SET HIGH PATTERN FOR CORRECT MEMORY
      ;START OF PATTERN LOOP
      CALL     MTEST     ;GO TEST CORRECT MEMORY
2$:   ;TERMINATION OF PATTERN LOOP
      DEC     PATTERN    ;IS THIS THE LAST PATTERN?
      BPL     2$        ;NO - LOOP ON THIS PATTERN
      ;TERMINATION OF BANK LOOP
4$:   CLR      CONTFLAG
      CALL     INCBNK    ;NEXT HIGHER BANK
      BGE     1$        ;IF NOT DONE - LOOP ON THIS BANK
      ;END OF LOOPS
      TST     RLFLAG    ;HAVE WE BEEN RELOCATED?
      BEQ     5$        ;NO - SKIP
      RETURN   ;YES - RETURN
5$:   CALL     RELOCATE  ;MOVE & MAP PROGRAM
      ON.ERROR THEN $RETURN
      ;**NOTE** RECURSIVE CALL
      CALL    BAFPAR    ;CALL SELF
      CALL    UNRELOCATE ;UNMOVE & UNMAP PROGRAM
      RETURN

```

```

3314 014756 005037 002100
3315
3316 014762 004737 044240
3317 014766 005737 002114
3318 014772 001412
3319 014774 005737 002122
3320 015000 001007
3321 015002 004737 044742
3322
3323 015006 004737 016150
3324
3325 015012 005337 002110
3326 015016 100373
3327
3328 015020 005037 002214
3329 015024 004737 044752
3330 015030 002354
3331
3332 015032 005737 002124
3333 015036 001401
3334 015040 000207
3335 015042 004737 042466
3336 015046
3337
3338 015052 004737 014756
3339 015056 004737 043356
3340 015062 000207

```

3343 015064

```
BAWPAF: SUBTST <<BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST      BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**
:*****
```

```
3344 015064 005037 002100
3345
3346 015070 004737 044240
3347 015074 005737 002114
3348 015100 001415
3349 015102 005737 002126
3350 015106 001412
3351 015110 005737 002122
3352 015114 001007
3353 015116 005037 002110
3354
3355 015122 004737 016150
3356
3357 015126 004737 044726
3358 015132 001373
3359
3360 015134 005037 002214
3361 015140 004737 044752
3362 015144 002351
3363
3364 015146 005137 002540
3365 015152 001003
3366
3367 015154 004737 015064
3368 015160 000207
3369 015162 005737 002124
3370 015166 001401
3371 015170 000207
3372 015172 004737 042466
3373 015176
3374
3375 015202 004737 015064
3376 015206 004737 043356
3377 015212 000207
```

```

:      CLR      BANK          ;SET BANK TO 0
:      ;START OF BANK LOOP
1$:   CALL     EXBANK         ;EXAMINE BANK
      TST      ACFLAG        ;CAN WE ACCESS THIS BANK?
      BEQ      4$             ;NO - GO TO BANK LOOP TERMINATION
      TST      BMFLAG        ;IS THIS BAD MEMORY (WORST FIRST)?
      BEQ      4$             ;NO - GO TO BANK LOOP TERMINATION
      TST      RRFLAG        ;RELOCATION REQUIRED?
      BNE      4$             ;YES - GO TO BANK LOOP TERMINATION
      CLR      PATTERN       ;SET PATTERN TO 0
:      ;START OF PATTERN LOOP
2$:   CALL     MTEST          ;GO TEST CORRECT MEMORY
      ;TERMINATION OF PATTERN LOOP
      CALL     INCPAT        ;GO SEE IF THIS IS THE LAST PATTERN
      BNE      2$             ;NO - LOOP ON THIS PATTERN
:      ;TERMINATION OF BANK LOOP
4$:   CLR      CONFLAG
      CALL     INCBNK        ;NEXT HIGHER BANK
      BGE      1$             ;IF NOT DONE - LOOP ON THIS BANK
:      ;END OF LOOPS
      COM      WORST         ;IS THIS AN EVEN NUMBERED PASS?
      BNE      5$             ;YES - SKIP
:      ;**NOTE** RECURSIVE CALL
      CALL     BAWPAF        ;CALL SELF
      RETURN
5$:   TST      RLFLAG        ;HAVE WE BEEN RELOCATED?
      BEQ      6$             ;NO - SKIP
      RETURN                ;YES - RETURN
6$:   CALL     RELOCATE       ;MOVE & MAP PROGRAM
      ON.ERROR THEN $RETURN
:      ;**NOTE** RECURSIVE CALL
      CALL     BAWPAF        ;CALL SELF
      CALL     UNRELOCATE    ;UNMOVE & UNMAP PROGRAM
      RETURN
```

```

3380 015214          BAWPAR: SUBTST <<BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**>>
;*****
;*SUBTEST          BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**
;*****
3381 015214 005037 002100      CLR      BANK          ;SET BANK TO 0
3382          ;START OF BANK LOOP
3383 015220 004737 044240      1$:     CALL     EXBANK      ;EXAMINE BANK
3384 015224 005737 002114      TST     ACFLAG        ;CAN WE ACCESS THIS BANK?
3385 015230 001415          BEQ     4$             ;NO - GO TO BANK LOOP TERMINATION
3386 015232 005737 002126      TST     BMFLAG        ;IS THIS BAD MEMORY (WORST FIRST)
3387 015236 001412          BEQ     4$             ;NO - GO TO BANK LOOP TERMINATION
3388 015240 005737 002122      TST     RRFLAG        ;RELOCATION REQUIRED?
3389 015244 001007          BNE     4$             ;YES - GO TO BANK LOOP TERMINATION
3390 015246 004737 044742      CALL    SETPAT        ;SET HIGH PATTERN FOR CORRECT MEMORY
3391          ;START OF PATTERN LOOP
3392 015252 004737 016150      2$:     CALL    MTEST        ;GO TEST CORRECT MEMORY
3393          ;TERMINATION OF PATTERN LOOP
3394 015256 005337 002110      DEC     PATTERN       ;IS THIS THE LAST PATTERN?
3395 015262 100373          BPL     2$             ;NO - LOOP ON THIS PATTERN
3396          ;TERMINATION OF BANK LOOP
3397 015264 005037 002214      4$:     CLR     CONTFLAG   ;NEXT HIGHER BANK
3398 015270 004737 044752      CALL    INCBNK        ;IF NOT DONE - LOOP ON THIS BANK
3399 015274 002351          BGE     1$             ;END OF LOOPS
3400          ;END OF LOOPS
3401 015276 005137 002540      COM     .WORST        ;IS THIS AN EVEN NUMBERED PASS?
3402 015302 001003          BNE     5$             ;YES - SKIP
3403          ;**NOTE** RECURSIVE CALL
3404 015304 004737 015214      CALL    BAWPAR        ;CALL SELF
3405 015310 000207          RETURN
3406 015312 005737 002124      5$:     TST     RLFLAG        ;HAVE WE BEEN RELOCATED?
3407 015316 001401          BEQ     6$             ;NO - SKIP
3408 015320 000207          RETURN                ;YES - RETURN
3409 015322 004737 042466      6$:     CALL    RELOCATE     ;MOVE & MAP PROGRAM
3410 015326          ON.ERROR THEN $RETURN
3411          ;**NOTE** RECURSIVE CALL
3412 015332 004737 015214      CALL    BAWPAR        ;CALL SELF
3413 015336 004737 043356      CALL    UNRELOCATE    ;UNMOVE & UNMAP PROGRAM
3414 015342 000207          RETURN
    
```

3417 015344

```

PAFBAF: SUBTST <<PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**>>
:*****
:*SUBTST PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**
:*****
CLR PATTERN ;SET PATTERN TO 0
;START OF PATTERN LOOP
1$: CLR BANK ;SET BANK TO 0
;START OF BANK LOOP
2$: CALL EXBANK ;EXAMINE BANK
CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF BANK LOOP
4$: CLR CONTFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
;TERMINATION OF PATTERN LOOP
CALL INCRPT ;NEXT HIGHER PATTERN
BNE 1$ ;OK - LOOP; ELSE CONTINUE
;END OF LOOPS
COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
;IS THIS AN EVEN NUMBER PASS?
BEQ 5$ ;YES - SKIP
;***NOTE** RECURSIVE CALL
CALL PAFBAF ;CALL SELF
RETURN
5$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 6$ ;NO - SKIP
RETURN ;YES - RETURN
6$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN
;***NOTE** RECURSIVE CALL
CALL PAFBAF ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETJRN
    
```

3418 015344 005037 002110  
 3419  
 3420 015350 005037 002100  
 3421  
 3422 015354 004737 044240  
 3423 015360 004737 044710  
 3424 015364 001010  
 3425 015366 005737 002114  
 3426 015372 001405  
 3427 015374 005737 002122  
 3428 015400 001002  
 3429 015402 004737 016150  
 3430  
 3431 015406 005037 002214  
 3432 015412 004737 044752  
 3433 015416 002356  
 3434  
 3435 015420 004737 044726  
 3436 015424 001351  
 3437  
 3438 015426 005137 002132  
 3439  
 3440 015432 001403  
 3441  
 3442 015434 004737 015344  
 3443 015440 000207  
 3444 015442 005737 002124  
 3445 015446 001401  
 3446 015450 000207  
 3447 015452 004737 042466  
 3448 015456  
 3449  
 3450 015462 004737 015344  
 3451 015466 004737 043356  
 3452 015472 000207



```

3455 015474
3456 015474 005037 002110
3457
3458 015500 005037 002100
3459
3460 015504 004737 044240
3461 015510 004737 044710
3462 015514 001013
3463 015516 005737 002114
3464 015522 001410
3465 015524 005737 002126
3466 015530 001405
3467 015532 005737 002122
3468 015536 001002
3469 015540 004737 016150
3470
3471 015544 005037 002214
3472 015550 004737 044752
3473 015554 002353
3474
3475 015556 004737 044726
3476 015562 001346
3477
3478 015564 005137 002132
3479
3480 015570 001403
3481
3482 015572 004737 015474
3483 015576 000207
3484 015600 005137 002540
3485 015604 001003
3486
3487 015606 004737 015474
3488 015612 000207
3489 015614 005737 002124
3490 015620 001401
3491 015622 000207
3492 015624 004737 042466
3493 015630
3494
3495 015634 004737 015474
3496 015640 004737 043356
3497 015644 000207

```

```

PAFBAW: SUBST <<PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**>>
:*****
:*SUBTEST PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**
:*****
          CLR PATTERN ;SET PATTERN TO 0
          ;START OF PATTERN LOOP
1$: CLR BANK ;SET BANK TO 0
          ;START OF BANK LOOP
2$: CALL EXBANK ;EXAMINE BANK
          CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
          BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
          TST ACFLAG ;CAN WE ACCESS THIS BANK?
          BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
          TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)
          BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
          TST RRFLAG ;RELOCATION REQUIRED?
          BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
          CALL MTEST ;GO TEST CORRECT MEMORY
          ;TERMINATION OF BANK LOOP
4$: CLR CONFLAG
          CALL INCBNK ;NEXT HIGHER BANK
          BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
          ;TERMINATION OF PATTERN LOOP
          CALL INCRPT ;NEXT HIGHER PATTERN
          BNE 1$ ;OK - LOOP; ELSE CONTINUE
          ;END OF LOOPS
          COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
          ;IS THIS AN EVEN NUMBER PASS?
          BEQ 5$ ;YES - SKIP
          ;**NOTE** RECURSIVE CALL
          CALL PAFBAW ;CALL SELF
          RETURN
5$: COM WORST ;4TH PASS?
          BNE 6$ ;YES - SKIP
          ;**NOTE** RECURSIVE CALL
          CALL PAFBAW ;CALL SELF
          RETURN
6$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
          BEQ 7$ ;NO - SKIP
          RETURN ;YES - RETURN
7$: CALL RELOCATE ;MOVE & MAP PROGRAM
          ON.ERROR THEN $RETURN
          ;**NOTE** RECURSIVE CALL
          CALL PAFBAW ;CALL SELF
          CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
          RETURN

```

3500 015646

```

PARBAF: SUBTST <<PATTERNS REVERSE,BANKS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST PATTERNS REVERSE,BANKS FORWARD **RECURSIVE**
:*****
3501 015646 004737 044742 CALL HIPAT ;SET HIGHEST PATTERNS
3502 ;START OF PATTERN LOOP
3503 015652 005037 002100 1$: CLR BANK ;SET BANK TO 0
3504 ;START OF BANK LOOP
3505 015656 004737 044240 2$: CALL EXBANK ;EXAMINE BANK
3506 015662 004737 044710 CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
3507 015666 001010 BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
3508 015670 005737 002114 TST ACFLAG ;CAN WE ACCESS THIS BANK?
3509 015674 001405 BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
3510 015676 005737 002122 TST RRFLAG ;RELOCATION REQUIRED?
3511 015702 001002 BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
3512 015704 004737 016150 CALL MTEST ;GO TEST CORRECT MEMORY
3513 ;TERMINATION OF BANK LOOP
3514 015710 005037 002214 4$: CLR CONFLAG
3515 015714 004737 044752 CALL INCBNK ;NEXT HIGHER BANK
3516 015720 002356 BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
3517 ;TERMINATION OF PATTERN LOOP
3518 015722 005337 002110 DEC PATTERN ;NEXT LOWER PATTERN
3519 015726 100351 BPL 1$ ;OK - LOOP; ELSE CONTINUE
3520 ;END OF LOOPS
3521 015730 005137 002132 COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
3522 ;IS THIS AN EVEN NUMBER PAGES?
3523 015734 001403 BEQ 5$ ;YES - SKIP
3524 ;**NOTE** RECURSIVE CALL
3525 015736 004737 015646 CALL PARBAF ;CALL SELF
3526 015742 000207 RETURN
3527 015744 005737 002124 5$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
3528 015750 001401 BEQ 6$ ;NO - SKIP
3529 015752 000207 RETURN ;YES - RETURN
3530 015754 004737 042466 6$: CALL RELOCATE ;MOVE & MAP PROGRAM
3531 015760 ON.ERROR THEN $RETUPN
3532 ;**NOTE** RECURSIVE CALL
3533 015764 004737 015646 CALL PARBAF ;CALL SELF
3534 015770 004737 043356 CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
3535 015774 000207 RETURN
    
```

3538 015776  
 3539 015776 004737 044742  
 3540  
 3541 016002 005037 002100  
 3542  
 3543 016006 004737 044240  
 3544 016012 004737 044710  
 3545 016016 001013  
 3546 016020 005737 002114  
 3547 016024 001410  
 3548 016026 005737 002126  
 3549 016032 001405  
 3550 016034 005737 002122  
 3551 016040 001002  
 3552 016042 004737 016150  
 3553  
 3554 016046 005037 002214  
 3555 016052 004737 044752  
 3556 016056 002353  
 3557  
 3558 016060 005337 002110  
 3559 016064 100346  
 3560  
 3561 016066 005137 002132  
 3562  
 3563 016072 001403  
 3564  
 3565 016074 004737 015776  
 3566 016100 000207  
 3567 016102 005137 002540  
 3568 016106 001003  
 3569  
 3570 016110 004737 015776  
 3571 016114 000207  
 3572 016116 005737 002124  
 3573 016122 001401  
 3574 016124 000207  
 3575 016126 004737 042466  
 3576 016132  
 3577  
 3578 016136 004737 015776  
 3579 016142 004737 043356  
 3580 016146 000207

```

PARBAW: SUBTST <<PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**>>
;*****
;*SUBTEST PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**
;*****
CALL HIPAT ;SET HIGHEST PATTERN
;START OF PATTERN LOOP
1$: CLR BANK ;SET BANK TO 0
;START OF BANK LOOP
2$: CALL EXBANK ;EXAMINE BANK
CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF BANK LOOP
4$: CLR CONFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
;TERMINATION OF PATTERN LOOP
DEC PATTERN ;NEXT LOWER PATTERN
BPL 1$ ;OK - LOOP; ELSE CONTINUE
;END OF LOOPS
COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
BEQ 5$ ;IS THIS AN EVEN NUMBER PASS?
;YES - SKIP
; **NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
RETURN
5$: COM WORST ;4TH PASS?
BNE 6$ ;YES - SKIP
; **NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
RETURN
6$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 7$ ;NO - SKIP
RETURN ;YES - RETURN
7$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN
; **NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN
  
```

3583 016150

```

MTEST: SUBTST <<SUBR SETUP MEMORY TEST>>
;*****
;*SUBTEST SUBR SETUP MEMORY TEST
;*****
SET HEADER ;INITIALIZE HEADER MESSAGE TYPEOUT
SET MUT ;INDICATE THERE IS A MEMORY UNDER TEST
CLR PASFLG
TST MKFLAG ;ECC?
BEQ MT1 ;NO - SKIP
BEGIN HOLDLOOP
IF CONTFLAG IS TRUE THEN LEAVE HOLDLOOP
IF SKIPMK IS FALSE
CALL MKCONTROL
END; OF IF SKIPMK
END HOLDLOOP
CALL MKTEST ;YES - DO ECC TESTS
BR MT2
MT1: CALL MJTEST ;DO PARITY TESTS
MT2: CLR MUT ;NOW - NO MEMORY UNDER TEST
SET HEADER ;ALLOW HEADERS NORMAL
RETURN
    
```

3584 016150  
 3585 016156  
 3586 016164 005037 002256  
 3587 016170 005737 002116  
 3588 016174 001413  
 3589 016176  
 3590 016176  
 3591 016204  
 3592 016212 004737 016244  
 3593 016216  
 3594 016216  
 3595 016216 004737 017456  
 3596 016222 000402  
 3597 016224 004737 017676  
 3598 016230 005037 002106  
 3599 016234  
 3600 016242 000207

3603 016244

```

MKCONTROL:SUBTST      <<SUBR TEST ECC CSR LOGIC DISPATCH>>
:*****
:*SUBTEST          SUBR TEST ECC CSR LOGIC DISPATCH
:*****
:THE NEXT TWO MODULES SOLVE THE PROBLEM OF
:HOW TO RUN THE CSR TESTS ON EACH ECC MEMORY
:
:IF SELONLY IS TRUE THEN $RETURN
:IF INHECC IS TRUE THEN $RETURN
PUSH BANK,R0,R1,R2,R3
MOV #FIRST,CSRFBANK ;SET FIRST TEST ADDRESS TO FIRST ADDR.
MOV #LAST,CSRLBANK
CLR CSRINT
CLR SPLTCSR
CLR CSRLOOP ; AND ZERO THE LOOP COUNTER
MOV BANKINDEX,R0 ;GET THE BANK INDEX
MOV CONFIG(R0),R1 ;GET CSR NUMBER
SWAB R1
BIC #^C17,R1
ASL R1
MOV R1,CSRHOLD ;STORE IN THE LOW BYTE
TST INTFLAG ;IS THIS BANK INTERLEAVED?
BEQ 1$ ;BRANCH IF NOT INTERLEAVED
INC SPLTCSR
MOV #120000,CSRLBANK ;WE MUST LOOP TWICE FOR AN INTERLEAVED BANK
INC CSRLOOP
INC CSRINT ;GET THE INTERLEAVE CSR NUMBER
MOV CONFIG(R0),R1
ASH #-3,R1
BIC #^C17000,R1
BIS R1,CSRHOLD ;STORE IT IN CSRHOLD'S UPPER BYTE
1$: CLR R3
MKLOOP: MOVB CSRHOLD(R3),CSRNO ;CLEAR ANY UNNECESSARY BITS
BIC #^C36,CSRNO
FOR R2 := #0 TO CSRINT
FOR CSRFIRST := CSRFBANK TO CSRLBANK BY #4000
MAP BANK ;MAP TEST SPACE TO BANK
INVALIDATE ;INVALIDATE BACKGROUND PATTERN
BEGIN CSRSTUFF
CLR SUCCESS
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
MOV CSRFIRST,CSRLAST
ADD #4000,CSRLAST
FOR TESTADD := CSRFIRST TO CSRLAST BY #4
MOV TESTADD,TESTADD+2
TST SPLTCSR
BEQ 1$
ADD #4000,TESTADD+2
BR 2$
1$: ADD #2,TESTADD+2
2$: CALL SBTEST
ON.NOERROR
CACHOFF ;TURN CACHE OFF
CLR NOPAR ;INDICATE PARITY ACTION
FOR I := #0 TO #27
SET HEADER
CLR PASFLG

```

3604  
3605  
3606  
3607 016244  
3608 016254  
3609 016264  
3610 016300 012737 060000 002224  
3611 016306 012737 157776 002226  
3612 016314 005037 002230  
3613 016320 005037 002232  
3614 016324 005037 002302  
3615 016330 013700 002102  
3616 016334 016001 002624  
3617 016340 000301  
3618 016342 042701 177760  
3619 016346 006301  
3620 016350 010137 002476  
3621 016354 005737 002134  
3622 016360 001421  
3623 016362 005237 002232  
3624 016366 012737 120000 002226  
3625 016374 005237 002302  
3626 016400 005237 002230  
3627 016404 016001 002624  
3628 016410 072127 177775  
3629 016414 042701 160777  
3630 016420 050137 002476  
3631 016424 005003  
3632 016426 116337 002476 002146  
3633 016434 042737 177741 002146  
3634 016442  
3635 016444  
3636 016452  
3637 016466 104511  
3638 016470  
3639 016470 005037 002304  
3640 016474  
3641 016510 013737 002220 002222  
3642 016516 062737 004000 002222  
3643 016524  
3644 016532 013737 002362 002364  
3645 016540 005737 002232  
3646 016544 001404  
3647 016546 062737 040000 002364  
3648 016554 000403  
3649 016556 062737 000002 002364  
3650 016564 004737 017052  
3651 016570  
3652 016572 104424  
3653 016574 005037 002074  
3654 016600  
3655 016604  
3656 016612 005037 002256

3657	016616				LET R0 := I	
3658	016622				PUSH R3	;SAVE LOOP COUNTER
3659	016624	010637	002142		MOV SP,CTLKVEC	;SAVE VECTOR IN CSR OF ^K
3660	016630	162737	000002	002142	SUB #2,CTLKVEC	
3661	016636	004737	017356		CALL CSRCASE	
3662	016642				POP R3	;RESTORE LOOP COUNTER
3663	016644				END ;OF FOR I	
3664	016660	104423			CACHON	;TURN CACHE ON
3665	016662				SET SUCCESS	
3666	016670				LEAVE CSRSTUFF	
3667	016672				END ;OF ON.NOERROR	
3668	016672				END ;OF FOR TESTADD	
3669	016710				END ;OF IF	
3670	016710				END CSRSTUFF	
3671	016710				IF SUCCESS IS FALSE	
3672	016716				TYPE MSGA34	
3673	016722				TYPOCS BANK,<TYPES BANK NUMBER>,3	
3674	016732				TYPE MSGB34	
3675	016736				END ;OF IF SUCCESS	
3676	016736				END; OF FOR CSRFIRST	
3677	016754	005237	002232		INC SPLTCSR	
3678	016760				END; OF FOR R2	
3679	016770	062737	000002	002224	ADD #2,CSRFBANK	
3680	016776	012737	000001	002232	MOV #1,SPLTCSR	
3681	017004	005203			INC R3	
3682	017006	020337	002302		CMP R3,CSRLOOP	
3683	017012	003002			BGT 1\$	
3684	017014	000137	016426		JMP MKLOOP	
3685	017020	104472		1\$:	ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)	
3686	017022				SET CONTFLAG	
3687	017030	005037	002232		CLR SPLTCSR	
3688	017034				POP R3,R2,R1,RU,BANK	
3689	017050	000207			RETURN	

3692 017052

```

SBTEST:SUBTST <<CHECK FOR SBE FREE LOCATIONS>>
:*****
:*SUBTEST      CHECK FOR SBE FREE LOCATIONS
:*****
:IN ORDER TO DETERMINE IF A LOCATION IS SBE FREE I DO THIS
:
:WRITE ZEROS WITH ECC DISABLE
:READ ZEROS BACK
:IF NOT ZEROS THEN RETURN ERROR
:
:WRITE ZEROS WITH ECC ENABLED BUT TRAPS DISABLED
:READ ZEROS BACK
:IF NOT ZEROS THEN RETURN ERROR
:
:TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
:IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
:
:COMPLIMENT ZEROS TO ONES WITH ECC DISABLE
:READ ONES BACK
:IF NOT ONES THEN RETURN ERROR
:
:WRITE 100,,100000,00000 (CHECKBITS COMPLIMENT OF BEFORE)
: WITH ECC ENABLED BUT TRAPS DISABLED
:TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
:IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
:
:IF NONE OF THE ABOVE FORCES A RETURN ERROR THEN RETURN NO.ERROR
.ENABL  LSB
PUSH   RO,R1,R4           ;PUSH RO,R1,R4 ONTO STACK
MOV    TESTADD,R1
MOV    TESTADD+2,R4
TESTAREA           ;ENTER TEST MODE
CACHOFF           ;TURN CACHE OFF
ECC1DIS           ;DISABLE ECC ON 1 SELECTED CSR
CLEAR   (R1),(R4)
TST    (R1)
BNE    SBENT
TST    (R4)
BNE    SBENT
CLP1CSR           ;CLEAR 1 SELECTED CSR
CLEAR   (R1),(R4)
TST    (R1)
BNE    SBENT
TST    (R4)
BNE    SBENT
TSTREAD           ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
IF #BIT15!BIT4 SET.IN CSR
SET SKPERR           ;DISABLE ERRGEN'S ERROR PRINTOUT
ERRGEN
MOV    ERRADD,RO
ASH   #-4,RO
BIC   #^C177,RO
IF BANK EQ RO THEN GOTO SBENT
END: OF IF #BIT15
ECC1DIS           ;DISABLE ECC ON 1 SELECTED CSR
  
```

3693  
 3694  
 3695  
 3696  
 3697  
 3698  
 3699  
 3700  
 3701  
 3702  
 3703  
 3704  
 3705  
 3706  
 3707  
 3708  
 3709  
 3710  
 3711  
 3712  
 3713  
 3714  
 3715  
 3716  
 3717 017052  
 3718 017060 013701 002362  
 3719 017064 013704 002364  
 3720 017070  
 3721 017076 104424  
 3722 017100 104471  
 3723 017102  
 3724 017106 005711  
 3725 017110 001107  
 3726 017112 005714  
 3727 017114 001105  
 3728  
 3729 017116 104503  
 3730 017120  
 3731 017124 005711  
 3732 017126 001100  
 3733 017130 005714  
 3734 017132 001076  
 3735  
 3736 017134 104510  
 3737 017136  
 3738 017146  
 3739 017154 104512  
 3740 017156 013700 002430  
 3741 017162 072027 177774  
 3742 017166 042700 177600  
 3743 017172  
 3744 017200  
 3745 017200 104471

CZMSDBO MS11-L/M DIAGNOSTIC  
CHECK FOR SBE FREE LOCATIONS

MACRO M1113 07-OCT-80 18:01 PAGE 127-1 SEQUENCE 176

SEQ 0183

```

3746 017202 005111      COM      (R1)
3747 017204 005114      COM      (R4)
3748 017206 023711 002554  CMP      ONES,(R1)
3749 017212 001046      BNE      SBFNT
3750 017214 023714 002554  CMP      ONES,(R4)
3751 017220 001043      BNE      SBENT
3752
3753 017222 104503      CLR1CSR      ;CLEAR 1 SELECTED CSR
3754 017224 005011      CLR      (R1)
3755 017226 012714 100000  MOV      #BIT15,(R4)
3756 017232 005711      TST      (R1)
3757 017234 001035      BNE      SBENT
3758 017236 022714 100000  CMP      #BIT15,(R4)
3759 017242 001032      BNE      SBENT
3760
3761 017244 104510      TSTREAD      ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
3762 017246      IF #BIT15.BIT4 SET.IN CSR
3763 017256      SET SKPERR      ;DISABLE ERRGEN'S ERROR PRINTOUT
3764 017264 104512      ERRGEN
3765 017266 013700 002430  MOV      ERRADD,R0
3766 017272 072027 177774  ASH      #-4,R0
3767 017276 042700 177600  BIC      #^C177,R0
3768 017302      IF BANK EQ R0 THEN GOTO SBENT
3769 017310      END; OF IF #BIT15
3770
3771 017310 104417      KERNEL      ;ENTER KERNEL MODE
3772 017312 104473      ECC1INIT    ;INITIALIZE 1 SELECTED CSR
3773 017314 104423      CACHON      ;TURN CACHE ON
3774 017316      POP      R4,R1,R0 ;POP R0,R1 & R4 FROM STACK
3775 017324      $RETURN NOERROR
3776
3777 017330 104503      SBENT: CLR1CSR      ;CLEAR 1 SELECTED CSR
3778 017332      CLEAR      (R1),(R4)
3779 017336 104417      KERNEL      ;ENTER KERNEL MODE
3780 017340 104473      ECC1INIT    ;INITIALIZE 1 SELECTED CSR
3781 017342 104423      CACHON      ;TURN CACHE ON
3782 017344      POP      R4,R1,R0 ;POP R0,R1 & R4 FROM STACK
3783 017352      $RETURN ERROR
3784      .DSABL  LSB

```



3787 017356

```
CSRCASE:SUBTST <<CSR PATTERN CASE STATEMENT>>
:*****
:*SUBTEST      CSR PATTERN CASE STATEMENT
:*****
```

3788 017356

3789  
3790  
3791  
3792

```
CASE R0
:WARNING IF YOU CHANGE THIS TABLE ALSO
:CHANGE "SDDW0" - "SDDW5" (THE PATTERN BIT MAP)
```

3793 017366 021132  
3794 017370 021230  
3795 017372 023622  
3796 017374 021264  
3797 017376 021332  
3798 017400 021426  
3799 017402 021502  
3800 017404 021560  
3801 017406 021626  
3802 017410 026354  
3803 017412 026354  
3804 017414 026354  
3805 017416 026354  
3806 017420 026354  
3807 017422 026354  
3808 017424 026354  
3809 017426 026354  
3810 017430 026354  
3811 017432 026354  
3812 017434 026354  
3813 017436 026354  
3814 017440 026354  
3815 017442 026354  
3816 017444 026354  
3817 017446  
3818 017454 000207

```
          :PAT  TIME      DISCRPTION
MKCSRT:  MT0006       :<1 SEC  INITIAL DATA TEST
          MT0010       :<1 SEC  BYTE ADDRESSING TEST
          MT0025       :<1 SEC  INTERRUPT ENABLE 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 BYTF WITH DBE
          MT0016       :<1 SEC  WRITE INHIBIT OF WORD WITH DBE
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          MT0999       : U SEC  NULL TEST
          MT0999       : 0 SEC  NULL TEST
          END ;OF CASE R0
          RETURN
```

3821 017456

```

MKTEST: SUBTST <<SUBR ECC TEST DISPATCH>>
;*****
;*SUBTEST SUBR ECC TEST DISPATCH
;*****
IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
ECCDIS ;DISABLE ERROR CORRECTION
ELSE
CLRCSR ;CLEAR ALL CSR'S
END ;OF IF
MOV #2,NOPAR ;INDICATE PARITY ACTION
MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
MOV PATTERN,RO ;GET PATTERN NUMBER
ASL RO ;MAKE IT A WORD ADDRESS
IF MKPAT(RO) NE #MT0034 AND MKPAT(RO) NE #MT0999
INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
END ;OF IF MKPAT(RO)
MOV SP,CTLKVEC ;SAVE VECTOR IN CASE OF ^K
SUB #2,CTLKVEC
CALL @MKPAT(RO) ;INDEX OFF TABLE
IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
ENASBE ;TRAP ON SINGLE BIT ERRORS
ELSE
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
END ;OF IF #SWO
CLR NOPAR ;INDICATE PARITY ACTION
RETURN

```

3825 017456  
3826 017474 104470  
3827 017476  
3828 017500 104502  
3829 017502  
3830 017502 012737 000002 002074  
3831 017510 012737 000002 002276  
3832 017516 013700 002110  
3833 017522 006300  
3834 017524  
3835 017544 104511  
3836 017546  
3837 017546 010637 002142  
3838 017552 162737 000002 002142  
3839 017560 004770 017616  
3840 017564  
3841 017602 104506  
3842 017604  
3843 017606 104472  
3844 017610  
3845 017610 005037 002074  
3846 017614 000207  
3847  
3848  
3849  
3850

;WARNING IF YOU CHANGE THIS TABLE ALSO  
;CHANGE '\$DDW0' - '\$DDW5' (THE PATTERN BIT MAP)

```

;PAT TIME DISRIPTION
MKPAT: ;NOTE MT0034 MUST BE FIRST & LAST
MT0034 ;<1 SEC ;SOFT ERROR - BACKGROUND PATTERN TEST
MT0017 ;<1 SEC ;HOLDING 1'S & 0'S TEST
MT0007 ;<1 SEC ;ADDRESS BIT TEST
MT0001 ;<1 SEC ;ADDRESS TEST
MT0002 ;<1 SEC ;COMPLEMENT ADDRESS TEST
MT0004 ; 1 SEC ;ROTATING ZEROS TEST
MT0005 ; 1 SEC ;ROTATING ONES TEST
MT0021 ; 1 SEC ;MARCHING 0'S & 1'S TEST
MT0020 ;<1 SEC ;MARCHING 1'S & 0'S IN CHECK BITS
MT0022 ;10 SEC ;REFRESH & SHIFTING DIAGONAL TEST
MT0026 ;<1 SEC ;RANDOM DATA TEST
MT0024 ;20 SEC ;FAST GALLOPING PATTERN TEST
MT0031 ; 3 SEC ;SOB-A-LONG TEST
MT0032 ;<1 SEC ;WRITE RECOVERY TEST
M 0033 ;35 SEC ;BRANCH GOBBLE TEST
MT0034 ;<1 SEC ;SOFT ERROR - BACKGROUND PATTERN TEST
;NOTE MT0034 MUST BE FIRST & LAST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST

```

3851 017616  
3852 017616 026070  
3853 017620 021674  
3854 017622 021166  
3855 017624 020142  
3856 017626 020262  
3857 017630 020654  
3858 017632 020776  
3859 017634 023006  
3860 017636 021716  
3861 017640 023260  
3862 017642 023670  
3863 017644 023356  
3864 017646 025160  
3865 017650 025350  
3866 017652 025702  
3867 017654 026070  
3868  
3869 017656 026354  
3870 017660 026354  
3871 017662 026354  
3872 017664 026354  
3873 017666 026354  
3874 017670 026354  
3875 017672 026354  
3876 017674 026354

3879 017676

MJTEST: SUBTST <<SUBR PARITY TEST DISPATCH>>

\*\*\*\*\*  
 :+SUBTEST SUBR PARITY TEST DISPATCH  
 \*\*\*\*\*

3883 017676 012737 000002 002074  
 3884 017704 012737 000002 002276  
 3885 017712 012737 060000 002362  
 3886 017720 012737 060002 002364  
 3887 017726 013700 002110  
 3888 017732 006300  
 3889 017734  
 3890 017754 104511  
 3891 017756  
 3892 017756 010637 002142  
 3893 017762 162737 000002 002142  
 3894 017770 004770 020002  
 3895 017774 005037 002074  
 3896 020000 000207

```

MOV #2,NOPAR ;INDICATE PARITY ACTION
MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
MOV #FIRST,TESTADD
MOV #FIRST+2,TESTADD+2
MOV PATTERN,RO ;GET PATTERN NUMBER
ASL RO ;MAKE IT A WORD ADDRESS
IF MJPAT(RO) NE #MT0034 AND MJPAT(RO) NE #MT0999
  INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
END ;OF IF MJPAT(RO)
MOV SP,CTLKVEC ;SAVE VECTOR IN CASE OF ^K
SUB #2,CTLKVEC
CALL @MJPAT(RO) ;INDEX OFF TABLE
CLR NOPAR ;INDICATE PARITY ACTION
RETURN
  
```

;WARNING IF YOU CHANGE THIS TABLE ALSO  
 ;CHANGE '\$DDW0' - '\$DDW5' (THE PATTERN BIT MAP)

3897  
 3898  
 3899  
 3900  
 3901  
 3902 020002  
 3903 020002 026070  
 3904 020004 021132  
 3905 020006 021674  
 3906 020010 021166  
 3907 020012 020142  
 3908 020014 020262  
 3909 020016 020422  
 3910 020020 020654  
 3911 020022 020776  
 3912 020024 023006  
 3913 020026 026242  
 3914 020030 023260  
 3915 020032 023312  
 3916 020034 023670  
 3917 020036 023356  
 3918 020040 025160  
 3919 020042 025350  
 3920 020044 025702  
 3921 020046 026070  
 3922  
 3923 020050 026354  
 3924 020052 026354  
 3925 020054 026354  
 3926 020056 026354  
 3927 020060 026354

```

.PAT TIME DISCIPTION
MJPAT: ;NOTE MT0034 MUST BE FIRST & LAST
MT0034 ;<1 SEC ;SOFT ERROR - BACKGROUND PATTERN TEST
MT0006 ;<1 SEC ;INITIAL DATA TEST
MT0017 ;<1 SEC ;HOLDING 1'S & 0'S TEST
MT0007 ;<1 SEC ;ADDRESS BIT 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
MT0021 ; 1 SEC ;MARCHING 0'S & 1'S TEST
MT0035 ;<1 SEC ;WORSE CASE NOISE PARITY TEST
MT0022 ;10 SEC ;REFRESH TEST
MT0023 ;10 SEC ;SHIFTING DIAGONAL TEST
MT0026 ;<1 SEC ;RANDOM DATA TEST
MT0024 ;20 SEC ;FAST GALLOPING PATTERN 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 - BACKGROUND PATTERN TEST
;NOTE MT0034 MUST BE FIRST & LAST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
MT0999 ; 0 SEC ;NULL TEST
  
```

3929  
3930  
3931  
3932 020062

.SBTTL PATTERNS

.SBTTL MEMORY TEST SETUP ROUTINES  
MT0000: SUBTST <<MT0000 SETUP DATA PATTERN TEST>>

\*\*\*\*\*  
: \*SUBTEST MT0000 SETUP DATA PATTERN TEST  
\*\*\*\*\*

3933 020062 005037 002260  
3934 020066 012700 060000  
3935 020072 012701 040000  
3936 020076 004737 036330  
3937 020102 022737 000001 003710  
3938 020110 001406  
3939 020112 012737 026774 002254  
3940 020120 004737 026692  
3941 020124 000207  
3942 020126  
3943 020134 004737 026424  
3944 020140 000207  
3945 020142

CLR REALPAT ; SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY  
MOV #FIRST,R0  
MOV #SIZE,R1  
CALL REGCOPY  
CMP #1,PROTYP ; ARE WE ON AN 11/44?  
BEQ 1\$ ; BRANCH IF YES  
MOV #MTP000,SUPDOADD ; ELSE DO PATTERN IN MAIN MEMORY  
CALL SUPDO3  
RETURN  
1\$: BMOV MTP000  
CALL SUPDO1 ; DO IT IN SUPERVISOR MODE  
RETURN

MT0001: SUBTST <<MT0001 SETUP ADDRESS TEST>>

\*\*\*\*\*  
: \*SUBTEST MT0001 SETUP ADDRESS TEST  
\*\*\*\*\*

3946 020142 012737 000001 002260  
3947 020150 012700 060000  
3948 020154 012701 040000  
3949 020160 005737 002426  
3950 020164 001005  
3951 020166 023737 172252 172254  
3952 020174 001007  
3953 020176 000404  
3954 020200 023737 177652 177654 2\$:  
3955 020206 001002  
3956 020210 012701 030000 3\$:  
3957 020214 005002 4\$:  
3958 020216 004737 036330  
3959 020222 022737 000001 003710  
3960 020230 001406  
3961 020232 012737 027020 002254  
3962 020240 004737 026602  
3963 020244 000207  
3964 020246  
3965 020254 004737 026424  
3966 020260 000207  
3967 020262

MOV #1,REALPAT ; SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY  
MOV #FIRST,R0  
MOV #SIZE,R1  
TST NOSUPER  
BNE 2\$  
CMP SIPAR5,SIPAR6  
BNE 4\$  
BR 3\$  
2\$: CMP UIPAR5,IIPAR6  
BNE 4\$  
3\$: MOV #30000,R1  
4\$: CLR R2  
CALL REGCOPY  
CMP #1,PROTYP ; IS THIS AN 11/44?  
BEQ 1\$ ; BRANCH IF IT IS  
MOV #MTP001,SUPDOADD ; SET UP CALLING ADDRESS  
CALL SUPDO3  
RETURN  
1\$: BMOV MTP001  
CALL SUPDO1 ; DO IT IN SUPERVISOR MODE  
RETURN

MT0002: SUBTST <<MT0002 SETUP COMPLEMENT ADDRESS TEST>>

\*\*\*\*\*  
: \*SUBTEST MT0002 SETUP COMPLEMENT ADDRESS TEST  
\*\*\*\*\*

3968 020262 012737 000002 002260  
3969 020270 012700 160000  
3970 020274 012701 040000  
3971 020300 012704 060000  
3972 020304 012705 100001  
3973 020310 005737 002426  
3974 020314 001005  
3975 020316 023737 172252 172254  
3976 020324 001013

MOV #2,REALPAT ; SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY  
MOV #LAST+2,R0  
MOV #SIZE,R1  
MOV #FIRST,R4  
MOV #100001,R5  
TST NOSUPER  
BNE 2\$  
CMP SIPAR5,SIPAR6  
BNE 4\$

3977	020326	000404				BR	3\$	
3978	020330	023737	177652	177654	2\$:	CMP	UIPAR5,UIPAR6	
3979	020336	001006				BNE	4\$	
3980	020340	012701	030000		3\$:	MOV	#30000,R1	
3981	020344	012700	140000			MOV	#140000,R0	
3982	020350	012705	120001			MOV	#120001,R5	
3983	020354	012702	000001		4\$:	MOV	#1,R2	
3984	020360	010103				MOV	R1,R3	
3985	020362	022737	000001	003710		CMP	#1,PROTYP	:IS THIS AN 11/44?
3986	020370	001406				BEQ	1\$	:BRANCH IF TRUE
3987	020372	012737	027052	002254		MOV	#MTP002,SUPDOADD	:SET UP CALLING ADDRESS
3988	020400	004737	026602			CALL	SUPDO3	
3989	020404	000207				RETURN		
3990	020406				1\$:	BMOV	MTP002	
3991	020414	004737	026424			CALL	SUPDO1	
3992	020420	000207				RETURN		

3995 020422

```

MT0003: SUBTST <<MT0003          SETUP 3 XOR 9 WORST CASE NOISE TEST>>
:*****
:*SUBTEST          MT0003  SETUP 3 XOR 9 WORST CASE NOISE TEST
:*****
          IF EUFLAG IS TRUE THEN $RETURN
3996 020422          MOV          #3,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3997 020432 012737 000003 002260          CLR          PCBUMP          ;TRAPS DO NOT ADD TO PC
3998 020440 005037 002276          1$: CALL          FLIPWARN          ;SETUP WARNING CONSTANTS & R2
3999 020444 004737 036340          2$: MOV          #FIRST,R1          ;R1 <-- STARTING ADDRESS
4000 020450 012701 060000          MOV          #20000,R3
4001 020454 012703 020000          ASH          #-8,R3          ;R3 <-- R3 / 256.
4002 020460 072327 177770          MOV          #4,R2          ;SMALL LOOP SIZE
4003 020464 012702 000004          MOV          #64,R5          ;MEDIUM LOOP SIZE
4004 020470 012705 000100          CMP          #1,PROTYP          ;IS THIS AN 11/44?
4005 020474 022737 000001 003710          BEQ          3$          ;BRANCH IF IT IS
4006 020502 001415          SAVREG
4007 020504 104415          MOV          #MTPA03,SUPDOADD
4008 020506 012737 027104 002254          CALL          SUPD03          ;DO IT IN MAIN MEMORY
4009 020514 004737 026602          RESREG
4010 020520 104416          MOV          #MTPB03,SUPDOADD
4011 020522 012737 027144 002254          CALL          SUPD04
4012 020530 004737 026616          BR          4$
4013 020534 000442          3$: BMOV          MTPA03
4014 020536          SAVREG
4015 020544 104415          CALL          SUPD01
4016 020546 004737 026424          BMOV          MTPB03
4017 020552          BMOV          MTPC03,KDPAR0,8.
4018 020560          BMOV          MTPD03,SDPAR0,8.
4019 020572          MOV          #KDPAR0,UIPAR1          ;SET UP PAR LINKS
4020 020604 012737 172360 177642          MOV          #SDPAR0,KDPAR6
4021 020612 012737 172260 172374          MOV          #UIPAR2,SDPAR7
4022 020620 012737 177644 172276          MOV          #1032,SDPAR5          ;CHANGE INST TO BR .+66 (BR TO KDPAR1)
4023 020626 012737 001032 172272          RESREG
4024 020634 104416          CALL          SUPD02
4025 020636 004737 026440          4$: CMP          #3,FLIPLOC          ;DONE WITH 4 PATTERNS
4026 020642 022737 000003 002550          BNE          1$          ;[(0,177777);(177777,0);(401,177777);(177777,401)]?
4027          RETURN          ;NO - LOOP
4028 020650 001275
4029 020652 000207
4030
4031 020654
    
```

3996 020422

3997 020432

3998 020440

3999 020444

4000 020450

4001 020454

4002 020460

4003 020464

4004 020470

4005 020474

4006 020502

4007 020504

4008 020506

4009 020514

4010 020520

4011 020522

4012 020530

4013 020534

4014 020536

4015 020544

4016 020546

4017 020552

4018 020560

4019 020572

4020 020604

4021 020612

4022 020620

4023 020626

4024 020634

4025 020636

4026 020642

4027

4028 020650

4029 020652

4030

4031 020654

```

MT0004: SUBTST <<MT0004          SETUP ROTATING ZEROS TEST>>
:*****
:*SUBTEST          MT0004  SETUP ROTATING ZERUS TEST
:*****
          MOV          #4,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4032 020654 012737 000004 002260          MOV          #4,PCBUMP          ;TRAPS ADD 4 TO PC
4033 020662 012737 000004 002276          MOV          ONES,R2
4034 020670 013702 002554          CALL          BACKGND          ;WRITE BACKGROUND OF ONES
4035 020674 004737 036470          MOV          #FIRST,R0
4036 020700 012700 060000          MOV          #SIZE,R1
4037 020704 012701 040000          CMP          #1,PROTYP          ;IS THIS AN 11/44?
4038 020710 022737 000001 003710          BEQ          1$          ;BRANCH IF IT IS
4039 020716 001406          MOV          #MTPA04,SUPDOADD          ;SET UP LINKS
4040 020720 012737 027242 002254          CALL          SUPD04
4041 020726 004737 026616          RETURN
4042 020732 000207          1$: BMOV          MTPA04
4043 020734          BMOV          MTPB04,KDPAR0,8.
4044 020742          MOV          #KDPAR0,UIPAR5
4045 020754 012737 172360 177652
    
```

4032 020654

4033 020662

4034 020670

4035 020674

4036 020700

4037 020704

4038 020710

4039 020716

4040 020720

4041 020726

4042 020732

4043 020734

4044 020742

4045 020754

4046 020762 012737 177654 172376  
 4047 020770 004737 026440  
 4048 020774 000207  
 4049 020776

```

MOV    #UIPAR6,KDPAR7
CALL   SUPD02
RETURN
MT0005: SUBTST <<MT0005      SETUP ROTATING ONES TEST>>
:*****
:SUBTEST      MT0005  SETUP ROTATING ONES TEST
:*****
MOV    #5,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV    #4,PCBUMP       ;TRAPS ADD 4 TO PC
CLR    R2
CALL   BACKGND        ;WRITE BACKGROUND OF ZEROS
MOV    #FIRST,R0
MOV    #SIZE,R1
CMP    #1,PROTYP      ;IS THIS AN 11/44?
BEQ    1$             ;BRANCH IF IT IS
MOV    #MTP005,SUPDOADD ;SET UP LINKS
MOV    #MTP005+14,MTPB04+16
CALL   SUPD04
MOV    #MTPA04+14,MTPB04+16 ;RESET TEST'S ORIGINAL VALUE
RETURN
1$:    BMOV    MTP005
        BMOV    MTPB04,KDPARG,8.
        MOV    #KDPAR0,UIPAR5
        MOV    #UIPAR6,KDPAR7
        CALL   SUPD02
        RETURN
    
```

4050 020776 012737 000005 002260  
 4051 021004 012737 000004 002276  
 4052 021012 005002  
 4053 021014 004737 036470  
 4054 021020 012700 060000  
 4055 021024 012701 040000  
 4056 021030 022737 000001 003710  
 4057 021036 001414  
 4058 021040 012737 027316 002254  
 4059 021046 012737 027332 027314  
 4060 021054 004737 026615  
 4061 021060 012737 027256 027314  
 4062 021066 000207  
 4063 021070  
 4064 021076  
 4065 021110 012737 172360 177652  
 4066 021116 012737 177654 172376  
 4067 021124 004737 026440  
 4068 021130 000207

4071 021132

MT0006: SUBST <<MT0006 SETUP INITIAL DATA TEST>>

\*\*\*\*\*  
 ;\*SUBTEST MT0006 SETUP INITIAL DATA TEST  
 ;\*\*\*\*\*

4072	021132	012737	000006	002260	MOV	#6,REALPAT	;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4073	021140	012737	000004	002276	MOV	#4,PCBUMP	;TRAPS ADD 4 TO PC
4074	021146	012701	002362		MOV	#TESTADD,R1	
4075	021152	012737	027352	002254	MOV	#MTP006,SUPDOADD	
4076	021160	004737	026602		CALL	SUPDO3	;DO IT IN SUPERVISOR MODE
4077	021164	000207			RETURN		
4078	021166						

MT0007: SUBST <<MT0007 SETUP ADDRESS BIT TEST>>

\*\*\*\*\*  
 ;\*SUBTEST MT0007 SETUP ADDRESS BIT TEST  
 ;\*\*\*\*\*

4079	021166	012737	000007	002260	MOV	#7,REALPAT	;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4080	021174	005002			CLR	R2	
4081	021176	004737	036470		CALL	BACKGND	;OF ZEROS
4082	021202	012701	060000		MOV	#FIRST,R1	
4083	021206	012702	000001		MOV	#1,R2	
4084	021212	050201			BIS	R2,R1	
4085	021214	012737	027552	002254	MOV	#MTP007,SUPDOADD	
4086	021222	004737	026602		CALL	SUPDO3	;DO IT IN SUPERVISOR MODE
4087	021226	000207			RETURN		
4088	021230						

MT0010: SUBST <<MT0010 SETUP BYTE ADDRESSING TEST>>

\*\*\*\*\*  
 ;\*SUBTEST MT0010 SETUP BYTE ADDRESSING TEST  
 ;\*\*\*\*\*

4089	021230	012737	000010	002260	MOV	#10,REALPAT	;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4090	021236	012737	000004	002276	MOV	#4,PCBUMP	;TRAPS ADD 4 TO PC
4091	021244	013704	002362		MOV	TESTADD,R4	
4092	021250	012737	027652	002254	MOV	#MTP010,SUPDOADD	
4093	021256	004737	026602		CALL	SUPDO3	;DO IT IN SUPERVISOR MODE
4094	021262	000207			RETURN		



```

4097 021264      MT0011: SUBTST <<MT0011      SETUP CREATE SINGLE BIT ERROR TEST>>
;*****
;*SUBTEST      MT0011  SETUP CREATE SINGLE BIT ERROR TEST
;*****
4098 021264      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
4099 021300      IF $PASS NE #0 THEN $RETURN
4100 021310      END; OF IF ACTFLAG
4101 021310      012737 000011 002260      MOV      #11,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4102 021316      012737 027760 002254      MOV      #MTP011,SUPDOADD
4103 021324      004737 026602      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
4104 021330      000207      RETURN
4105 021332      MT0012: SUBTST <<MT0012      SETUP WRITE BYTE CLEARS SBE TEST>>
;*****
;*SUBTEST      MTOC12  SETUP WRITE BYTE CLEARS SBE TEST
;*****
4106 021332      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
4107 021346      IF $PASS NE #0 THEN $RETURN
4108 021356      END; OF IF ACTFLAG
4109 021356      012737 00C012 002260      MOV      #12,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4110 021364      013700 002102      MOV      BANKINDEX,R0
4111 021370      IF #BIT12 SET.IN CONFIG+2(R0)
4112 021400      012705 040000      MOV      #40000,R5
4113 021404      ELSE
4114 021406      012705 000002      MOV      #2,R5
4115 021412      END; OF IF #BIT12
4116 021412      012737 03C556 002254      MOV      #MTP012,SUPDOADD
4117 021420      004737 026602      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
4118 021424      000207      RETURN
4119 021426      MT0013: SUBTST <<MT0013      SETUP CREATE DOUBLE BIT ERROR TEST>>
;*****
;*SUBTEST      MT0013  SETUP CREATE DOUBLE BIT ERROR TEST
;*****
4120 021426      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
4121 021442      IF $PASS NE #0 THEN $RETURN
4122 021452      END; OF IF ACTFLAG
4123 021452      012737 000013 002260      MOV      #13,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4124 021460      012737 031144 002254      MOV      #MTP013,SUPDOADD
4125 021466      012737 000003 002074      MOV      #3,NOPAR        ;INDICATE PARITY ACRION
4126 021474      004737 026602      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
4127 021500      000207      RETURN
4128 021502      MT0014: SUBTST <<MT0014      SETUP WRITE INHIBIT DURING DATIP WITH DBE>>
;*****
;*SUBTEST      MT0014  SETUP WRITE INHIBIT DURING DATIP WITH DBE
;*****
4129 021502      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
4130 021516      IF $PASS NE #0 THEN $RETURN
4131 021526      END; OF IF ACTFLAG
4132 021526      IF KFLAG IS FALSE THEN $RETURN
4133 021536      012737 000014 002260      MOV      #14,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4134 021544      012737 031660 002254      MOV      #MTP014,SUPDOADD
4135 021552      004737 026602      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
4136 021556      000207      RETURN
    
```

```

4139 021560      MT0015: SUBTST <<MT0015      SETUP WRITE INHIBIT OF BYTE WITH DBE>>
:*****
:*SUBTEST      MT0015  SETUP WRITE INHIBIT OF BYTE WITH DBE
:*****
4140 021560      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
4141 021574      IF $PASS NE #0 THEN $RETURN
4142 021604      END ;OF IF ACTFLAG
4143 021604      012737 000015 002260      MOV      #15,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4144 021612      012737 032442 002254      MOV      #MTP015,SUPDOADD
4145 021620      004737 026602      CALL     SUPDO3          ;DO IT IN SUPERVISOR MODE
4146 021624      000207      RETURN
4147 021626      MT0016: SUBTST <<MT0016      SETUP WRITE INHIBIT OF WORD WITH DBE>>
:*****
:*SUBTEST      MT0016  SETUP WRITE INHIBIT OF WORD WITH DBE
:*****
4148 021626      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
4149 021642      IF $PASS NE #0 THEN $RETURN
4150 021652      END ;OF IF ACTFLAG
4151 021652      012737 000016 002260      MOV      #16,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4152 021660      012737 033206 002254      MOV      #MTP016,SUPDOADD
4153 021666      004737 026602      CALL     SUPDO3          ;DO IT IN SUPERVISOR MODE
4154 021672      000207      RETURN
4155 021674      MT0017: SUBTST <<MT0017      SETUP HOLDING 1'S & 0'S>>
:*****
:*SUBTEST      MT0017  SETUP HOLDING 1'S & 0'S
:*****
4156 021674      012737 000017 002260      MOV      #17,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4157 021702      012737 033770 002254      MOV      #MTP017,SUPDOADD
4158 021710      004737 026602      CALL     SUPDO3          ;DO IT IN SUPERVISOR MODE
4159 021714      000207      RETURN
    
```

4162 021716

MTC020: SUBTST <<MTO020 SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST>>

\*\*\*\*\*  
 :\*SUBTEST MTO020 SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST  
 \*\*\*\*\*

4163 021716  
 4164 021732  
 4165 021742  
 4166 021742 012737 000020 002260  
 4167 021750 012737 000003 002074  
 4168 021756 005001  
 4169 021760 005004  
 4170 021762 012700 060000  
 4171 021766 013702 002102  
 4172 021772  
 4173 022000  
 4174 022000  
 4175 022006  
 4176 022016 012737 140000 002362  
 4177 022024 012705 140002  
 4178 022030  
 4179 022032  
 4180 022032  
 4181 022032  
 4182 022040  
 4183 022050 012737 140000 002362  
 4184 022056 012705 140000  
 4185 022062  
 4186 022064  
 4187 022064  
 4188 022064 012737 160000 002362  
 4189 022072 012705 160002  
 4190 022076  
 4191 022076 010537 002364  
 4192 022102  
 4193 022104 005737 002312  
 4194 022110 001401  
 4195 022112 000207  
 4196 022114 012737 120000 002362 1\$:  
 4197 022122 012705 160000  
 4198 022126 010537 002364  
 4199 022132 005237 002232  
 4200 022136 005201  
 4201 022140 005204  
 4202 022142  
 4203 022142 016203 002624  
 4204 022146  
 4205 022154 060437 002362  
 4206 022160 060437 002364  
 4207 022164 060400  
 4208 022166 060405  
 4209 022170 072327 177775  
 4210 022174  
 4211 022176 006303  
 4212 022200  
 4213 022200 000303  
 4214 022202 042703 177741  
 4215 022206 010337 002146

```

IF ACTFLAG IS TRUE OR ACTFLAG IS TRUE
  IF $PASS NE #0 THEN $RETURN
END ;OF IF ACTFLAG
MOV #20,REALPAT ;SETUP PATTERN NUMBER FOR TYPEDOUT & DISPLAY
MOV #3,NOPAR ;INDICATE PARITY ACTION
CLR R1 ;CLEAR LOOP COUNTER
CLR R4 ;CLEAR INTERLEAVE ODD/EVEN FLAG
MOV #FIRST,R0
MOV BANKINDEX,R2 ;SET BANK INDEX
MTC020: IF INTFLAG IS FALSE
  BEGIN MTB020
  IF NO22BIT IS TRUE
    IF BANK EQ #7
      MOV #14000,TESTADD ;SET UP 12K NON-INTERLEAVED VIRT ADDR
      MOV #140002,R5
      LEAVE MTB020
    END; OF IF BANK
  END; OF IF NO22BIT
  IF NO22BIT IS FALSE
    IF BANK EQ #177
      MOV #140000,TESTADD
      MOV #140000,R5
      LEAVE MTB020
    END; OF IF BANK
  END; OF IF NO22BIT
  MOV #LAST+2,TESTADD
  MOV #LAST+4,R5
  END MTB020
  MOV R5,TESTADD+2 ;SET UP NON-INTERLEAVED VIRT. ADDR.
ELSE
  TST SKIPMK ;IS THIS BANK IN SKIP RANGE?
  BEQ 1$ ;BANK IS OUT OF RANGE - DO TEST
  RETURN ;LEAVE TEST-BANK'S ALREADY TESTED
  MOV #120000,TESTADD ;SET UP 1ST INTERLEAVED VIRT. ADDR.
  MOV #LAST+2,R5 ;SET UP END OF BANK FLAG
  MOV R5,TESTADD+2 ;SET UP 2ND INT'L. VIRT. ADDR.
  INC SPLTCSR ;FLAG THE MAPPING ROUTINE FOR INTERLEAVING
  INC R1 ;SET LOOP COUNTER FOR INTERLEAVING
  INC R4 ;SET ODD/EVEN FLAG
END; OF IF INTFLAG
MOV CONFIG(R2),R3 ;SET UP CSR NUMBER
IF R4 EQ #2 ;IF THE SECOND TIME AROUND
  ADD R4,TESTADD
  ADD R4,TESTADD+2 ;TEST THE A1 ASSERTED ADDRESSES
  ADD R4,R0
  ADD R4,R5
  ASH #-3,R3 ;MOVE INTERLEAVED CSR NUMBER
ELSE
  ASL R3 ;MOVE CSR NUMBER
END; IF R4
SWAB R3
BIC #^C36,R3
MOV R3,CSRNO ;MOVE R3 INTO CSR NUMBER
  
```

```

4216 022212          IF #SWO SET.IN @SWR
4217 022222 104506   ENASBE                      ;TRAP ON SINGLE BIT ERRORS
4218 022224          ELSE
4219 022226 104472   ECCINIT                      ;TRAP ON UNCORRECTABLE ERRORS
4220 022230          END; OF IF #SWO
4221 022230          PUSH R2,R4
4222 022234          FOR MTV020 := #0 TO R1
4223 022240          PUSH R1
4224 022242 005002   CLR R2                      ;PATTERN TO WRITE INTO BANK
4225 022244 004737 036470 CALL BACKGND          ;SET UP ZEROS IN BANK
4226 022250          IF NO22BIT IS TRUE AND MTV020 EQ #1 AND BANK EQ #3
4227 022276 162737 020000 002362 SUB #20000,TESTADD    ;SET UP 12K INTERLEAVED BANK
4228 022304 162705 020000          SUB #20000,R5
4229 022310 010537 002364          MOV R5,TESTADD+2
4230 022314          END; OF IF NO22BIT
4231 022314 004737 022370          CALL MTO20Z          ;START TEST
4232 022320 005237 002232          INC SPLTCSR          ;UPDATE INTERLEAVED MAPPING FLAG
4233 022324          POP R1
4234 022326          END; OF FOR MTV020
4235 022340          POP R4,R2
4236 022344 005001   CLR R1                      ;RESET LOOP FLAG
4237 022346 005037 002232          CLR SPLTCSR          ;RESET INTERLEAVED MAP FLAG
4238 022352 022704 000001          CMP #1,R4          ;ODD/EVEN FLAG SET?
4239 022356 001605          BEQ MTLO20          ;BRANCH IF TRUE
4240 022360 005037 002074          CLR NOPAR          ;INDICATE PARITY ACTION
4241 022364 000207          RETURN
4242 022366 000000          MTV020: 0          ;VARIABLE FOR PAT 20
  
```

```

4244 022370 012702 000004          MTO20Z: MOV      #4,R2          ;SET UP WORD INCR/DECR AMOUNT
4245 022374 013701 002362          MOV      TESTADD,R1
4246 022400 013704 002364          MOV      TESTADD+2,R4
4247 022404 012703 100000          MOV      #BIT15,R3
4248 022410          IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
4249 022426          GOTO    MTO20Y
4250 022430          END ;OF IF #SW11
4251 022430 022737 000001 003710        CMP      #1,PROTYP          ;IS THIS AN 11/44?
4252 022436 001411          BEQ      1$                ;BRANCH IF IT IS
4253 022440 012737 034046 002254        MOV      #MTPA20,SUPDOADD
4254 022446 012737 034062 002264        MOV      #MTPA20+14,PARHERE ;VECTOR FOR TRAPS
4255 022454 004737 026602          CALL     SUPD03
4256 022460 000410          BR       2$
4257 022462          1$: BMOV     MTPA20
4258 022470 012737 177654 002264        MOV      #UIPAR6,PARHERE    ;VECTOR FOR TRAPS
4259 022476 004737 026424          CALL     SUPD01
4260 022502 022737 000001 003710        2$: CMP      #1,PROTYP          ;IS THIS AN 11/44?
4261 022510 001411          BEQ      4$                ;BRANCH IF IT IS
4262 022512 012737 034076 002254        MOV      #MTPB20,SUPDOADD
4263 022520 012737 034106 002264        MOV      #MTPB20+10,PARHERE ;VECTOR FOR TRAPS
4264 022526 004737 026616          CALL     SUPD04
4265 022532 000410          BR       MTO20Y
4266 022534          4$: BMOV     MTPB20
4267 022542 012737 177650 002264        MOV      #UIPAR4,PARHERE    ;VECTOR FOR TRAPS
4268 022550 004737 026440          CALL     SUPD02
4269 022554 005737 002134          MTO20Y: TST      INTFLAG          ;ARE WE INTERLEAVED?
4270 022560 001405          BEQ      7$                ;BRANCH IF NOT INTERLEAVED
4271 022562 162701 040000          SUB      #40000,R1          ;RESET FIRST WORD TO BEGINNING OF BANK
4272 022566 162704 040000          SUB      #40000,R4          ;RESET SECOND WORD TO BEGINNING OF BANK
4273 022572 000404          BR       8$
4274 022574 012701 060000          7$: MOV      #FIRST,R1          ;RESET FIRST WORD TO BEGINNING OF BANK
4275 022600 012704 060002          MOV      #FIRST+2,R4        ;RESET SECOND WORD TO BEGINNING OF BANK
4276 022604 022737 000001 003710        8$: CMP      #1,PROTYP          ;IS THIS AN 11/44?
4277 022612 001411          BEQ      1$                ;BRANCH IF IT IS
4278 022614 012737 034126 002254        MOV      #MTPC20,SUPDOADD
4279 022622 012737 034136 002264        MOV      #MTPC20+10,PARHERE ;VECTOR FOR TRAPS
4280 022630 004737 026602          CALL     SUPD03
4281 022634 000410          BR       2$
4282 022636          1$: BMOV     MTPC20
4283 022644 012737 177650 002264        MOV      #UIPAR4,PARHERE    ;VECTOR FOR TRAPS
4284 022652 004737 026424          CALL     SUPD01
4285 022656 022737 000001 003710        2$: CMP      #1,PROTYP          ;IS THIS AN 11/44?
4286 022664 001411          BEQ      3$                ;BRANCH IF IT IS
4287 022666 012737 034156 002254        MOV      #MTPD20,SUPDOADD
4288 022674 012737 034172 002264        MOV      #MTPD20+14,PARHERE ;VECTOR FOR TRAPS
4289 022702 004737 026616          CALL     SUPD04
4290 022706 000410          BR       4$
4291 022710          3$: BMOV     MTPD20
4292 022716 012737 177654 002264        MOV      #UIPAR6,PARHERE    ;VECTOR FOR TRAPS
4293 022724 004737 026440          CALL     SUPD02
4294 022730 022737 000001 003710        4$: CMP      #1,PROTYP          ;IS THIS AN 11/44?
4295 022736 001411          BEQ      5$                ;BRANCH IF IT IS
4296 022740 012737 034206 002254        MOV      #MTPE20,SUPDOADD
4297 022746 012737 034216 002264        MOV      #MTPE20+10,PARHERE ;VECTOR FOR TRAPS
4298 022754 004737 026616          CALL     SUPD04
4299 022760 000410          BR       6$
4300 022762          5$: BMOV     MTPE20
    
```

```
4301 022770 012737 177650 002264      MOV    #UIPAR4,PARTHERE      ;VECTOR FOR TRAPS
4302 022776 004737 026440              CALL   SUPD02
4303 023002 104503                      6$: CLR1CSR                  ;CLEAR 1 SELECTED CSR
4304 023004 000207                      RETURN
```

```

4307 023006      MT0021: SUBTST <<MT0021      SETUP MARCHING 0'S & 1'S TEST>>
:*****
:*SUBTEST      MT0021 SETUP MARCHING 0'S & 1'S TEST
:*****
4308 023006      SET NOSCOPE
4309 023014      MOV      #21,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4310 023022      MOV      BAKPAT,R2
4311 023026      CALL     BACKGND
4312 023032      MOV      R2,R3
4313 023034      SWAB    R3
4314 023036      MOV      #LAST+2,R1
4315 023042      MOV      R1,R5
4316 023044      MOV      #FIRST,R4
4317 023050      CMP      #1,PROTYP      ;IS THIS AN 11/44?
4318 023056      BEQ     1$              ;BRANCH IF IT IS
4319 023060      CMP      #3,PROTYP      ;IS THIS AN 11/24?
4320 023066      BEQ     3$              ;BRANCH IF SO
4321 023070      CMP      #7,BANK
4322 023076      BNE     3$
4323 023100      MOV      #140000,R1
4324 023104      MOV      R1,R5
4325 023106      MOV      #MTPA21,SUPDOADD 3$:
4326 023114      CALL     SUPD03
4327 023120      MOV      #MTPB21,SUPDOADD 002254
4328 023126      CALL     SUPD04
4329 023132      MOV      R4,R1
4330 023134      MOV      #MTPC21,SUPDOADD 002254
4331 023142      CALL     SUPD04
4332 023146      MOV      #MTPD21,SUPDOADD 002254
4333 023154      CALL     SUPD04
4334 023160      BR      2$
4335 023162      CMP      #177,BANK      1$:
4336 023170      BNE     4$
4337 023172      MOV      #140000,R1
4338 023176      MOV      R1,R5
4339 023200      BMOV    MTPA21          4$:
4340 023206      CALL     SUPD01
4341
4342 023212      BMOV    MTPB21
4343 023220      CALL     SUPD02
4344
4345 023224      MOV      R4,R1
4346 023226      BMOV    MTPC21
4347 023234      CALL     SUPD02
4348
4349 023240      BMOV    MTPD21
4350 023246      CALL     SUPD02
4351 023252      CLR     NOSCOPE      2$:
4352 023256      RETURN

```

4354 023260

MT0022: SUBTST <<MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST>>

\*\*\*\*\*

\*SUBTEST MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST

\*\*\*\*\*

4355 023260 004737 026370

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE

4356 023264

ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN

4357 023270 012737 000022 002260

MOV #22,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

4358 023276 012737 034402 002254

MOV #MTP022,SUPDOADD

4359 023304 004737 026602

CALL SUPDO3 ;DO IT IN SUPERVISOR MODE

4360 023310 000207

RETURN

4361

4362 023312

MT0023: SUBTST <<MT0023 SHIFTING DIAGONAL TEST>>

\*\*\*\*\*

\*SUBTEST MT0023 SHIFTING DIAGONAL TEST

\*\*\*\*\*

4363 023312 004737 026370

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE

4364 023316

ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN

4365 023322 012737 000023 002260

MOV #23,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

4366 023330 012737 034402 002254

MOV #MTP022,SUPDOADD

4367 023336

SET DIAGFLAG ;IDENTIFY DIAGONAL TEST TO MTP022

4368 023344 004737 026602

CALL SUPDO3 ;DO IT IN SUPERVISOR MODE

4369 023350 005037 002002

CLR DIAGFLAG

4370 023354 000207

RETURN



4372 023356

4373 023356 004737 026370

4374 023362

4375 023366

4376 023374 012737 000024 002260

4377 023402 013702 002572

4378 023406 004737 036470

4379 023412 010203

4380 023414 010304

4381 023416 000304

4382 023420 012701 060000

4383 023424 012705 157776

4384 023430 022737 000001 003710

4385 023436 001417

4386 023440 022737 000003 003710

4387 023446 001406

4388 023450 022737 000007 002100

4389 023456 001002

4390 023460 012705 137776

4391 023464 104415

4392 023466 012737 035116 002254

4393 023474 000440

4394 023476 022737 000177 002526

4395 023504 001002

4396 023506 012705 137776

4397 023512 104415

4398 023514

4399 023522

4400 023534

4401 023546 012737 172260 002254

4402 023554 012737 172260 177676

4403 023562 012737 172360 172272

4404 023570 012737 177660 172374

4405 023576 004737 026616

4406

4407

4408 023602 104416

4409 023604 000302

4410 023606 000303

4411 023610 004737 026616

4412 023614 005037 002410

4413 023620 000207

4414 023622

4415 023622

4416 023636

4417 023646

4418 023646 012737 000025 002260

4419 023654 012737 035150 002254

4420 023662 004737 026602

4421 023666 000207

```

MT0024: SUBTST <<MT0024      SETUP FAST GALLOPING PATTERN TEST>>
:*****
:*SUBTEST      MT0024      SETUP FAST GALLOPING PATTERN TEST
:*****
CALL      KAMITEST      ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN $RETURN  ;IF NOT IN KAMIKAZE MODE RETURN
SET      NOSCOPE
MOV      #24,REALPAT    ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV      BAKPAT,R2
CALL     BACKGND
MOV      R2,R3
MOV      R3,R4
SWAB     R4
MOV      #FIRST,R1
MOV      #LAST,R5
CMP      #1,PROTYP
BEQ      1$
CMP      #3,PROTYP
BEQ      3$
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      #MT0025,SUPDOADD
CALL     SUPD03      ;DO IT IN SUPERVISOR MODE
RETURN

```

4424 023670

MT0026: SUBTST <<MT0026 SETUP RANDOM DATA TEST>>

\*\*\*\*\*  
 ;+SUBTEST MT0026 SETUP RANDOM DATA TEST  
 \*\*\*\*\*

4425	023670	012737	000026	002260		MOV	#26,REALPAT	
4426	023676	005037	002276			CLR	PCBUMP	;TRAPS DO NOT ADD TO THE PC
4427	023702	013703	002544			MOV	SEEDLO,R3	;INITIALIZE RANDOM NUMBERS
4428	023706	013702	002542			MOV	SEEDHI,R2	
4429	023712	010305				MOV	R3,R5	
4430	023714	010204				MOV	R2,R4	
4431	023716	012701	060000			MOV	#FIRST,R1	
4432	023722	012700	020000			MOV	#SIZE/2,R0	
4433	023726	022737	000001	003710		CMP	#1,PROTYP	;DO WE HAVE AN 11/44?
4434	023734	001437				BEQ	1\$	;BRANCH IF WE DO
4435	023736	022737	000003	003710		CMP	#3,PROTYP	;11/24?
4436	023744	001406				BEQ	3\$	;BRANCH IF SO
4437	023746	022737	000007	002100		CMP	#7,BANK	
4438	023754	001002				BNE	3\$	
4439	023756	012700	014000			MOV	#14000,R0	
4440	023762	104415			3\$:	SAVREG		
4441	023764	012737	035622	035722		MOV	#MTPA26+4,MTPD26+14	
4442	023772	012737	035616	002254		MOV	#MTPA26,SUPDOADD	
4443	024000	004737	026602			CALL	SUPDO3	
4444	024004	005037	035646			CLR	RANODD	;FOR ERROR REPORTING
4445	024010	012737	035636	035722		MOV	#MTPB26+4,MTPD26+14	;SET UP NEXT LINK
4446	024016	012737	035632	002254		MOV	#MTPB26,SUPDOADD	
4447	024024	104416				RESREG		
4448	024026	004737	026602			CALL	SUPDO3	
4449	024032	000452				BR	2\$	
4450	024034	022737	000177	002100	1\$:	CMP	#177,BANK	
4451	024042	001002				BNE	4\$	
4452	024044	012700	014000			MOV	#14000,R0	
4453	024050	104415			4\$:	SAVREG		
4454	024052					BMOV	MTPA26	;WRITE ROUTINE TO FAST MEMORY
4455	024060					BMOV	MTPC26,KDPAR0,8.	;RANDOM SUBPROGRAM TO FAST MEMORY
4456	024072	012737	000730	172376		MOV	#730,KDPAR7	;WRITES 'BR .-116' IN (BR SDPAR0)
4457	024100					BMOV	MTPD26,SDPAR0,8.	;RANDOM SUBSUBPROGRAM TO FAST MEMORY
4458	024112	012737	172360	177642		MOV	#KDPAR0,UIPAR1	
4459	024120	012737	177644	172274		MOV	#UIPAR2,SDPAR6	
4460	024126	004737	026424			CALL	SUPDO1	;WRITE RANDOM DATA
4461	024132	005037	035646			CLR	RANODD	;FOR ERROR REPORTING
4462	024136					BMOV	MTPB26	;READ ROUTINE TO FAST MEMORY
4463	024144	012737	172360	177642		MOV	#KDPAR0,UIPAR1	;SET UP PAR LINK
4464	024152	104416				RESREG		
4465	024154	004737	026424			CALL	SUPDO1	;READ RANDOM DATA
4466	024160	010337	002544		2\$:	MOV	R3,SEEDLO	;UPDATE FOR NEW RANDOM NUMBERS
4467	024164	010237	002542			MOV	R2,SEEDHI	
4468	024170	000207				RETURN		

4471 024172

MT0027: SUBTST <<MT0027 UNIQUE BANK TEST>>

\*\*\*\*\*  
\*SUBTEST MT0027 UNIQUE BANK TEST  
\*\*\*\*\*

4472  
4473  
4474 024172 012737 000027 002260  
4475 024200 104502  
4476 024202 022737 000001 003710  
4477 024210 001404  
4478 024212 012737 026602 002472  
4479 024220 000414  
4480 024222  
4481 024230 012737 177646 002254  
4482 024236 012737 026424 002472  
4483 024244  
4484 024252  
4485 024260  
4486 024264 004737 044240  
4487 024270  
4488 024304 104511  
4489 024306  
4490 024312 012700 060000  
4491 024316 010004  
4492 024320 012701 040000  
4493 024324 010103  
4494 024326  
4495 024336 022737 000001 003710  
4496 024344 001403  
4497 024346 012737 036142 002254  
4498 024354 004777 156112  
4499 024360  
4500 024360  
4501 024370 022737 000001 003710  
4502 024376 001403  
4503 024400 012737 036150 002254  
4504 024406 004737 026602  
4505 024412  
4506 024412  
4507 024412  
4508 024426  
4509 024442  
4510 024450 005037 002376  
4511 024454 000207  
4512 024456  
4513 024456  
4514 024464  
4515 024472 004737 044240  
4516 024476  
4517 024512  
4518 024516 005102  
4519 024520 012700 060000  
4520 024524 010004  
4521 024526 012701 040000  
4522 024532 010103  
4523 024534  
4524 024544 022737 000001 003710

:MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA  
:WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)  
MOV #27,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY  
CLRCRS ;CLEAR CSRS  
CMP #1,PROTYP ;IS THIS AN 11/44?  
BEQ 1\$ ;BRANCH IF TRUE  
MOV #SUPD03,LINK1 ;SET UP LINK  
BR STAR27 ;BRANCH TO RUN  
1\$: BMOV MTP034  
WARN7: MOV #UIPAR3,SUPDOADD  
MOV #SUPD01,LINK1 ;SET UP LINK  
SET NOFSMODE  
STAR27: FOR I := #1 TO #2  
FOR BANK := #0 TO LASTBANK  
CALL EXBANK  
IF ACFLAG IS TRUE AND RRFLAG IS FALSE  
INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON 'BANK'  
LET R2 := BANK  
MOV #FIRST,R0  
MOV R0,R4  
MOV #SIZE,R1  
MOV R1,R3  
IF I EQ #1  
CMP #1,PROTYP  
BEQ 2\$  
MOV #MTP034,SUPDOADD  
2\$: CALL @LINK1  
END ;OF IF  
IF I EQ #2  
CMP #1,PROTYP  
BEQ 3\$  
MOV #MTP034+6,SUPDOADD  
3\$: CALL SUPD03  
END ;OF IF  
END ;OF IF  
END ;OF FOR BANK  
END ;OF FOR I  
IF FS7FLAG IS TRUE  
CLR NOFSMODE  
RETURN  
END ;OF IF FS7FLAG  
FOR I := #1 TO #2  
FOR BANK := LASTBANK DOWNT0 #0  
CALL EXBANK  
IF ACFLAG IS TRUE AND RRFLAG IS FALSE  
LET R2 := BANK  
COM R2  
MOV #FIRST,R0  
MOV R0,R4  
MOV #SIZE,R1  
MOV R1,R3  
IF I EQ #1  
CMP #1,PROTYP

CZMSDB0 MS11-L/M DIAGNOSTIC  
MT0027 UNIQUE BANK TEST

MACRO M1113 07-OCT-80 18:01 PAGE 153-1 SEQUENCE 196

I 16

SEQ 0203

4525	024552	001403		
4526	024554	012737	036142	002254
4527	024562	004777	155704	
4528	024566			
4529	024566			
4530	024576	022737	000001	003710
4531	024604	001403		
4532	024606	012737	036150	002254
4533	024614	004737	026602	
4534	024620			
4535	024620			
4536	024620			
4537	024634			
4538	024650	005037	002376	
4539	024654	000207		

```
BEQ 4$  
MOV #MTPO34,SUPDOADD  
4$: CALL @LINK1  
END ;OF IF  
IF I EQ #2  
CMP #1,PROTYP  
BEQ 5$  
MOV #MTPO34+6,SUPDGADD  
5$: CALL SUPD03  
END ;OF IF  
END ;OF IF  
END ;OF FOR BANK  
END ;OF FOR I  
CLR NOFSMODE  
RETURN
```

4542 024656

MT0030: SUBTST <<MT0030 SETUP FLUSH OUT DBE'S TEST>>

\*\*\*\*\*  
; \*SUBTEST MT0030 SETUP FLUSH OUT DBE'S TEST  
\*\*\*\*\*

4543 024656 005037 002256  
4544 024662  
4545 024670 012737 000030 002260  
4546 024676 012737 000001 002074  
4547 024704 022737 000001 003710  
4548 024712 001007  
4549 024714  
4550 024722 012737 026424 002472  
4551 024730 000406  
4552 024732 012737 026602 002472 4\$:  
4553 024740 012737 035724 002254  
4554 024746 104470 1\$:  
4555 024750  
4556 024764  
4557 024770 004737 044240  
4558 024774  
4559 025002  
4560 025016 012701 040000  
4561 025022 012700 060000  
4562 025026 004777 155440  
4563 025032  
4564 025032  
4565 025032  
4566 025046  
4567 025054  
4568 025062 104502  
4569 025064 004737 042466  
4570 025070  
4571 025072 104472  
4572 025074  
4573 025110 000207  
4574 025112  
4575 025112 013737 002270 002\*00  
4576 025120 004737 044240  
4577 025124 004737 024670  
4578 025130 104472  
4579 025132 004737 043356  
4580 025136 000207  
4581 025140  
4582 025140 104472  
4583 025142  
4584 025156 000207

CLR PASFLG  
SET FULLREL  
MTA030: MOV #30,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY  
MOV #1,NOPAR ;INDICATE COUNT PARITY ERRORS  
CMP #1,PROTYP  
BNE 4\$  
BMOV MTP030  
MOV #SUPD01,LINK1  
BR 1\$  
4\$: MOV #SUPD03,LINK1  
MOV #MTP030,SUPDOADD  
1\$: ECCDIS ;DISABLE ERROR CORRECTION  
SET NOFSMODE,NOSCOPE  
FOR BANK := #0 TO LASTBANK  
CALL EXBANK  
IF MKFLAG IS TRUE  
IF ACFLAG IS TRUE AND RRFLAG IS FALSE  
MOV #SIZE,R1  
MOV #FIRST,R0  
CALL @LINK1  
END ;OF IF ACFLAG  
END ;OF IF MKFLAG  
END ;OF FOR  
IF PASFLG IS FALSE  
SET PASFLG  
CLRCR ;CLEAR CSRS  
CALL RELOCATE  
ON.ERROR  
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)  
CLEAR NOFSMODE,NOSCOPE,FULLREL  
RETURN  
END ;OF ON.ERROR  
MOV NEWBANK,BANK  
CALL EXBANK  
CALL MTA030  
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)  
CALL UNRELOCATE  
RETURN  
END ;OF IF PASFLG  
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)  
CLEAR NOFSMODE,NOSCOPE,FULLREL  
RETURN

4587 025160

MT0031: SUBTST <<MT0031 SETUP SOB-A-LONG TEST>>  
 :\*\*\*\*\*  
 :\*SUBTEST MT0031 SETUP SOB-A-LONG TEST  
 :\*\*\*\*\*

4588	025160	004737	026370		CALL	KAMITEST		;CHECK FOR KAMIKAZE MODE
4589	025164				ON.ERROR	THEN \$RETURN		;IF NOT IN KAMIKAZE MODE RETURN
4590	025170				SET	NOSCOPE		
4591	025176	012737	000031	002260	MOV	#31,REALPAT		;SETUP PATTERN NUMBER FOR TYPECJT & DISPLAY
4592	025204	005037	002074		CLR	NOPAR		;SETUP PARITY ACTION
4593	025210				MAP	BANK		;MAP FIRST SO BLOCK MOVE WORKS
4594	025224				TESTAREA			;ENTER TEST MODE
4595	025232				BMOV	MTPO31,FIRST,SOBLENGTH/2		
4596	025244	104417			KERNEL			;ENTER KERNEL MODE
4597	025246	013702	002532		MOV	SOBK,R2		
4598	025252	010200			MOV	R2,R0		
4599	025254	012701	100776		MOV	#100776,R1		;COMPLEMENT OF INSTRUCTION 'SOB R0,DOT'
4600	025260	012705	060056		MOV	#FIRST+SOBLENGTH,R5		
4601	025264	012737	060002	002254	MOV	#FIRST+2,SUPDOADD		
4602	025272	012737	160000	002472	MOV	#LAST+2,LINK1		
4603	025300	005737	002426		TST	NOSUPER		
4604	025304	001005			BNE	1\$		
4605	025306	023737	172252	172254	CMP	SIPAR5,SIPAR6		
4606	025314	001405			BEQ	2\$		
4607	025316	000407			BR	3\$		
4608	025320	023737	177652	177654	1\$:	CMP	UIPAR5,UIPAR6	
4609	025326	001003			BNE	3\$		
4610	025330	012737	140000	002472	2\$:	MOV	#140000,LINK1	
4611	025336	004737	026616		3\$:	CALL	SUPD04	
4612	025342	005037	002410		CLR	NOSCOPE		
4613	025346	000207			RETURN			

4616 025350

MT0032: SUBTST <<MT0032 SETUP WRITE RECOVERY TEST>>

\*\*\*\*\*  
 :\*SUBTEST MT0032 SETUP WRITE RECOVERY TEST  
 \*\*\*\*\*

4617	025350	004737	026370			CALL	KAMITEST		;CHECK FOR KAMIKAZE MODE
4618	025354					ON.ERROR	THEN \$RETURN		;IF NOT IN KAMIKAZE MODE RETURN
4619	025360					SET	NOSCOPE		
4620	025366	012737	000032	002260		MOV	#32,REALPAT		;SETUP PATTERN NUMBFR FOR TYPEOUT & DISPLAY
4621	025374	005037	002074			CLR	NOPAR		;SETUP PARITY ACTION
4622	025400					MAP	BANK		;MAP FIRST SO THAT THE BLOCK MOVE WORKS
4623	025414	012700	010247			MOV	#10247,R0		;OP CODE OF INSTRUCTION 'MOV R2,-(PC)'
4624	025420	012701	177667			MOV	#177667,R1		;OP CODE OF COMPLEMENT OF INSTRUCTION 'JMP (R0)'
4625	025424	012702	020000			MOV	#SIZE/2,R2		;USED FOR 1/2 BANK LOOP
4626	025430	010237	002472			MOV	R2,LINK1		
4627	025434	012703	060000			MOV	#FIRST,R3		
4628	025440	012704	160000			MOV	#LAST+2,R4		
4629	025444	005037	002474			CLR	LINK2		
4630	025450	005737	002426			TST	NOSUPER		
4631	025454	001005				BNE	1\$		
4632	025456	023737	172252	172254		CMP	SIPAR5,SIPAR6		
4633	025464	001405				BEQ	2\$		
4634	025466	000415				BR	3\$		
4635	025470	023737	177652	177654	1\$:	CMP	UIPAR5,UIPAR6		
4636	025476	001011				BNE	3\$		
4637	025500	012704	140000		2\$:	MOV	#140000,R4		
4638	025504	012702	014000			MOV	#14000,R2		
4639	025510	010237	002472			MOV	R2,LINK1		
4640	025514	012737	000001	002474		MOV	#1,LINK2		
4641									
4642	025522				3\$:	TESTAREA			;ENTER TEST MODE
4643									;MOVE TEST TO MEMORY UNDER TEST
4644	025530	010023			4\$:	MOV	R0,(R3)+		
4645	025532	010144				MOV	R1,-(R4)		
4646	025534	077203				SOB	R2,4\$		
4647									
4648	025536	022737	000001	003710		CMP	#1,PROTYP		
4649	025544	001003				BNE	5\$		
4650									;MOVE LAST PART OF TEST TO FASTCITY
4651	025546					BMOV	MTPO32		
4652	025554	104417			5\$:	KERNEL			;ENTER KERNEL MODE
4653									
4654	025556	012702	005141			MOV	#5141,R2		;OP CODE OF INSTRUCTION 'COM -(R1)'
4655	025562	012700	025700			MOV	#10\$,R0		;ADDRESS TO RETURN TO IN R0
4656	025566	012701	160000			MOV	#LAST+2,R1		;TOP OF BANK
4657	025572	012737	060000	002254		MOV	#FIRST,SUPDOADD		
4658	025600	005737	002474			TST	LINK2		
4659	025604	001402				BEQ	6\$		
4660	025606	012701	140000			MOV	#140000,R1		
4661	025612	004737	026616		6\$:	CALL	SUPDO4		
4662	025616	012703	020000			MOV	#SIZE/2,R3		
4663	025622	012705	000110			MOV	#110,R5		
4664	025626	012704	060000			MOV	#FIRST,R4		
4665	025632	005737	002474			TST	LINK2		
4666	025636	001402				BEQ	7\$		
4667	025640	012703	014000			MOV	#14000,R3		
4668	025644	022737	000001	003710	7\$:	CMP	#1,PROTYP		
4669	025652	001406				BEQ	8\$		

```
4670 025654 012737 036012 002254      MOV      #MTP032,SUPDOADD
4671 025662 004737 026616              CALL     SUPD04
4672 025666 000402                      BR       9$
4673 025670 004737 026440      8$:     CALL     SUPD02
4674 025674 005037 002410      9$:     CLR      NOSCOPE
4675 025700 000207      10$:    RETURN
4676
4677
```

;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0032  
;ALSO A RETURN FROM THE 'CALL SUPD04' ABOVE



```

4680 025702          MT0033: SUBTST <<MT0033          SETUP BRANCH GOBBLE TEST>>
:*****
:*SUBTEST          MT0033  SETUP BRANCH GOBBLE TEST
:*****
4681 025702  004737  026370          CALL      KAMITEST          ;CHECK FOR KAMIKAZE MODE
4682 025706          ON.ERROR THEN $RETURN          ;IF NOT IN KAMIKAZE MODE RETURN
4683 025712          SET      NOSCOPE
4684 025720  012737  000033  002260  MOV      #33,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4685 025726  005037  002074          CLR      NOPAR          ;SETUP PARITY ACTION
4686 025732          MAP      BANK          ;MAP FIRST SO THAT BLOCK MOVE WORKS
4687
4688 025746          TESTAREA          ;ENTER TEST MODE
4689 025754          BMOV     MTP033,FIRST,GBLENGTH/2
4690 025766  104417          KERNEL          ;ENTER KERNEL MODE
4691
4692 025770  012705  060076          MOV      #FIRST+GBLENGTH,R5
4693 025774  012737  060004  002254  MOV      #FIRST+4,SUPDOADD
4694 026002  012701  060002          MOV      #FIRST+2,R1
4695 026006  012702  060003          MOV      #FIRST+3,R2
4696 026012  012737  160000  002472  MOV      #LAST+2,LINK1
4697 026020  005737  002426          TST      NOSUPER
4698 026024  001005          BNE     1$
4699 026026  023737  17225?  172254  CMP      SIPAR5,SIPAR6
4700 026034  001405          BEQ     2$
4701 026036  000407          BR      3$
4702 026040  023737  177652  177654  1$:    CMP      UIPAR5,UIPAR6
4703 026046  001003          BNE     3$
4704 026050  012737  140000  002472  2$:    MOV      #140000,LINK1
4705
4706 026056  004737  026616          3$:    CALL     SUPD04
4707 026062  005037  002410          CLR      NOSCOPE
4708 026066  000207          RETURN
4709
4710 026070          MTC034: SUBTST <<MT0034          SOFT ERROR - BACKGROUND PATTERN TEST>>
:*****
:*SUBTEST          MTC034  SOFT ERROR - BACKGROUND PATTERN TEST
:*****
4711 026070  012737  000034  002260  MOV      #34,REALPAT
4712 026076  012700  060000          MOV      #FIRST,R0
4713 026102  012701  040000          MOV      #SIZE,R1
4714 026106  013702  002560          MOV      SOFTPAT,R2
4715 026112  010103          MOV      R1,R3
4716 026114  013705  002102          MOV      BANKINDEX,R5
4717 026120  010004          MOV      R0,R4
4718 026122  022737  000001  003710  CMP      #1,PROTYP          ;IS THIS AN 11/44?
4719 026130  001006          BNE     1$          ;BRANCH IF NOT
4720 026132          BMOV     MTP034
4721 026140  012737  177646  002254  1$:    MOV      #UIPAR3,SUPDOADD
4722 026146          IF #BIT13 SET.IN CONFIG+2(R5)
4723          ;BACKGROUND PATTERN IS VALID
4724 026156  022737  000001  003710  CMP      #1,PROTYP
4725 026164  001403          BEQ     2$
4726 026166  012737  036150  002254  2$:    MOV      #MTP034+6,SUPDOADD
4727 026174  004737  026602          CALL     SUPD03          .READ IT
4728 026200          ELSE
4729          ;BACKGROUND PATTERN HAS BEEN INVALIDATED
4730 026202  022737  000001  003710  CMP      #1,PROTYP
    
```

```

4731 026210 001406          BEQ 3$
4732 026212 012737 036142 002254  MOV #MTPO34,SUPDOADD
4733 026220 004737 026602          CALL SUPDO3
4734 026224 000402          BR 4$
4735 026226 004737 026424 3$: CALL SUPDO1 ;WRITE IT
4736 026232 052765 020000 002626 4$: BIS #BIT13,CONFIG+2(R5) ;VALIDATE IT
4737 026240          END ;OF IF #BIT13
4738 026240 000207          RETURN
4739
4740 026242
    
```

```

MT0035: SUBTST <<MT0035          SETUP WORST CASE NOISE PARITY TEST>>
;*****
;*SUBTEST          MT0035          SETUP WORST CASE NOISE PARITY TEST
;*****
    
```

```

4741 026242 012737 000035 002260  MOV #35,REALPAT ;SET UP TEST NUMBER FOR DISPLAY
4742 026250 013703 002102  MOV BANKINDEX,R3
4743 026254 016301 002624  MOV CONFIG(R3),R1
4744 026260 000301          SWAB R1
4745 026262 042701 177760  BIC #^C17,R1
4746 026266 006301          ASL R1
4747 026270 010137 002146  MOV R1,CSRNO
4748 026274 023737 002146 002502  CMP CSRNO,PGMCSR
4749 026302 001001          BNE 1$
4750 026304 000207          RETURN
4751 026306 012702 052524 1$: MOV #52524,R2
4752 026312 004737 036470          CALL BACKGND ;WRITE BACKGROUND OF ALMOST ALT. 1'S AND 0'S
4753 026316 012737 036166 002254  MOV #MTPO35,SUPDOADD
4754 026324 004737 026602          CALL SUPDO3
4755 026330          IF QVFLAG IS TRUE THEN $RETURN
4756 026340 005102          COM R2
4757 026342 004737 036470          CALL BACKGND ;WRITE COMPLEMENT PATTERN INTO MUT
4758 026346 004737 026616          CALL SUPDO4
4759 026352 000207          RETURN
    
```

4762 026354

```
MT0999: SUBTST <<MT0999      SETUP NULL TEST>>
:*****
:*SUBTEST      MT0999  SETUP NULL TEST
:*****
      CLR      REALPAT
      SET      NULLFLAG
      RETURN
```

4763 026354 005037 002260

4764 026360

4765 026366 000207

4766

4767 026370

```
KAMITEST: SUBTST <<CHECK FOR KAMIKAZE MODE>>
:*****
:*SUBTEST      CHECK FOR KAMIKAZE MODE
:*****
      IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE
      $RETURN NOERROR          ;RUN THE TEST
      ELSE
      $RETURN ERROR           ;DON'T RUN THE TEST
      END ;OF IF KAMIKAZE
```

4768 026370

4769 026412

4770 026416

4771 026420

4772 026424

4775 026424

```
SUPD01: SUBTST <<SUBR EXECUTE PATTERN IN SUPERVISOR>>  
:*****  
:*SUBTEST SUBR EXECUTE PATTERN IN SUPERVISOR  
:*****  
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
```

4776 026424  
4777 026440 004737 055570  
4778 026444  
4779 026454 010037 002152  
4780 026460 012700 002154  
4781 026464 010120  
4782 026466 010220  
4783 026470 010320  
4784 026472 010420  
4785 026474 010520  
4786 026476 010620  
4787 026500 013700 002152  
4788 026504 012737 026520 002562  
4789 026512 013737 002562 002564  
4790 026520 012700 002170  
4791 026524 014006  
4792 026526 014005  
4793 026530 014004  
4794 026532 014003  
4795 026534 014002  
4796 026536 014001  
4797 026540 014000  
4798 026542  
4799 026550 012706 000740  
4800 026554 104424  
4801 026556 004737 177640  
4802 026562 104423  
4803 026564 104417  
4804 026566 000004  
4805 026570  
4806 026600 000207

```
SUPD02: CALL GETDIS  
PUSH $LPERR,$LPADR  
MOV R0,SUPDR0  
MOV #SUPDR1,R0  
MOV R1,(R0)+  
MOV R2,(R0)+  
MOV R3,(R0)+  
MOV R4,(R0)+  
MOV R5,(R0)+  
MOV SP,(R0)+  
MOV SUPDR0,R0  
MOV #TAG4$,$LPADR  
MOV $LPADR,$LPERR  
TAG4$: MOV #SUPDR6+2,R0  
MOV -(R0),SP  
MOV -(R0),R5  
MOV -(R0),R4  
MOV -(R0),R3  
MOV -(R0),R2  
MOV -(R0),R1  
MOV -(R0),R0  
SUPERVISOR ;ENTER SUPERVISOR MODE  
MOV #SUPSTK,SSP  
CACHOFF ;TURN CACHE OFF  
CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S  
CACHON ;TURN CACHE ON  
KERNEL ;ENTER KERNEL MODE  
SCOPE  
POP $LPADR,$LPERR  
RETURN
```

```

4809 026602          SUPD03: MAP      BANK
4810 026616 004737 055570 SUPD04: CALL    GETDIS
4811 026622          PUSH    $LPERR,$LPADR
4812 026632 010037 002152 MOV     R0,SUPDRO
4813 026636 012700 002154 MOV     #SUPDR1,R0
4814 026642 010120          MOV     R1,(R0)+
4815 026644 010220          MOV     R2,(R0)+
4816 026646 010320          MOV     R3,(R0)+
4817 026650 010420          MOV     R4,(R0)+
4818 026652 010520          MOV     R5,(R0)+
4819 026654 010620          MOV     SP,(R0)+
4820 026656 013700 002152 MOV     SUPDRO,R0
4821 026662 012737 026676 002562 MOV     #TBG4$, $LPADR
4822 026670 013737 002562 002564 MOV     $LPADR,$LPERR
4823 026676 012700 002170 TBG4$: MOV     #SUPDR6+2,R0
4824 026702 014006          MOV     -(R0),SP
4825 026704 014005          MOV     -(R0),R5
4826 026706 014004          MOV     -(R0),R4
4827 026710 014003          MOV     -(R0),R3
4828 026712 014002          MOV     -(R0),R2
4829 026714 014001          MOV     -(R0),R1
4830 026716 014000          MOV     -(R0),R0
4831 026720          TESTAREA
4832 026726 005737 002426 TST     NOSUPER
4833 026732 001403          BEQ     1$
4834 026734 012706 000700 MOV     #USESTK,USP
4835 026740 000402          BR      2$
4836 026742 012706 000740 1$: MOV     #SUPSTK,SSP
4837 026746 104424          2$: CACHOFF
4838 026750 004777 153300 CALL    @SUPDOADD
4839 026754 104423          CACHON
4840 026756 104417          KERNEL
4841 026760 000004          SCOPE
4842 026762          POP     $LPADR,$LPERR
4843 026772 000207          RETURN
;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
;ENTER SUPERVISOR MODE
;TURN CACHE OFF
;TURN CACHE ON
;ENTER KERNEL MODE

```

4846  
4847  
4848  
4849  
4850  
4851  
4852  
4853  
4854  
4855  
4856 026774

4857 026774 010220  
4858 026776 077102  
4859 027000 000240  
4860 027002 012401  
4861 027004 020102  
4862 027006 001402  
4863 027010 104430  
4864 027012 000240  
4865 027014 077306  
4866 027016 000207  
4867 027020

4868 027020 010220  
4869 027022 062702 000002  
4870 027026 077104  
4871 027030 000240  
4872 027032 012400  
4873 027034 020005  
4874 027036 001401  
4875 027040 104427  
4876 027042 062705 000002  
4877 027046 077307  
4878 027050 000207  
4879 027052

4880 027052 010540  
4881 027054 062705 000002  
4882 027060 077104  
4883 027062 000240  
4884 027064 162702 000002  
4885 027070 012401  
4886 027072 020102  
4887 027074 001401  
4888 027076 104430  
4889 027100 077307  
4890 027102 000207

```

.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)
*****
MTP000: SUBTST <<MTP000 BASIC DATA TEST>>
*****
*SUBTEST MTP000 BASIC DATA TEST
*****
1$: MOV R2, (R0)+ :V177640
SOB R1, MTP000 :V177642
NOP :V177644
2$: MOV (R4)+, R1 :V177646
CMP R1, R2 :V177650
BEQ 3$ :V177652
PERR02 :V177654
NOP :V177656
3$: SOB R3, 2$ :V177660
RETURN :V177662
MTP001: SUBTST <<MTP001 ADDRESS TEST>>
*****
*SUBTEST MTP001 ADDRESS TEST
*****
3$: MOV R2, (R0)+ :V177640
ADD #2, R2 :V177642
SOB R1, 3$ :V177646
NOP :V177650
1$: MOV (R4)+, R0 :V177652
CMP R0, R5 :V177654
BEQ 2$ :V177656
PERR01 :V177660
2$: ADD #2, R5 :V177662
SOB R3, 1$ :V177666
RETURN :V177672
MTP002: SUBTST <<MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)>>
*****
*SUBTEST MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
*****
3$: MOV R5, -(R0) :V177640
ADD #2, R5 :V177642
SOB R1, 3$ :V177646
NOP :V177650
1$: SUB #2, R2 :V177652
MOV (R4)+, R1 :V177656
CMP R1, R2 :V177660
BEQ 2$ :V177662
PERR02 :V177664
2$: SOB R3, 1$ :V177666
RETURN :V177670

```

4893 027104

MTPA03: SUBTST <<MTPA03 3 XOR 9 WORST CASE NOISE TEST (WRITE)>>

\*\*\*\*\*  
 :\*SUBTEST MTPA03 3 XOR 9 WORST CASE NOISE TEST (WRITE)  
 \*\*\*\*\*

4894  
 4895  
 4896  
 4897  
 4898  
 4899  
 4900 027104 010421  
 4901 027106 010421  
 4902 027110 077203  
 4903 027112 005104  
 4904 027114 052704  
 4905 027116 000401  
 4906 027120 012702 000004  
 4907 027124 077511  
 4908 027126 005104  
 4909 027130 052704  
 4910 027132 000401  
 4911 027134 012705 000100  
 4912 027140 077317  
 4913 027142 000207  
 4914  
 4915  
 4916 027144

```

:R1 = ADDRESS
:R2 = SMALL LOOP CONSTANT
:R3 = NUM OF ADD TO TEST (LARGE LOOP)
:R4 = GOOD DATA
:R5 = MEDIUM LOOP CONSTANT
.ENABL LSB
1$: MOV R4,(R1)+ ;V177640
MOV R4,(R1)+ ;V177642
SOB R2,1$ ;V177644
COM R4 ;V177646
BIS (PC)+,R4 ;V177650
WARN2: 401 ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #4,R2 ;V177654
SOB R5,1$ ;V177660
COM R4 ;V177662
BIS (PC)+,R4 ;V177664
WARN3: 401 ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #64,R5 ;V177670
SOB R3,1$ ;V177674
RETURN ;V177676
.DSABL LSB
  
```

MTPB03: SUBTST <<MTPB03 3 XOR 9 WORST CASE NOISE TEST (READ)>>

\*\*\*\*\*  
 :\*SUBTEST MTPB03 3 XOR 9 WORST CASE NOISE TEST (READ)  
 \*\*\*\*\*

4917  
 4918 027144 000137 027204  
 4919 027150 077203  
 4920 027152 005104  
 4921 027154 052704  
 4922 027156 000401  
 4923 027160 012702 000004  
 4924 027164 077511  
 4925 027166 005104  
 4926 027170 052704  
 4927 027172 000401  
 4928 027174 012705 000100  
 4929 027200 077317  
 4930 027202 000207  
 4931

```

.ENABL LSB
1$: JMP @#MTPC03 ;V177640 GO TO V172360
SOB R2,1$ ;V177644
COM R4 ;V177646
BIS (PC)+,R4 ;V177650
WARN4: 401 ;V177652 WARNING LOCATION IS MODIFIED BFFORE LOADING
MOV #4,R2 ;V177654
SOB R5,1$ ;V177660
COM R4 ;V177662
BIS (PC)+,R4 ;V177664
WARN5: 401 ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #64,R5 ;V177670
SOB R3,1$ ;V177674
RETURN ;V177676
.DSABL LSB
  
```

4934 027204

```

MTPC03: SUBTST <<MTPC03 TEST DATA SUBPROGRAM>>
:*****
:*SUBTEST MTPC03 TEST DATA SUBPROGRAM
:*****
      CMP R4,(R1)+ ;V172360
      BEQ 1$ ;V172362
      PERRO3 ;V172364
1$: COM -(R1) ;V172366
      COM (R1) ;V172370
      JMP @#MTPD03 ;V172372 GO TO V172260
    
```

4935 027204 020421  
4936 027206 001401  
4937 027210 104431  
4938 027212 005141  
4939 027214 005111  
4940 027216 000137 027222  
4941  
4942 027222

```

MTPD03: SUBTST <<MTPD03 TEST DATA SUBSUBPROGRAM>>
:*****
:*SUBTEST MTPD03 TEST DATA SUBSUBPROGRAM
:*****
      CMP R4,(R1)+ ;V172260
      BEQ 1$ ;V172262
      PERRO3 ;V172264
1$: COM (PC)+ ;V172266
      0 ;V172270
      BNE MTPC03 ;V172272 GO TO V172360
      JMP @#MTPB03+4 ;V172274 GO TO V177644
    
```

4943 027222 020421  
4944 027224 001401  
4945 027226 104431  
4946 027230 005127  
4947 027232 000000  
4948 027234 001363  
4949 027236 000137 027150



4952 027242

MTPA04: SUBTST <<MTPA04 ROTATING ZEROS TEST>>

\*\*\*\*\*  
: \*SUBTEST MTPA04 ROTATING ZEROS TEST  
\*\*\*\*\*

4953 027242 012705 000010  
4954 027246 010504  
4955 027250 000241  
4956 027252 000137 027276  
4957 027256 016004 177776  
4958 027262 103402  
4959 027264 020204  
4960 027266 001401  
4961 027270 104432  
4962 027272 077115  
4963 027274 000207  
4964

1\$: MOV #8.,R5 ;V177640  
MOV R5,R4 ;V177644  
CLC ;V177646  
JMP @#MTPB04 ;V177650  
MOV -2(R0),R4 ;V177654  
BCS 2\$ ;V177660  
CMP R2,R4 ;V177662  
BEQ 3\$ ;V177664  
2\$: PERR04 ;V177666  
3\$: SOB R1,1\$ ;V177670  
RETURN ;V177672

4965 027276

MTPB04: SUBTST <<MTPB04 SUBR ROTATING BIT>>

\*\*\*\*\*  
: \*SUBTEST MTPB04 SUBR ROTATING BIT  
\*\*\*\*\*

4966 027276 106110  
4967 027300 077502  
4968 027302 106120  
4969 027304 106110  
4970 027306 077402  
4971 027310 106120  
4972 027312 000137 027256  
4973  
4974 027316

1\$: ROLB (R0) ;V172360  
SOB R5,1\$ ;V172362  
ROLB (R0)+ ;V172364  
2\$: ROLB (R0) ;V172366  
SOB R4,2\$ ;V172370  
ROLB (R0)+ ;V172372  
JMP @#MTPA04+14 ;V172374

MTP005: SUBTST <<MTP005 ROTATION ONES TEST>>

\*\*\*\*\*  
: \*SUBTEST MTP005 ROTATION ONES TEST  
\*\*\*\*\*

4975 027316 012705 000010  
4976 027322 010504  
4977 027324 000261  
4978 027326 000137 027276  
4979 027332 016004 177776  
4980 027336 103002  
4981 027340 020204  
4982 027342 001401  
4983 027344 104432  
4984 027346 077115  
4985 027350 000207

1\$: MOV #8.,R5 ;V177640  
MOV R5,R4 ;V177644  
SEC ;V177646  
JMP @#MTPB04 ;V177650  
MOV -2(R0),R4 ;V177654  
BCC 2\$ ;V177660  
CMP R2,R4 ;V177662  
BEQ 3\$ ;V177664  
2\$: PERR04 ;V177666  
3\$: SOB R1,1\$ ;V177670  
RETURN ;V177672

IF THIS HAPPENS THE GOOD & BAD MATCH

4988 027352

MTP006: SUBTST <<MTP006 INITIAL DATA TEST>>  
 :\*\*\*\*\*  
 :\*SUBTEST MTP006 INITIAL DATA TEST  
 :\*\*\*\*\*

```

4989
4990
4991 027352 012737 000001 002234      MOV    #1,DATBUF      ;SET THE FIRST TEST BIT
4992 027360 005037 002236              CLR    DATBUF+2      ;CLEAR 2ND WORD
4993 027364 013771 002234 000000 1$:  MOV    DATBUF,@(R1)   ;WRITE TEST WORD 1
4994 027372 013771 002236 000002      MOV    DATBUF+2,@2(R1);AND TEST WORD 2
4995 027400 017102 000000              MOV    @(R1),R2
4996 027404 023702 002234              CMP    DATBUF,R2     ;NOW READ THEM
4997 027410 001401              BEQ    2$            ;BR IF FIRST 16 OK
4998 027412 104433              PERR07              ;ERROR TRAP
4999
5000 027414 017102 000002 2$:  MOV    @2(R1),R2
5001 027420 023702 002236              CMP    DATBUF+2,R2   ;NOW READ SECOND WORD
5002 027424 001401              BEQ    3$            ;BR IF OK
5003 027426 104434              PERR10              ;ERROR TRAP
5004
5005 027430 005737 002236 3$:  TST    DATBUF+2      ;HAS LAST BIT BEEN TESTED ?
5006 027434 100405              BMI    4$            ;MINUS MEANS BIT 31
5007 027436              DLEFT  DATBUF        ;NO, SHIFT TEST BIT LEFT
5008 027446 000746              BR     1$            ;GO WRITE NEW TEST DATA
5009
5010 027450 012737 177776 002234 4$:  MOV    #177776,DATBUF ;PUT A 0 IN BIT 0
5011 027456 012737 177777 002236      MOV    #-1,DATBUF+2  ;AND 1'S IN ALL OTHERS
5012 027464 013771 002234 000000 5$:  MOV    DATBUF,@(R1)   ;WRITE THE DATA
5013 027472 013771 002236 000002      MOV    DATBUF+2,@2(R1);2 WORDS WORTH
5014 027500 017102 000000              MOV    @(R1),R2
5015 027504 023702 002234              CMP    DATBUF,R2     ;NOW READ FIRST WORD
5016 027510 001401              BEQ    6$            ;BR IF OK
5017 027512 104433              PERR07
5018
5019 027514 017102 000002 6$:  MOV    @2(R1),R2
5020 027520 023702 002236              CMP    DATBUF+2,R2   ;NOW, READ SECOND WORD
5021 027524 001401              BEQ    7$            ;BR IF OK
5022 027526 104434              PERR10
5023
5024 027530 005737 002236 7$:  TST    DATBUF+2      ;TESTED BIT 31 YET?
5025 027534 100005              BPL    8$            ;BR IF YES, WE'RE DONE
5026 027536              DLEFT  DATBUF
5027 027546 000746              BR     5$            ;KEEP GOING
5028 027550 000207 8$:  RETURN
  
```

5031 027552

MTP007: SUBTST <<MTP007 ADDRESS BIT TEST>>

\*\*\*\*\*  
:SUBTEST MTP007 ADDRESS BIT TEST  
\*\*\*\*\*

5032

5033

5034

5035

5036 027552 111100

5037 027554 105700

5038 027556 001401

5039 027560 104435

5040

5041 027562 105111

5042 027564 111100

5043 027566 105700

5044 027570 001001

5045 027572 104436

5046

5047 027574 040201

5048 027576 006302

5049 027600 050201

5050 027602 011100

5051 027604 005700

5052 027606 001401

5053 027610 104437

5054

5055 027612 005111

5056 027614 011100

5057 027616 005700

5058 027620 001001

5059 027622 104440

5060

5061 027624 022702 100000

5062 027630 001407

5063 027632 022702 010000

5064 027636 001356

5065 027640 006302

5066 027642 012701 160000

5067 027646 000752

5068 027650 000207

```

:
: 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.
:
: MOVB (R1),R0
: TSTB R0 ;READ AND COMPARE FOR ZEROS
: BEQ 1$ ;BR IF OK
: PERR11
:
1$: COMB (R1) ;COMPLEMENT THE BYTE
: MOVB (R1),R0
: TSTB R0 ;READ FOR NON ZEROS
: BNE 2$ ;BR IF OK
: PERR12
:
2$: BIC R2,R1 ;MASK OFF THE ASSERTED BIT
: ASL R2 ;SHIFT R2 FOR NEXT BIT
: BIS R2,R1 ;SET THE NEW BIT INTO R1
: MOV (R1),R0
: TST R0 ;READ THE NEW ADDRESS
: BEQ 3$ ;READ FOR ZEROS
: PERR13
:
3$: COM (R1) ;COMPL THE WORD
: MOV (R1),R0
: TST R0 ;READ IT AGAIN
: BNE 4$
: PERR14
:
4$: CMP #100000,R2
: BEQ 5$
: CMP #10000,R2 ;CHECK FOR MSB IN 4K BANK
: BNE 2$ ;NOT LAST BIT, BRANCH
: ASL R2
: MOV #160000,R1
: BR 2$
:
5$: RETURN

```

5071 027652

```
MTP010: SUBTST <<MTP010      BYTE ADDRESSING TEST>>
:*****
:*SUBTEST      MTP010  BYTE ADDRESSING TEST
:*****
```

```
5072      ;TEST 3 THIS TEST CHECKS FOR PROPER
5073      ;      BYTE ADDRESSING WITH ECC DISABLED
5074 027652 010402      MOV      R4,R2      ;R4 HAS LOWEST ADDRESS
5075 027654 010403      MOV      R4,R3      ;PUT IT IN R3 ALSO
5076 027656 062702 000004  ADD      #4,R2      ;POINT R2 TO LAST BYTE +1
5077 027662 012713 177777  MOV      #-1,(R3)   ;WRITE ALL ONES IN
5078 027666 012763 177777 000002  MOV      #-1,2(R3)  ;THE 4 TEST BYTES
5079 027674 105013      1$:      CLR      (R3)      ;CLEAR A BYTE
5080 027676 010401      MOV      R4,R1      ;INITIALIZE R1 FOR EACH PASS
5081 027700 020201      2$:      CMP      R2,R1      ;IF EQUAL, JUST READ LAST BYTE
5082 027702 001420      BEQ      6$         ;BR IF EQUAL
5083 027704 020301      CMP      R3,R1      ;IS THIS THE BYTE OF ZEROS
5084 027706 001007      BNE      4$         ;BR IF NOT
5085 027710 111100      MOV      (R1),R0
5086      ;WARNING IF YOU OPTOMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
5087 027712 022700 000000  CMP      #0,R0      ;IT IS, COMPARE FOR ZEROS
5088 027716 001401      BEQ      3$
5089 027720 104435      PERR11
5090
5091 027722 005201      3$:      INC      R1          ;NEXT BYTE
5092 027724 000765      BR      2$          ;RETURN
5093 027726 111100      4$:      MOV      (R1),R0
5094 027730 122700 177777  CMP      #-1,R0     ;ITS NOT THE BYTE OF 0'S, READ 1'S
5095 027734 001401      BEQ      5$
5096 027736 104436      PERR12
5097
5098 027740 005201      5$:      INC      R1          ;MOVE TO NEXT BYTE
5099 027742 000756      BR      2$
5100 027744 112713 177777  6$:      MOV      #-1,(R3)  ;RESTORE 1'S TO BYTE JUST TESTED
5101 027750 005203      INC      R3          ;INC TO NEXT BYTE
5102 027752 020302      CMP      R3,R2      ;WAS THAT JUST THE LAST ONE?
5103 027754 001347      BNE      1$         ;BR IF NO
5104 027756 000207      RETURN
```

5107 027760

```

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)
CLR1CSR :CLEAR 1 SELECTED CSR
TST PHEBE :TEST SPECIAL CASE INDICATOR
BEQ MTLA11 :BRANCH IF NOT SET
MOV SIPAR3,R2 :SAVE CONTENTS OF SIPAR #3
MOV SIPAR5,@#SIPAR3 :COPY CONTENTS OF #5 INTO #3
MOV R2,@#SIPAR5 :COPY CONTENTS OF #3 INTO #5
:BIG LOOP
5131 030006 012737 000001 002234 MTLA11: MOV #1,DATBUF :INITIAL DATA
5132 030014 005037 002236 : CLR DATBUF+2 :32 BITS WORTH
5133 :MEDIUM LOOP
5134 030020 012737 000001 002244 MTLB11: MOV #1,SBEMSK :INITIAL ERROR MASK
5135 030026 005037 002246 : CLR SBEMSK+2 :32 BITS WORTH
5136 :LITTLE LOOP
5137 030032 013737 002234 002240 MTLCL11: MOV DATBUF,TSTDAT :
5138 030040 013737 002236 002242 : MOV DATBUF+2,TSTDAT+2:TO SAVE ORIG DATA
5139 030046 105737 002256 : TSTB PASFLG ;COMP DATA ON SECOND PASS ONLY
5140 030052 001404 : BEQ 4$ ;BR IF FIRST PASS
5141 030054 005137 002240 : COM TSTDAT ;SECOND PASS, LOMP BOTH WORDS
5142 030060 005137 002242 : COM TSTDAT+2
5143 030064 013702 002240 4$: MOV TSTDAT,R2
5144 030070 013703 002242 : MOV TSTDAT+2,R3
5145 030074 012737 002240 002272 : MOV #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
5146 030102 004737 041662 : CALL CHKGEN ;GEN CHECKBITS ON TSTDAT
:*****
:** CREATE A SINGLE BIT ERROR **
:*****
5150 030106 013701 002244 : MOV SBEMSK,R1
5151 030112 074137 002240 : XOR R1,TSTDAT
5152 030116 013701 002246 : MOV SBEMSK+2,R1
5153 030122 074137 002242 : XOR R1,TSTDAT+2
5154 030126 013701 002362 MTLD11: MOV TESTADD,R1 :FIRST TEST ADDRESS
5155 030132 013705 002364 : MOV TESTADD+2,R5 :SECOND TEST ADDRESS
5156 030136 104471 : ECC1DIS :DISABLE ECC ON 1 SELECTED CSR
5157 030140 013711 002240 : MOV TSTDAT,(R1) :WRITE FIRST 16 BITS
5158 030144 104475 : CB1CSR :WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5159 030146 013715 002242 : MOV TSTDAT+2,(R5) :WRITE SECOND 16 BITS AND
5160 : :CHECK BITS. WE NOW HAVE CHECKBITS

```

```

5161                                     ;GENERATED ON DATBUF AND DATA WITH
5162                                     ;ONE BIT IN ERROR (AS PER SBEMSK).
5163 030152 104471                       ECC1DIS                               ;DISABLE ECC ON 1 SELECTED CSR
5164 030154 011100                       MOV      (R1),R0
5165 030156 020037 002240                 CMP      R0,TSTDAT                       ;READ THE LOW WORD (UNCORRECTED)
5166 030162 001403                       BEQ      6$                               ;BR IF OK
5167 030164 010137 002032                 MOV      R1,ADDRESS
5168 030170 104455                       PERR31
5169
5170 030172 011500                       6$:  MOV      (R5),R0
5171 030174 020037 002242                 CMP      R0,TSTDAT+2                     ;READ THE HIGH WORD (UNCORRECTED)
5172 030200 001403                       BEQ      7$                               ;BR IF OK
5173 030202 010537 002032                 MOV      R5,ADDRESS
5174 030206 104455                       PERR31
5175
5176 030210                               7$:  IF KFLAG IS FALSE
5177 030216 104426                       READCSR
5178 030220                               IF #BIT4 OFF.IN CSR OR #BIT15 OFF.IN CSR
5179 030240 104045                       ERROR      +45
5180 030242                               END; OF IF #BIT4
5181 030242                               END; OF IF KFLAG
5182 030242 005737 013076                 TST      PHEBE
5183 030246 001001                       BNE      17$
5184 030250 104512                       ERRGEN
5185 030252 104503                       17$:  CLR1CSR                               ;CLEAR 1 SELECTED CSR
5186 030254 011100                       MOV      (R1),R0
5187 030256 020002                       CMP      R0,R2
5188 030260 001401                       BEQ      8$                               ;SEE IF ITS BEEN CORRECTED
5189 030262 104456                       PERR32                               ;IT SHOULD HAVE BEEN
5190
5191 030264 104510                       8$:  TSTREAD                               ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
5192 030266 103411                       BCS      9$                               ;BR IF IT IS SET
5193 030270                               SET      HEADER                           ;ENABLE PRINTING OF ERROR HEADER INFO
5194 030276 010137 002032                 MOV      R1,ADDRESS
5195 030302 104460                       PERR34
5196 030304                               SET      HEADER                           ;ENABLE PRINTING OF ERROR HEADER INFO
5197
5198 030312 104503                       9$:  CLR1CSR                               ;CLEAR 1 SELECTED CSR
5199 030314 011500                       MOV      (R5),R0
5200 030316 020003                       CMP      R0,R3
5201 030320 001401                       BEQ      10$                              ;SEE IF ITS BEEN CORRECTED
5202 030322 104456                       PERR32                              ;BR IF OK
5203
5204 030324 104510                       10$: TSTREAD                               ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
5205 030326 103411                       BCS      11$                              ;BR IF YES
5206 030330                               SET      HEADER                           ;ENABLE PRINTING OF ERROR HEADER INFO
5207 030336 010137 002032                 MOV      R1,ADDRESS
5208 030342 104460                       PERR34
5209 030344                               SET      HEADER                           ;ENABLE PRINTING OF ERROR HEADER INFO
5210 030352 104512                       11$: ERRGEN                               ;TEST ERROR ADDRESS
5211 030354 105737 002256                 TSTB     PASFLG
5212 030360 100452                       BMI      15$
5213 030362 005737 002246                 TST      SBEMSK+2                       ;TEST FOR LAST MASK BIT
5214 030366 100405                       BMI      12$                              ;MINUS MEANS BIT 31
5215 030370                               DLEFT   SBEMSK
5216 030400 000614                       BR       MTLCL1
5217 030402                               12$: IF #SW11 SET.IN @SWR THEN GOTO 13$

```

```

5218 030412
5219 030420 005737 002236
5220 030424 100406
5221 030426
5222 030436 000137 030020
5223 030442 105737 002256      13$:
5224 030446 001004
5225 030450 105237 002256
5226 030454 000137 030006
5227 030460 052737 000200 002256 14$:
5228 030466 005002
5229 030470 005003
5230 030472 005037 002240
5231 030476 005037 002242
5232 030502 012704 000040
5233 030506 012737 003740 002274 15$:
5234 030514 074437 002274
5235 030520 006304
5236 030522 032704 020000
5237 030526 001002
5238 030530 000137 030126
5239
5240 030534 104471      16$:
5241 030536 013701 002362
5242 030542 013705 002364
5243 030546
5244 030552 104503
5245 030554 000207

IF QVFLAG IS TRUE THEN GOTO 13$
TST   DATBUF+2           ;LAST DATA BIT ?
BMI   13$                ;WHICH IS BIT 31
DLEFT DATBUF
JMP   MTLB11
TSTB  PASFLG ;FIRST OR SECOND PASS ?
BNE   14$                ;NON ZERO MEANS WE'RE DONE
INCB  PASFLG ;NOT DONE, GO DO SECOND PASS
JMP   MTLA11
BIS   #BIT7,PASFLG
CLR   R2
CLR   R3
CLR   TSTDAT
CLR   TSTDAT+2
MOV   #40,R4
MOV   #3740,CHECK
XOR   R4,CHECK
ASL   R4
BIT   #BIT13,R4
BNE   16$
JMP   MTLD11
;CLEAR OUT ANY DBE'S OR SBE'S
ECCDIS ;DISABLE ECC ON 1 SELECTED CSR
MOV   TESTADD,R1
MOV   TESTADD+2,R5
CLEAR (R1),(R5)
CLR1CSR ;CLEAR 1 SELECTED CSR
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.
CLR1CSR ;CLEAR 1 SELECTED CSR
MOV #1,DATBUF ;INITIAL DATA
CLR DATBUF+2 ;32 BITS WORTH
1$: MOV #1,SBEMSK ;INITIAL ERROR MASK
CLR CBEMSK+2 ;32 BITS WORTH
2$: MOV DATBUF,TSTDAT ;SAVE ORIGINAL DATA
MOV DATBUF+2,TSTDAT+2 ;BOTH WORDS
MOV #TSTDAT,SOURCE ;NEED ADDRESS FOR CHKGEN
CALL CHKGEN ;GENERATE CHECK BITS
MOV SBEMSK,R1
XOR R1,TSTDAT
MOV SBEMSK+2,R1
XOR R1,TSTDAT+2
MOV TESTADD,R4 ;FIRST TEST ADDRESS
MOV R4,R1 ;PUT IT IN R1 ALSO
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
MOV TSTDAT,(R1) ;WRITE 16 BITS
CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
ADD R5,R1 ;INDEX UP TO SECOND WORD
MOV TSTDAT+2,(R1) ;WRITE HIGH WORD+CHECKBITS
CLR1CSR ;CLEAR 1 SELECTED CSR
;IT'S DANGEROUS IF WE DON'T
40V #SBEMSK,R2 ;ADDRESS OF ERROR MASK
SUB R5,R1 ;RETURN TO FIRST WORD
3$: MOVB #-1,(R1) ;WRITE A BYTE OF 1'S
TST KFLAG ;IS THIS MF11S-K
BEQ 4$ ;BRANCH IF NOT - IT'S MS11-M
BITB #-1,(R2) ;DID THIS BYTE HAVE THE BAD BIT IN IT?
BEQ 6$ ;NO - BRANCH
4$: TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
BCC 5$ ;NO - SKIP
SET HEADEP ;ENABLE PRINTING OF ERROR HEADER INFO
MOV R1,ADDRESS
ERROR +1
SET HEADER ;ENABLE PRINTING OF ERROR HEADER INFO
5$: MOVB (R1),R0
CMPB #-1,R0 ;CHECK DATA
BEQ 7$ ;BR IF OK
PERR33
6$: TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
;READ THE BYTE
;SBE ERROR BIT ONLY SET ?
;SHOULD BE SET, BR IF OK
BCS 5$ ;ENABLE PRINTING OF ERROR HEADER INFO
SET HEADER
MOV R1,ADDRESS
PERR34
SET HEADER ;ENABLE PRINTING OF ERROR HEADER INFO
5300
7$: BITB #-1,(R2) ;CHECK FOR LAST BYTE

```

5249  
5250  
5251 030556 104503  
5252 030560 012737 000001 002234  
5253 030566 005037 002236  
5254 030572 012737 000001 002244 1\$:  
5255 030600 005037 002246  
5256 030604 013737 002234 002240 2\$:  
5257 030612 013737 002236 002242  
5258 030620 012737 002240 002272  
5259 030626 004737 041662  
5260 030632 013701 002244  
5261 030636 074137 002240  
5262 030642 013701 002246  
5263 030646 074137 002242  
5264 030652 013704 002362  
5265 030656 010401  
5266 030660 104471  
5267 030662 013711 002240  
5268 030666 104475  
5269 030670 060501  
5270 030672 013711 002242  
5271 030676 104503  
5272  
5273 030700 012702 002244  
5274 030704 160501  
5275 030706 112711 177777 3\$:  
5276 030712 005737 002500  
5277 030716 001403  
5278 030720 132712 177777  
5279 030724 001420  
5280 030726 104510 4\$:  
5281 030730 103011  
5282 030732  
5283 030740 010137 002032  
5284 030744 104017  
5285 030746  
5286  
5287 030754 111100 5\$:  
5288 030756 122700 177777  
5289 030762 001414  
5290 030764 104457  
5291  
5292 030766 104510 6\$:  
5293  
5294  
5295 030770 103771  
5296 030772  
5297 031000 010137 002032  
5298 031004 104460  
5299 031006  
5300  
5301 031014 132712 177777 7\$:



```

5302 031020 001012      BNE      8$      ;
5303 031022 005202      INC      R2
5304 031024 005201      INC      R1      ;MOVE TO NEXT BYTE
5305 031026 013704 002362  MOV     TESTADD,R4 ;FIRST TEST ADDRESS
5306 031032 032701 000002  BIT     #2,R1     ;TEST FOR LOWER WORD
5307 031036 001723      BEQ     3$      ;BR IF IT'S LOW 16 BITS
5308 031040 062704 000007  ADD     #2,R4     ;ADJUST POINTER FOR ERROR REPT.
5309 031044 000720      BR      3$
5310 031046 005757 002246  8$:    TST     SBEMSK+2 ;LAST ERROR BIT ?
5311 031052 100405      BMI     9$      ;MINUS MEANS BIT 31
5312 031054      DLEFT  SBEMSK
5313 031064 000647      BR      2$
5314 031066      9$:    IF #SW11 SET.IN @SWR THEN GOTO 10$
5315 031076      IF QVFLAG IS TRUE THEN GOTO 10$
5316 031104 005737 002236  TST     DATBUF+2 ;LAST DATA BIT?
5317 031110 100405      BMI     10$     ;MINUS = BIT 31
5318 031112      DLEFT  DATBUF
5319 031122 000623      BR      1$
5320      ;CLEAR OUT ANY DBE'S OR SBE'S
5321 031124 104471 002362  10$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5322 031126 013701      MOV     TESTADD,R1
5323 031132 005011      CLR     (R1)
5324 031134 060501      ADD     R5,R1
5325 031136 005011      CLR     (R1)
5326 031140 104503      CLR1CSR ;CLEAR 1 SELECTED CSR
5327 031142 000207      RETURN

```

5330 031144

```

MTP013: SUBTST <<MTP013          CREATE DOUBLE BIT ERROR TEST>>
:*****
:*SUBTEST          MTP013  CREATE DOUBLE BIT ERROR TEST
:*****
          ;DOUBLE BIT ERROR FORCE TO CHECK DOUBLE ERROR LOGIC
          ;CLEAR 1 SELECTED CSR
          CLR1CSR
          MOV      #TESTADD,R1
1$:      CLR      DATBUF          ;MAKE INITIAL DATA
          CLR      DATBUF+2      ;ALL ZEROS
2$:      MOV      #1,SBEMSK      ;INITIAL SINGLE ERROR MASK
          CLR      SBEMSK+2      ;SECOND WORD
3$:      MOV      #1,DBEMSK      ;INITIAL DOUBLE ERROR MASK
          CLR      DBEMSK+2      ;32 BITS HERE ALSO
4$:      MOV      DATBUF,TSTDAT
          MOV      DATBUF+2,TSTDAT+2
          TSTB    PASFLG ;NO COMPLEMENTING FIRST PASS
          BEQ     5$
          COM     TSTDAT          ;COMP FIRST WORD
          COM     TSTDAT+2        ;SECOND WORD
5$:      CLR1CSR
          CMP     SBEMSK,DBEMSK  ;CAN'T HAVE THE SAME ERROR BIT SET
          BNE    6$              ;IN BOTH MASKS
          CMP     SBEMSK+2,DBEMSK+2;COULD BE EQUAL IN SECOND WORD
          BEQ    13$             ;GO MAKE THEM NOT EQUAL
6$:      MOV      #TSTDAT,SOURCE ;SOURCE ADDRESS FOR CHKGEN
          CALL   CHKGEN          ;GO GENERATE CHECK BITS
          MOV     SBEMSK,R2
          XOR    R2,TSTDAT
          MOV     SBEMSK+2,R2
          XOR    R2,TSTDAT+2
          MOV     DBEMSK,R2
          XOR    R2,TSTDAT
          MOV     DBEMSK+2,R2
          XOR    R2,TSTDAT+2
16$:     ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
          MOV     TSTDAT,@(R1)+ ;WRITE 16 BITS
          CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
          MOV     TSTDAT+2,@(R1) ;WRITE HIGH WORD
          CLR1CSR ;CLEAR 1 SELECTED CSR
          SUB     #2,R1          ;ADJUST TEST ADDRESS
          TST    @(R1)          ;READ THE LOCATION
          WAS1DBE ;WAS THERE ANY DOUBLE BIT ERRORS ON 1 SELECTED CSR
          BCS    9$              ;IT SHOULD BE SET
          SET    HEADER
          MOV     (R1),ADDRESS
          ERROR  +30
          SET    HEADER
    
```

5331  
5332 031144 104503  
5333 031146 012701 002362  
5334 031152 005037 002234  
5335 031156 005037 002236  
5336 031162 012737 000001 002244  
5337 031170 005037 002246  
5338 031174 012737 000001 002250  
5339 031202 005037 002252  
5340 031206 013737 002234 002240  
5341 031214 013737 002236 002242  
5342 031222 105737 002256  
5343 031226 001404  
5344 031230 005137 002240  
5345 031234 005137 002242  
5346 031240 104503  
5347 031242 023737 002244 002250  
5348 031250 001004  
5349 031252 023737 002246 002252  
5350 031260 001460  
5351 031262 012737 002240 002272  
5352 031270 004737 041662  
5353 031274 013702 002244  
5354 031300 074237 002240  
5355 031304 013702 002246  
5356 031310 074237 002242  
5357 031314 013702 002250  
5358 031320 074237 002240  
5359 031324 013702 002252  
5360 031330 074237 002242  
5361 031334 104471  
5362 031336 013731 002240  
5363 031342 104475  
5364 031344 013771 002242 000000  
5365 031352 104503  
5366 031354 162701 000002  
5367 031360 005771 000000  
5368 031364 104501  
5369 031366 103411  
5370 031370  
5371 031376 011137 002032  
5372 031402 104030  
5373 031404

```

5376 031412 104512          9$:  ERRGEN
5377 031414 105737 002256    TSTB  PASFLG
5378 031420 100452          BMI    14$
5379 031422 005737 002252    13$: TST  DBEMSK+2      ;CHECK MASK FOR LAST BIT
5380 031426 100405          BMI    10$           ;MINUS = BIT31
5381 031430          DLEFT DBEMSK
5382 031440 000662          BR    4$
5383 031442          10$:  IF #SW11 SET.IN @SWR THEN GOTO 11$
5384 031452          IF QVFLAG IS TRUE THEN GOTO 11$
5385 031460 005737 002246    TST  SBEMSK+2      ;CHECK SINGLE ERROR MASK TOO
5386 031464 100405          BMI    11$           ;BR IF DONE
5387 031466          DLEFT SBEMSK
5388 031476 000636          BR    3$
5389 031500 105737 002256    11$: TSTB  PASFLG  ;FIRST PASS
5390 031504 001003          BNE   12$           ;NON ZERO MEANS WE'RE DONE
5391 031506 105237 002256    INCB  PASFLG  ;FIRST PASS, NOT DONE
5392          ;CLEAR OUT ANY DBE'S OR SBE'S
5393 031512 000617          BR    1$           ;KEEP GOING
5394 031514 052737 000200 002256 12$: BIS  #BIT7,PASFLG  ;SET UP FOR CHECK BIT TEST
5395 031522 005037 002240    CLR  TSTDAT
5396 031526 005037 002242    CLR  TSTDAT+2
5397 031532 012737 000040 002244    MOV  #40,SBEMSK
5398 031540 012737 000100 002250    MOV  #100,DBEMSK
5399 031546 012737 003740 002274 14$: MOV  #3740,CHECK
5400 031554 013702 002244    MOV  SBEMSK,R2
5401 031560 074237 002274    XOR  R2,CHECK
5402 031564 013702 002250    MOV  DBEMSK,R2
5403 031570 074237 002274    XOR  R2,CHECK
5404 031574 006337 002250    ASL  DBEMSK
5405 031600 032737 020000 002250    BIT  #BIT13,DBEMSK
5406 031606 001652          BEQ  16$
5407 031610 006337 002244    ASL  SBEMSK
5408 031614 032737 004000 002244    BIT  #BIT11,SBEMSK
5409 031622 001006          BNE  15$
5410 031624 013737 002244 002250    MOV  SBEMSK,DBEMSK
5411 031632 006337 002250    ASL  DBEMSK
5412 031636 000743          BR   14$
5413 031640 104471          15$: ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
5414 031642 012701 002362    MOV  #TESTADD,R1
5415 031646          CLEAR @ (R1)+,@ (R1)
5416 031654 104503          CLR1CSR      ;CLEAR 1 SELECTED CSR
5417 031656 000207          RETURN
  
```

5420 031660

MTP014: SUBTST <<MTP014 WRITE INHIBIT DURING DATIP WITH DBE TEST>>  
:\*\*\*\*\*  
:\*SUBTEST MTP014 WRITE INHIBIT DURING DATIP WITH DBE TEST  
:\*\*\*\*\*

5421  
5422  
5423

: THIS TEST CHECKS THE WRITE INHIBIT ON DOUBLE  
: BIT ERRORS DURING A DATIP OPERATION BY USE  
: OF AN 'ASRB' INSTRUCTION.

5424 031660

IF KFLAG IS TRUE THEN \$RETURN

5425  
5431

: NOTE- THIS TEST WILL ONLY BE RUN FOR MF11S-K MEMORY.

5432 031670 005037 002234  
5433 031674 005037 002236  
5434 031700 012737 000001 002244  
5435 031706 005037 002246  
5436 031712 012737 000001 002250  
5437 031720 005037 002252  
5438 031724 013737 002234 002240  
5439 031732 013737 002236 002242  
5440 031740 105737 002256  
5441 031744 001404  
5442 031746 005137 002240  
5443 031752 005137 002242  
5444 031756 104503  
5445 031760 023737 002250 002244  
5446 031766 001004  
5447 031770 023737 002252 002246  
5448 031776 001476  
5449 032000 012737 002240 002272  
5450 032006 004737 041662  
5451 032012 013701 002244  
5452 032016 074137 002240  
5453 032022 013701 002246  
5454 032026 074137 002242  
5455 032032 013701 002250  
5456 032036 074137 002240  
5457 032042 013701 002252  
5458 032046 074137 002242  
5459 032052 012701 002362  
5460 032056 104471  
5461 032060 013731 002240  
5462 032064 104475  
5463 032066 013771 002242 000000  
5464 032074 105037 002257  
5465 032100 013703 002362  
5466 032104 104503  
5467 032106 106223  
5468 032110 015100  
5469 032112 023700 002240  
5470 032116 001404  
5471 032120 017137 000000 002032  
5472 032126 104455  
5473  
5474 032130 062701 000002  
5475 032134 017100 000000  
5476 032140 023700 002242  
5477 032144 001404  
5478 032146 017137 000000 002032

1\$: CLR DATBUF ; INITIAL DATA  
CLR DATBUF+2 ; 2 WORDS WORTH  
2\$: MOV #1,SBEMSK ; INITIAL ERROR MASK  
CLR SBEMSK+2 ;  
3\$: MOV #1,DBEMSK ; DOUBLE ERROR MASK  
CLR DBEMSK+2 ; 2 WORDS  
4\$: MOV DATBUF,TSTDAT ; PRESERVE ORIG DATA  
MOV DATBUF+2,TSTDAT+2 ;  
TSTB PASFLG ; SECOND PASS YET ?  
5\$: BEQ 5\$ ; BR IF NO  
COM TSTDAT ; COMPL DATA ON SECOND PASS  
COM TSTDAT+2 ;  
5\$: CLR1CSR ; CLEAR 1 SELECTED CSR  
CMP DBEMSK,SBEMSK ; CHECK FOR SAME MASKS  
BNE 6\$ ; BR IF OK  
CMP DBEMSK+2,SBEMSK+2 ;  
6\$: BEQ 11\$ ; BR IF THEY'RE EQUAL  
MOV #TSTDAT,SOURCE ; SET UP ADDRESS FOR CHKGEN  
CALL CHKGEN ; GENERATE CHECK BITS  
MOV SBEMSK,R1  
XOR R1,TSTDAT  
MOV SBEMSK+2,R1  
XOR R1,TSTDAT+2  
MOV DBEMSK,R1  
XOR R1,TSTDAT  
MOV DBEMSK+2,R1  
XOR R1,TSTDAT+2  
7\$: MOV #TESTADD,R1 ; TEST ADDRESS  
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,@(R1) ; SECOND 16 BITS+CHECKBITS  
CLRB UPPFLG ; INDICATE LOWER WORD  
MOV TESTADD,R3 ; TEST ADDRESS  
8\$: CLR1CSR ; CLEAR 1 SELECTED CSR  
ASRB (R3)+ ; SPECIAL DATIP INSTRUCTION  
MOV @-(R1),R0  
CMP TSTDAT,R0 ; CHECK FOR UNCHANGED DATA  
BEQ 9\$ ; SHOULD BE UNCHANGED  
MOV @(R1),ADDRESS  
PERR31  
9\$: ADD #2,R1 ; POINT TO UPPER WORD  
MOV @(R1),R0  
CMP TSTDAT+2,R0 ; READ IT  
BEQ 10\$ ; BR IF UNCHANGED  
MOV @(R1),ADDRESS

```

5479 032154 104455 PERR31
5480
5481 032156 122737 000003 002257 10$: CMPB #3,UPPFLG ;LOWER WORD
5482 032164 001403 BEQ 11$ ;BR IF NO
5483 032166 105237 002257 INCB UPPFLG
5484 032172 000744 BR 8$
5485 032174 105737 002256 11$: TSTB PASFLG
5486 032200 100453 BMI 15$ ;BRANCH IF WE'RE TESTING CHECK BITS
5487 032202 005737 002252 TST DBEMSK+2 ;LAST BIT IN MASK ?
5488 032206 100405 BMI 12$ ;BR IF BIT 31
5489 032210 DLEFT DBEMSK
5490 032220 000641 BR 4$
5491 032222 12$: IF #SW11 SET.IN @SWR THEN GOTO 13$
5492 032232 IF QVFLAG IS TRUE THEN GOTO 13$
5493 032240 005737 002246 TST SBEMSK+2 ;LAST BIT IN SINGLE ERROR MASK ?
5494 032244 100405 BMI 13$ ;BR IF YES
5495 032246 DLEFT SBEMSK
5496 032256 000615 BR 3$
5497 032260 105737 002256 13$: TSTB PASFLG ;WHICH PASS
5498 032264 001004 BNE 14$ ;BR IF WE'RE DONE
5499 032266 105237 0.2256 INCB PASFLG ;INDICATE SECOND PASS COMING
5500 ;CLEAR OUT ANY DBE'S OR SBE'S
5501 032272 000137 0 1670 JMP 1$ ;GO DO IT!
5502 032276 052737 000200 002256 14$: BIS #BIT7,PASFLG
5503 032304 005037 002240 CLR TSTDAT
5504 032310 005037 002242 CLR TSTDAT+2
5505 032314 012737 000040 002244 MOV #40,SBEMSK
5506 032322 012737 000100 002250 MOV #100,DBEMSK
5507 032330 012737 003740 002274 15$: MOV #3740,CHECK
5508 032336 013702 002244 MOV SBEMSK,R2
5509 032342 074237 002274 XOR R2,CHECK
5510 032346 013702 002250 MOV DBEMSK,R2
5511 032352 074237 002274 XOR R2,CHECK
5512 032356 006337 002250 ASL DBEMSK
5513 032362 032737 020000 002250 BIT #BIT13,DBEMSK
5514 032370 001630 BEQ 7$
5515 032372 006337 002244 ASL SBEMSK
5516 032376 032737 004000 002244 BIT #BIT11,SBEMSK
5517 032404 001006 BNE 16$
5518 032406 013737 002244 002250 MOV SBEMSK,DBEMSK
5519 032414 006337 002250 ASL DBEMSK
5520 032420 000743 BR 15$
5521 032422 104471 16$: ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5522 032424 012701 002362 MOV #TESTADD,R1
5523 032430 CLEAR @R1+,@R1
5524 032436 104503 CLR1CSR ;CLEAR 1 SELECTED CSR
5525 032440 000207 RETURN

```

```

5528 032442 MTP015: SUBTST <<MTP015 WRITE INHIBIT OF BYTE WITH DBE>>
;*****
;+SUBTEST MTP015 WRITE INHIBIT OF BYTE WITH DBE
;*****
5529 ;CHECK FOR WRITE INHIBIT DURING A WRITE BYTE.
5530 ;CHECKS FOR UNCORRECTED DATA.
5531 032442 005037 002234 1$: CLR DATBUF ;INITIAL DATA
5532 032446 005037 002236 CLR DATBUF+2 ;32 BITS WORTH
5533 032452 012737 000001 002244 2$: MOV #1,SBEMSK ;SINGLE ERROR MASK
5534 032460 005037 002246 CLR SBEMSK+2 ;
5535 032464 012737 000001 002250 3$: MOV #1,DBEMSK ;DOUBLE ERROR MASK
5536 032472 005037 002252 CLR DBEMSK+2 ;
5537 032476 013737 002234 002240 4$: MOV DATBUF,TSTDAT ;PRESERVE ORIG DATA
5538 032504 013737 002236 002242 MOV DATBUF+2,TSTDAT+2
5539 032512 105737 002256 TSTB PASFLG ;WHICH PASS ?
5540 032516 001404 BEQ 5$ ;FIRST PASS, NO COMPLEMENTING
5541 032520 005137 002240 COM TSTDAT
5542 032524 005137 002242 COM TSTDAT+2 ;SECOND PASS, COMPLEMENT TSTDAT
5543 032530 104503 CLR1CSR ;CLEAR 1 SELECTED CSR
5544 032532 023737 002244 002250 5$: CMP SBEMSK,DBEMSK ;CHECK FOR SAME MASKS
5545 032540 001004 BNE 6$ ;BR IF NOT EQUAL
5546 032542 023737 002246 002252 CMP SBEMSK+2,DBEMSK+2 ;SECOND WORD ALSO
5547 032550 001474 BEQ 11$ ;BR TO MAKE THEM NOT EQUAL
5548 032552 012737 002240 002272 6$: MOV #TSTDAT,SOURCE ;ADDRESS FOR CHKGEN
5549 032560 004737 041662 CALL CHKGEN ;GO GENERATE CHECK BITS
5550 032564 013701 002244 MOV SBEMSK,R1
5551 032570 074137 002240 XOR R1,TSTDAT
5552 032574 013701 002246 MOV SBEMSK+2,R1
5553 032600 074137 002242 XOR R1,TSTDAT+2
5554 032604 013701 002250 MOV DBEMSK,R1
5555 032610 074137 002240 XOR R1,TSTDAT
5556 032614 013701 002252 MOV DBEMSK+2,R1
5557 032620 074137 002242 XOR R1,TSTDAT+2
5558 032624 012701 002242 7$: MOV #TESTADD,R1 ;TEST LOCATION
5559 032630 104471 ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5560 032632 013731 002240 MOV TSTDAT,@(R1)+ ;WRITE FIRST 16 BITS
5561 ;LOAD CSR WITH IMAGE FROM R2
5562 032636 104475 CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5563 032640 013771 002242 000000 MOV TSTDAT+2,@(R1) ;WRITE SECOND 16 BITS + CHECKBITS
5564 032646 104503 CLR1CSR ;CLEAR 1 SELECTED CSR
5565 032650 013702 002362 MOV TESTADD,R2 ;GET ADDRESS OF TEST LOC
5566 032654 010203 MOV R2,R3 ;R2 DESIGNATES FIRST BYTE
5567 032656 062703 000003 ADD #3,R3 ;R3 DESIGNATES LAST BYTE
5568 032662 112722 000360 8$: MOVB #360,(R2)+ ;TRY WRITING A BYTE
5569 032666 012701 002362 MOV #TESTADD,R1
5570 032672 017100 000000 MOV @(R1),R0
5571 032676 023700 002240 CMP TSTDAT,R0 ;CHECK FOR UNCHANGED DATA
5572 032702 001404 BEQ 9$ ;BR IF OK
5573 032704 017137 000000 002032 MOV @(R1),ADDRESS
5574 032712 104455 PERR31
5575
5576 032714 017100 000002 9$: MOV @2(R1),R0
5577 032720 023700 002242 CMP TSTDAT+2,R0 ;READ SECOND WORD
5578 032724 001404 BEQ 10$ ;BR IF UNCHANGED
5579 032726 017137 000002 002032 MOV @2(R1),ADDRESS
5580 032734 104455 PERR31
5581

```

```

5582 032736 020203          10$:  CMP      R2,R3          ;TESTED LAST BYTE ?
5583 032740 001350          BNE      8$             ;BR IF NO
5584 032742 105737 002256      11$:  TSTB    PASFLG
5585 032746 100452          BMI      15$          ;BRANCH IF TESTING CHECK BITS
5586 032750 005737 002252      TST     DBEMSK+2      ;CHECKING FOR LAST ERROR BIT
5587 032754 100405          BMI      12$          ;BR IF DONE HERE
5588 032756          DLEFT   DBEMSK
5589 032766 000643          BR      4$
5590 032770          12$:  IF #SW11 SET.IN @SWR THEN GOTO 13$
5591 033000          IF QVFLAG IS TRUE THEN GOTO 13$
5592 033006 005737 002246      TST     SBEMSK+2      ;LAST SBE MASK
5593 033012 100405          BMI      13$          ;BR IF DONE WITH THIS PASS
5594 033014          DLEFT   SBEMSK
5595 033024 000617          BR      3$
5596 033026 105737 002256      13$:  TSTB    PASFLG ;TEST PASS FLAG
5597 033032 001003          BNE      14$          ;NON ZERO MEANS WE'RE DONE
5598 033034 105237 002256      INCB    PASFLG ;NOT DONR
5599 033040 000600          BR      1$
5600 033042 052737 000200 002256 14$:  BIS     #BIT7,PASFLG
5601 033050 005037 002240          CLR     TSTDAT
5602 033054 005037 002242          CLR     TSTDAT+2
5603 033060 012737 000040 002244          MOV     #40,SBEMSK
5604 033066 012737 000100 002250          MOV     #100,DBEMSK
5605 033074 012737 003740 002274 15$:  MOV     #3740,CHECK
5606 033102 013702 002244          MOV     SBEMSK,R2
5607 033106 074237 002274          XOR     R2,CHECK
5608 033112 013702 002250          MOV     DBEMSK,R2
5609 033116 074237 002274          XOR     R2,CHECK
5610 033122 006337 002250          ASL     DBEMSK
5611 033126 032737 020000 002250          BIT     #BIT13,DBEMSK
5612 033134 001633          BEQ     7$
5613 033136 006337 002244          ASL     SBEMSK
5614 033142 032737 004000 002244          BIT     #BIT11,SBEMSK
5615 033150 001006          BNE     16$
5616 033152 013737 002244 002250          MOV     SBEMSK,DBEMSK
5617 033160 006337 002250          ASL     DBEMSK
5618 033164 000743          BR      15$
5619 033166 104471          16$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5620 033170 012701 002362          MOV     #TESTADD,R1 ;TEST LOCATION
5621 033174          CLEAR  @ (R1)+,@ (R1) ;TO ERASE ANY DBE'S FROM TESTING
5622          ;RESTORE CSR
5623 033202 104503          CLR1CSR ;CLEAR 1 SELECTED CSR
5624 033204 000207          RETURN

```

5627 033206

MTP016: SUBTST <<MTP016 WRITE INHIBIT OF WORD WITH DBE>>

\*\*\*\*\*  
: \*SUBTEST MTP016 WRITE INHIBIT OF WORD WITH DBE  
\*\*\*\*\*

5628  
5629  
5630  
5631

: DOUBLE BIT ERROR WRITE CANCEL WITH  
: WORD WRITE.  
: CHECKS WRITE INHIBIT WITH WORD WRITES TO  
: WORD WITH DOUBLE ERROR.

5632 033206 005037 002234  
5633 033212 005037 002236  
5634 033216 012737 000001 002244  
5635 033224 005037 002246  
5636 033230 012737 000001 002250  
5637 033236 005037 002252  
5638 033242 013737 002234 002240  
5639 033250 013737 002236 002242  
5640 033256 105737 002256  
5641 033262 001404  
5642 033264 005137 002240  
5643 033270 005137 002242  
5644 033274 023737 002244 002250  
5645 033302 001004  
5646 033304 023737 002246 002252  
5647 033312 001502  
5648 033314 012737 002240 002272  
5649 033322 004737 041662  
5650 033326 013701 002244  
5651 033332 074137 002240  
5652 033336 013701 002246  
5653 033342 074137 002242  
5654 033346 013701 002250  
5655 033352 074137 002240  
5656 033356 013701 002252  
5657 033362 074137 002242  
5658 033366 012701 002362  
5659 033372 104471  
5660 033374 013731 002240  
5661 033400 104475  
5662 033402 013771 002242 000000  
5663 033410 105037 002257  
5664 033414 162701 000002  
5665 033420 104503  
5666 033422 012771 177400 000000  
5667 033430 012701 002362  
5668 033434 017100 000000  
5669 033440 023700 002240  
5670 033444 001404  
5671 033446 017137 000000 002032  
5672 033454 104455  
5673  
5674 033456 062701 000002  
5675 033462 017100 000000  
5676 033466 023700 002242  
5677 033472 001404  
5678 033474 017137 000000 002032  
5679 033502 104455

T12A: CLR DATBUF ;BACKGROUND FOR DOUBLE ERRORS  
CLR DATBUF+2 ;2 WORDS WORTH  
MOV #1,SBEMSK ;SINGLE ERROR MASK  
CLR SBEMSK+2 ;  
T12B: MOV #1,DBEMSK ;DOUBLE ERROR MASK  
CLR DBEMSK+2 ;  
1\$: MOV DATBUF,TSTDAT ;DATA FOR TEST  
MOV DATBUF+2,TSTDAT+2 ;BOTH WORDS  
TSTB PASFLG ;COMP DATA ON SECOND PASS ONLY  
BEQ 2\$ ;BR IF FIRST PASS  
COM TSTDAT ;COMP FIRST WORD  
COM TSTDAT+2 ;NOW SECOND WORD  
2\$: CMP SBEMSK,DBEMSK ;CHECK FOR IDENTICAL MASKS  
BNE 3\$ ;BR IF DIFFERENT  
CMP SBEMSK+2,DBEMSK+2 ;UPPER WORD TOO  
BEQ 8\$ ;BR TO MAKE THEM NOT EQUAL  
3\$: MOV #TSTDAT,SOURCE ;NEED ADDR OF DATA FOR CHKGEN  
CALL CHKGEN ;GO GENERATE CHECK BITS  
MOV SBEMSK,R1  
XOR R1,TSTDAT  
MOV SBEMSK+2,R1  
XOR R1,TSTDAT+2  
MOV DBEMSK,R1  
XOR R1,TSTDAT  
MOV DBEMSK+2,R1  
XOR R1,TSTDAT+2  
4\$: MOV #TESTADD,R1 ;FIRST TEST ADDRESS  
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,@(R1) ;WRITE SECOND 16 BITS + CHECKBITS  
CLRB UPPFLG ;SET FOR 2 LOOPS  
SUB #2,R1 ;POINT TO LOW WORD  
5\$: CLR1CSR ;CLEAR 1 SELECTED CSR  
MOV #177400,@(R1) ;TRY WRITING LOCATION  
MOV #TESTADD,R1  
MOV @(R1),R0  
CMP TSTDAT,R0 ;CHECK FOR ORIGINAL DATA  
BEQ 6\$ ;SHOULD BE UNCHANGED  
MOV @(R1),ADDRESS  
PERR31  
6\$: ADD #2,R1  
MOV @(R1),R0  
CMP TSTDAT+2,R0 ;THIS SHOULD BE UNCHANGED ALSO  
BEQ 7\$  
MOV @(R1),ADDRESS  
PERR31



```

5682 033504 105737 002257      7$:  TSTB  UPPFLG      ;WHICH LOOP ?
5683 033510 001003              BNE    8$          ;SECOND, BR OUT
5684 033512 105237 002257      INCB  UPPFLG      ;FIRST, KEEP GOING
5685 033516 000740              BR    5$
5686 033520 105737 002256      8$:  TSTB  PASFLG
5687 033524 100454              BMI   12$
5688 033526 005737 002252      TST   DBEMSK+2    ;LAST BIT ?
5689 033532 100405              BMI   9$          ;MINUS = BIT 31
5690 033534              DLEFT DBEMSK
5691 033544 000636              BR    1$
5692 033546              9$:  IF #SW11 SET.IN @SWR THEN GOTO 10$
5693 033556              IF QVFLAG IS TRUE THEN GOTO 10$
5694 033564 005737 002246      TST   SBEMSK+2    ;LAST BIT IN THIS MASK ?
5695 033570 100406              BMI   10$        ;BR IF LAST BIT
5696 033572              DLEFT SBEMSK
5697 033602 000137 033230      JMP   T12B
5698 033606 105737 002256      10$: TSTB  PASFLG ;FIRST PASS ?
5699 033612 001004              BNE   11$        ;BR IF SECOND
5700 033614 105237 002256      INCB  PASFLG ;INDICATE SECOND PASS COMING
5701 033620 000137 033206      JMP   T12A
5702 033624 052737 000200 002256 11$: BIS   #BIT7,PASFLG
5703 033632 005037 002240      CLR   TSTDAT
5704 033636 005037 002242      CLR   TSTDAT+2
5705 033642 012737 000040 002244  MOV   #40,SBEMSK
5706 033650 012737 000100 002250  MOV   #100,DBEMSK
5707 033656 012737 003740 002274 12$: MOV   #3740,CHECK
5708 033664 013702 002244      MOV   SBEMSK,R2
5709 033670 074237 002274      XOR   R2,CHECK
5710 033674 013702 002250      MOV   DBEMSK,R2
5711 033700 074237 002274      XOR   R2,CHECK
5712 033704 006337 002250      ASL   DBEMSK
5713 033710 032737 020000 002250  BIT   #BIT13,DBEMSK
5714 033716 001623              BEQ   4$
5715 033720 006337 002244      ASL   SBEMSK
5716 033724 032737 004000 002244  BIT   #BIT11,SBEMSK
5717 033732 001006              BNE   13$
5718 033734 013737 002244 002250  MOV   SBEMSK,DBEMSK
5719 033742 006337 002250      ASL   DBEMSK
5720 033746 000743              BR    12$
5721 033750 104471              13$: ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5722 033752 012701 002362      MOV   #TESTADD,R1 ;RESTORE TEST ADDRESS
5723 033756 005031              CLR   @(R1)+ ;CLEAR ANY DBE'S FROM TEST
5724 033760 005071 000000      CLR   @(R1)
5725 033764 104503              CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
5726 033766 000207              RETURN

```

5729 033770

MTP017: SUBTST <<MTP017 HOLDING 1'S & 0'S TEST>>

\*\*\*\*\*  
: \*SUBTEST MTP017 HOLDING 1'S & 0'S TEST  
\*\*\*\*\*

```

5730          ;*(1) THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
5731          ;*   OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
5732          ;*   OF 000377 AND READING IT
5733          ;*(2) MEMORY IS WRITTEN USING A BYTE AT A TIME
5734          ;*(3) STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
5735          ;NOTE: THIS TEST WRITES BYTES & READS WORDS
5736 033770 012701 060000  MOV    #FIRST,R1
5737 033774 010104        MOV    R1,R4
5738 033776 012705 160000  MOV    #LAST+2,R5
5739 034002 012700 000377  MOV    #377,R0      ;GET THE PATTERN INTO R0
5740 034006 010003        MOV    R0,R3
5741 034010 000303        SWAB   R3
5742 034012 110021 1$:   MOVB   R0,(R1)+   ;WRITE A BYTE
5743 034014 110321        MOVB   R3,(R1)+   ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
5744 034016 020105        CMP    R1,R5      ;COMPARE TEST LOC TO TOP + 2
5745 034020 103774        BLO    1$        ;BRANCH IF LOWER
5746
5747 034022 014102 2$:   MOV    -(R1),R2
5748 034024 020002        CMP    R0,R2      ;TEST THE MEMORY TO SEE IF IT CONTAINS
5749                                     ;THE WORD STORED IN BAKPAT
5750 034026 001401        BEQ    3$
5751 034030 104446        PERR22
5752
5753 034032 020104 3$:   CMP    R1,R4      ;KEEP ON TESTING THE MEMORY UNTIL
5754 034034 101372        BHI    2$        ;R1 EQUALS THE LOWEST ADDRESS
5755 034036 000303        SWAB   R3        ;CHANGE THE DATA PATTERN
5756 034040 000300        SWAB   R0
5757 034042 001763        BEQ    1$        ;IF THE DATA PATTERN DOES NOT HAVE LOW
5758                                     ; BYTE =0 THEN FALL THRU
5759 034044 000207        RETURN

```

5762 034046

MTP020: SUBTST <<MTP020 MARCHING 1'S & 0'S IN CHECK BITS TEST>>  
:\*\*\*\*\*  
:\*SUBTEST MTP020 MARCHING 1'S & 0'S IN CHECK BITS TEST  
:\*\*\*\*\*

:\*THIS TEST IS CONCERNED ONLY WITH THE INTEGRITY  
:\*OF THE MOS RAMS THAT STORE THE CHECKBITS.

5763  
5764  
5765  
5766  
5767 034046 160201  
5768 034050 160204  
5769 034052 005711  
5770 034054 001002  
5771 034056 005714  
5772 034060 001401  
5773 034062 104453  
5774 034064 010314  
5775 034066 005011  
5776 034070 020100  
5777 034072 101365  
5778 034074 000207  
5779  
5780  
5781 034076 005711  
5782 034100 001002  
5783 034102 020314  
5784 034104 001401  
5785 034106 104452  
5786 034110 005014  
5787 034112 005011  
5788 034114 060201  
5789 034116 060204  
5790 034120 020405  
5791 034122 001365  
5792 034124 000207  
5793  
5794  
5795 034126 005711  
5796 034130 001002  
5797 034132 005714  
5798 034134 001401  
5799 034136 104453  
5800 034140 010314  
5801 034142 005011  
5802 034144 060204  
5803 034146 060201  
5804 034150 020405  
5805 034152 001365  
5806 034154 000207

:077 --> 100 DOWN  
MTPA20: SUB R2,R1 :V177640  
SUB R2,R4 :V177642  
TST (R1) :V177644 ;1ST WORD OK?  
BNE 1\$ :V177646 ;NO - SKIP  
TST (R4) :V177650 ;2ND WORD OK?  
BEQ 2\$ :V177652 ;YES - SKIP  
1\$: PERR27 :V177654 ;GOOD=000000,,000000,,077  
2\$: MOV R3,(R4) :V177656 ;2ND WORD <= 100000  
CLR (R1) :V177660 ;CLEAR 1ST WORD  
CMP R1,R0 :V177662 ;ARE WE DONE?  
BHI MTPA20 :V177664 ;BRANCH IF NOT  
RETURN :V177666

:100 --> 077 UP  
MTPB20: TST (R1) :V177640 ;1ST WORD OK?  
BNE 3\$ :V177642 ;NO - SKIP  
CMP R3,(R4) :V177644 ;2ND WORD OK?  
BEQ 4\$ :V177646 ;YES - SKIP  
3\$: PERR26 :V177650 ;GOOD=000000,,100000,,100  
4\$: CLR (R4) :V177652 ;CLEAR 2ND WORD  
CLR (R1) :V177654 ;CLEAR 1ST WORD  
ADD R2,R1 :V177656  
ADD R2,R4 :V177660  
CMP R4,R5 :V177662 ;TOP + 2 YET?  
BNE MTPB20 :V177664 ;NO - LOOP  
RETURN :V177666

:077 --> 100 UP  
MTPC20: TST (R1) :V177640 ;1ST WORD OK?  
BNE 5\$ :V177642 ;NO - SKIP  
TST (R4) :V177644 ;2ND WORD OK?  
BEQ 6\$ :V177646 ;YES - SKIP  
5\$: PERR27 :V177650 ;GOOD=000000,,000000,,077  
6\$: MOV R3,(R4) :V177652 ;WRITE 1ST WORD  
CLR (R1) :V177654 ;WRITE 2ND WORD  
ADD R2,R4 :V177656  
ADD R2,R1 :V177660  
CMP R4,R5 :V177662 ;TOP + 2 YET?  
BNE MTPC20 :V177664 ;NO - LOOP  
RETURN :V177666

5809						
5810	034156	160201	MTPD20:	:100 --> 077 DOWN		
5811	034160	160204		SUB R2,R1	:V177640	
5812	034162	020314		SUB R2,R4	:V177642	
5813	034164	001002		CMP R3,(R4)	:V177644	:2ND WORD OK?
5814	034166	005711		BNE 7\$	:V177646	:NO - SKIP
5815	034170	001401		TST (R1)	:V177650	:1ST WORD OK?
5816	034172	104452	7\$:	BEQ 8\$	:V177652	:YES - SKIP
5817	034174	005014	8\$:	PERR26	:V177654	:GOOD=000000,,100000,,100
5818	034176	005011		CLR (R4)	:V177656	:WRITE 1ST WORD
5819	034200	020100		CLR (R1)	:V177660	:WRITE 2ND WORD
5820	034202	101365		CMP R1,R0	:V177662	
5821	034204	000207		BHI MTPD20	:V177664	
5822				RETURN	:V177666	
5823						
5824	034206	005711	MTPE20:	:077 UP		
5825	034210	001002		TST (R1)	:V177640	:1ST WORD OK?
5826	034212	005714		BNE 9\$	:V177642	:NO - SKIP
5827	034214	001401		TST (R4)	:V177644	:2ND WORD OK?
5828	034216	104453	9\$:	BEQ 10\$	:V177646	:YES - SKIP
5829	034220	060201	10\$:	PERR27	:V177650	:GOOD=000000,,000000,,077
5830	034222	060204		ADD R2,R1	:V177652	
5831	034224	020405		ADJ R2,R4	:V177654	
5832	034226	001367		CMP R4,R5	:V177656	:TOP + 2 YET?
5833	034230	000207		BNE MTPE20	:V177660	:NO - LOOP
				RETURN	:V177662	

5836 034232

MTPA21: SUBTST <<MTPA21 MARCHING 1'S & 0'S PATTERN TEST>>  
 :\*\*\*\*\*  
 :\*SUBTEST MTPA21 MARCHING 1'S & 0'S PATTERN TEST  
 :\*\*\*\*\*

5837  
 5838 034232 014100  
 5839 034234 020200  
 5840 034236 001401  
 5841 034240 104443

:READ, BYTESWAP-MODIFY, READ, DOWN  
 1\$: MOV -(R1), R0 :V177640  
 CMP R2, R0 :V177642  
 BEQ 2\$ :V177644  
 PERR17 :V177646

5842  
 5843 034242 000311  
 5844 034244 011100  
 5845 034246 020300  
 5846 034250 001401  
 5847 034252 104444

2\$: SWAB (R1) :V177650  
 MOV (R1), R0 :V177652  
 CMP R3, R0 :V177654  
 BEQ 3\$ :V177656  
 PERR20 :V177660

5848  
 5849 034254 020401  
 5850 034256 001365  
 5851 034260 000207

3\$: CMP R4, R1 :V177662 :DONE?  
 BNE 1\$ :V177664 :NO - LOOP  
 RETURN :V177666 :YES - RETURN

5852  
 5853 034262  
 5854 034262 011100  
 5855 034264 020300  
 5856 034266 001401  
 5857 034270 104444

MTPB21: :READ, BYTESWAP-MODIFY, READ, UP  
 1\$: MOV (R1), R0 :V177640  
 CMP R3, R0 :V177642  
 BEQ 2\$ :V177644  
 PERR20 :V177646

5858  
 5859 034272 000311  
 5860 034274 011100  
 5861 034276 020200  
 5862 034300 001401  
 5863 034302 104443

2\$: SWAB (R1) :V177650  
 MOV (R1), R0 :V177652  
 CMP R2, R0 :V177654  
 BEQ 3\$ :V177656  
 PERR17 :V177660

5864  
 5865 034304 062701 000002  
 5866 034310 020501  
 5867 034312 001363  
 5868 034314 000207

3\$: ADD #2, R1 :V177662  
 CMP R5, R1 :V177666 :DONE?  
 BNE 1\$ :V177670 :NO - LOOP  
 RETURN :V177672 :YES - RETURN

5869  
 5870 034316  
 5871 034316 011100  
 5872 034320 020200  
 5873 034322 001401  
 5874 034324 104443

MTPC21: :READ, BYTESWAP-MODIFY, READ, UP  
 1\$: MOV (R1), R0 :V177640  
 CMP R2, R0 :V177642  
 BEQ 2\$ :V177644  
 PERR17 :V177646

5875  
 5876 034326 000311  
 5877 034330 011100  
 5878 034332 020300  
 5879 034334 001401  
 5880 034336 104444

2\$: SWAB (R1) :V177650  
 MOV (R1), R0 :V177652  
 CMP R3, R0 :V177654  
 BEQ 3\$ :V177656  
 PERR20 :V177660

5881  
 5882 034340 062701 000002  
 5883 034344 020501  
 5884 034346 001363  
 5885 034350 000207

3\$: ADD #2, R1 :V177662  
 CMP R5, R1 :V177666 :DONE?  
 BNE 1\$ :V177670 :NO - LOOP  
 RETURN :V177672 :YES - RETURN

```
5888 034352 MTPD21: ;READ,BYTESWAP-MODIFY,READ,DOWN
5889 034352 014100 1$: MOV -(R1),R0 ;V177640
5890 034354 020300 CMP R3,R0 ;V177642
5891 034356 001401 BEQ 2$ ;V177644
5892 034360 104444 PERR20 ;V177646
5893
5894 034362 000311 2$: SWAB (R1) ;V177650
5895 034364 011100 MOV (R1),R0 ;V177652
5896 034366 020200 CMP R2,R0 ;V177654
5897 034370 001401 BEQ 3$ ;V177656
5898 034372 104443 PERR17 ;V177660
5899
5900 034374 020401 3$: CMP R4,R1 ;V177662 ;DONE?
5901 034376 001365 BNE 1$ ;V177664 ;NO - LOOP
5902 034400 000207 RETURN ;V177666 ;YES - RETURN
5903
```

5906 034402  
 5907  
 5908  
 5909  
 5910  
 5911 000010  
 5912 034402  
 5913 034410  
 5914 034420  
 5915 034424  
 5916 034430  
 5917 034432  
 5918 034436  
 5919 034442  
 5920 034442  
 5921  
 5922  
 5923 034446 104423  
 5924 034450  
 5925 034456  
 5926 034462  
 5927 034470  
 5928 034504  
 5929 034516  
 5930 034524  
 5931 034526  
 5932 034532  
 5933 034534  
 5934 034536  
 5935 034542  
 5936 034542  
 5937 034546  
 5938 034552  
 5939  
 5940  
 5941 034554  
 5942  
 5943 034566  
 5944 034574  
 5945 034600 104424

```

MTP022: SUBTST <<MTP022 REFRESH & SHIFTING DIAGONAL TEST>>
:*****
:*SUBTEST MTP022 REFRESH & SHIFTING DIAGONAL TEST
:*****
:(1) WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
:(2) IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
:(3) WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
: (WITH CACHE OFF).
KDIAG=8. ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
FOR EVEN := #1 TO #2 ;FOR DATA & COMPLEMENT DATA
IF EVEN EQ #1
LET R2 := ZEROS
LET R3 := ONES
ELSE
LET R2 := ONES
LET R3 := ZEROS
END ;OF IF EVEN
FOR STRIPES := #0 TO #KDIAG-1 ;FOR THE NUMBER OF STRIPES

;WRITE LOOP
CACHON ;TURN CACHE ON
LET COUNT := STRIPES
LET R1 := #FIRST
WHILE R1 LOS #LAST
IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
IF #374 OFF. IN R1 THEN LET COUNT := COUNT - #1
IF COUNT NE #0
LET (R1) := R2
LET 2(R1) := R2
ELSE
LET (R1) := R3
LET 2(R1) := R3
END ;OF IF COUNT
LET COUNT := COUNT - #1
LET R1 := R1 + #4
END ;OF WHILE
;END OF WRITE LOOP

IF DIAGFLAG IS FALSE THEN $CALL REFRESH
;READ LOOP
LET COUNT := STRIPES
LET R1 := #FIRST
CACHOFF ;TURN CACHE OFF
  
```

5947 034602  
 5948 034610  
 5949 034624  
 5950 034636  
 5951 034644  
 5952 034646  
 5953 034652 104443  
 5954 034654  
 5955 034654  
 5956 034660  
 5957 034664 104443  
 5958 034666  
 5959 034666  
 5960 034670  
 5961 034672  
 5962 034676 104444  
 5963 034700  
 5964 034700  
 5965 034704  
 5966 034710 104444  
 5967 034712  
 5968 034712  
 5969 034712  
 5970 034716  
 5971 034722  
 5972  
 5973  
 5974 034724  
 5975 034740  
 5976 034754 000207  
 5977  
 5978 034756  
 5979  
 5980 034756 004737 035026  
 5981 034762  
 5982 034766  
 5983 035000  
 5984 035004  
 5985 035012 004737 035026  
 5986 035016  
 5987 035022  
 5988 035024 000207  
 5989 035026 012704 000640  
 5990 035032 062700 000002  
 5991 035036 005140  
 5992 035040 005120  
 5993 035042 005110  
 5994 035044 005110  
 5995 035046 077405  
 5996 035050 162700 000002  
 5997 035054 000207

```

WHILE R1 LOS #LAST
  IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
  IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
  IF COUNT NE #0
    LET R0 := (R1)
    IF R2 NE R0
      PERR17
    END ;OF IF R2
    LET R0 := 2(R1)
    IF R2 NE R0
      PERR17
    END ;OF IF R2
  ELSE
    LET R0 := (R1)
    IF R3 NE R0
      PERR20
    END ;OF IF R3
    LET R0 := 2(R1)
    IF R3 NE R0
      PERR20
    END ;OF IF R3
  END ;OF IF COUNT
  LET COUNT := COUNT - #1
  LET R1 := R1 + #4
END ;OF WHILE
;END OF READ LOOP

END ;OF FOR STRIPES
END ;OF FOR EVEN
RETURN

REFRESH:SUBTST <<SUBR REFRESH DELAY>>
:*****
:*SUBTEST SUBR REFRESH DELAY
:*****
;DISTURB EACH ROW FOR > 3.2 MS
FOR R0 := #FIRST TO #FIRST+374 BY #4
  CALL REFSUB
END ;OF FOR R0
LET R0 := #FIRST+BIT14
WHILE R0 LOS #LAST+BIT14+374
  CALL REFSUB
  LET R0 := R0 + #4
END ;OF WHILE
RETURN

REFSUB: MOV #640,R4 ;TIME FOR A > 3.2 MS LOOP
ADD #2,R0
1$: COM -(R0)
COM (R0)+
COM (R0)
COM (R0)
SOB R4,1$
SUB #2,R0
RETURN
  
```



6001 035056

```
MTPA24: SUBTST <<MTPA24 FAST GALLOPING PATTERN TEST>>
:*****
:*SUBTEST MTPA24 FAST GALLOPING PATTERN TEST
:*****
```

```
6002 :THE TOTAL TEST (INCLUDING SETUP) IS AS FOLLOWS
6003 :*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
6004 :* STORED AT LOCATION BAKPAT
6005 :*(2) TEST BEGINS AT LOWEST LOCATION BEING TESTED
6006 :* (LETS NAME IT 'A')
6007 :*(3) LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
6008 :*(4) SWAPS BYTES FOR LOCATION 'A'.
6009 :*(5) READS 'A', READS 'B'
6010 :*(6) 'B' = 'B'+400 (ADDS 64 DOUBLE WORDS TO 'B')
6011 :*(7) REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
6012 :*(8) END OF THE BANK A+2
6013 :*(9) REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
6014 :*(10) AFTER EXECUTING THE TEST DATA IS COMPLEMENTED
6015 :* AND STEPS 1-9 ARE REPEATED
6016 :REGISTERS ARE USED AS FOLLOWS
6017 :R0 TEST DATA
6018 :R1 'A'
6019 :R2 'B'
6020 :R3 BAKPAT
6021 :R4 SWAPAT
6022 :R5 LAST
```

:NOTE THE PATTERN STARTS AT MTPB24.....!!!.....'

```
6023 :UIPAR'S
6024
6025
6026
6027 035056 011100 1$: MOV (R1),R0 :V177640 :READ 'A'
6028 035060 020004 CMP R0,R4 :V177642 :CHECK 'A'
6029 035062 001401 BEQ 2$ :V177644 :BR IF OK
6030 035064 104447 PERR23 :V177646 :REPORT ERROR
6031
6032 035066 011200 2$: MOV (R2),R0 :V177650 :READ 'B'
6033 035070 020003 CMP R0,R3 :V177652 :CHECK 'B'
6034 035072 001401 BEQ 3$ :V177654 :BR IF OK
6035 035074 104450 PERR24 :V177656 :REPORT ERROR
6036
6037 035076 062702 000400 3$: ADD #400,R2 :V177660 :BUMP 'B'
6038 035102 020205 CMP R2,R5 :V177664 :AT END YET?
6039 035104 101764 BLOS 1$ :V177666 :BR IF NO
6040
6041 035106 062701 000002 ADD #2,R1 :V177670 :BUMP 'A'
6042 035112 000137 035116 JMP @MTPB24 :V177674 :GO10 V177260
```

6045 035116

```
MTPB24: SUBTST <<MTPB24 FAST GALLOP PART B>>
:*****
:*SUBTEST MTPB24 FAST GALLOP PART B
:*****
```

6046  
6047 035116 010411  
6048 035120 020105  
6049 035122 001001  
6050 035124 000207  
6051 035126 000137 035132  
6052  
6053 035132

```
:SDPAR'S
MOV R4,(R1) :V172260 :WRITE 'A'
CMP R1,R5 :V172262 :DONE?
BNE 1$ :V172264 :BR IF NO
RETURN :V172266 :YES - RETURN
1$: JMP @#MTPC24 :V172270 :GOTO V172360
```

```
MTPC24: SUBTST <<MTPC24 FAST GALLOP PART C>>
:*****
:*SUBTEST MTPC24 FAST GALLOP PART C
:*****
```

6054  
6055 035132 010102  
6056 035134 011100  
6057 035136 020004  
6058 035140 001401  
6059 035142 104447  
6060 035144 000137 035076

```
:KDPAR'S
MOV R1,R2 :V172360 :RESET 'B' <--- 'A'
MOV (R1),R0 :V172362 :READ 'A'
CMP R0,R4 :V172364 :CHECK 'A'
BEQ 1$ :V172366 :BR IF OK
PERR23 :V172370 :REPORT ERROR
1$: JMP @#MTPA24+20 :V172372 :GOTO V177660
```

```

MTP025: SUBTST <<MTP025      INTERRUPT ENABLE TEST>>
:.....
: *SUBTEST      MTP025  INTERRUPT ENABLE TEST
:.....
6064 035150 005037 002240      CLR      TSTDAT      ;GENERATE CHECKBITS ON 0,,0
6065 035154 005037 002242      CLR      TSTDAT+2
6066 035160 012737 002240 002272      MOV      #TSTDAT,SOURCE
6067 035166 004737 041662      CALL     CHKGEN
6068 035172 012737 000003 002074      MOV      #3,NOPAR      ;SETUP PARITY ACTION
6069 035200 012701 002362      MOV      #TESTADD,R1   ;FIRST TEST ADDRESS
6070 035204 012737 035244 002264      MOV      #18,PARTHERE  ;SETUP TRAP DESTINATION
6071 035212 004737 035466      CALL     MTPA25        ;WRITE DATA & CHECKBITS
6072 035216 104473      ECC1INIT      ;INITIALIZE 1 SELECTED MK11 CSR
6073 035220 005771 000000      TST      @ (R1)        ;ACCESS LOCATIONS FOR DBE TRAPS
6074 035224 005771 000002      TST      @2 (R1)
6075      ;NONE - GOOD - ACCESS FOR SBE TRAPS
6076 035230 104507      ENA1SBE      ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
6077 035232 005771 000000      TST      @ (R1)
6078 035236 005771 000002      TST      @2 (R1)
6079 035242 000404      BR        28          ;NONE - GOOD - SKIP
6080 035244 104426      18: READCSR
6081 035246      FATAL 27
6082 035254 005237 002240      28: INC      TSTDAT      ;CHECK FOR CORRECT ACTION ON SBE'S
6083 035260 004737 035414      CALL     MTPD25        ;IN ALL 4 BYTE
6084 035264 012737 000400 002240      MOV      #400,TSTDAT
6085 035272 004737 035414      CALL     MTPD25
6086 035276 005037 002240      CLR      TSTDAT
6087 035302 005237 002242      INC      TSTDAT+2
6088 035306 004737 035414      CALL     MTPD25
6089 035312 012737 000400 002242      MOV      #400,TSTDAT+2
6090 035320 004737 035414      CALL     MTPD25
6091
6092 035324 005037 002242      CLR      TSTDAT+2      ;CHECK FOR CORRECT ACTION ON DBE'S
6093 035330 012737 000003 002240      MOV      #3,TSTDAT     ;IN ALL 4 BYTES
6094 035336 004737 035436      CALL     MTPD25
6095 035342 012737 001400 002240      MOV      #1400,TSTDAT
6096 035350 004737 035436      CALL     MTPD25
6097 035354 005037 002240      CLR      TSTDAT
6098 035360 012737 000003 002242      MOV      #3,TSTDAT+2
6099 035366 004737 035436      CALL     MTPD25
6100 035372 012737 001400 002242      MOV      #1400,TSTDAT+2
6101 035400 004737 035436      CALL     MTPD25
6102 035404 104503      CLR1CSR      ;CLEAR 1 SELECTED MK11 CSR
6103 035406 005037 002074      CLR      NOPAR        ;INDICATE PARITY ACTION
6104 035412 000207      RETURN
6105
6106 035414 004737 035466      MTPD25: CALL     MTPA25        ;WRITE DATA & CHECKBITS
6107 035420 104471      ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
6108 035422 004737 035510      CALL     MTPB25        ;CHECK FOR NO TRAPS
6109 035426 104507      ENA1SBE      ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
6110 035430 004737 035550      CALL     MTPC25        ;CHECK FOR EXPECTED TRAP
6111 035434 000207      RETURN

```

```

6114 035436 004737 035466      MTP025: CALL      MTPA25      ;WRITE DATA & CHECKBITS
6115 035442 104471              ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
6116 035444 004737 035510      CALL      MTPB25      ;CHECK FOR NO TRAPS
6117                               ;ENABLE DBE TRAPS
6118 035450 104473              ECC1INIT      ;INITIALIZE 1 SELECTED MK11 CSR
6119 035452 004737 035550      CALL      MTPC25      ;CHECK FOR EXPECTED TRAP
6120 035456 104507              ENA1SBE      ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
6121 035460 004737 035550      CALL      MTPC25      ;CHECK FOR EXPECTED TRAP
6122 035464 000207              RETURN
6123
6124                               ;WRITE TSTDAT & TSTDAT+2 & CHECKBITS
6125 035466 104471      MTPA25: ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
6126 035470 013771 002240 000000  MOV      TSTDAT,@(R1) ;WRITE FIRST 16 BITS
6127 035476 104475      CB1CSR      ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
6128 035500 013771 002242 000002  MOV      TSTDAT+2,@2(R1) ;WRITE 2ND 16 BITS & CHECKBITS
6129 035506 000207              RETURN
6130
6131                               ;CHECK FOR NO TRAP OCCURING CONDITION
6132 035510 012737 035530 002264  MTPB25: MOV      #1$,PARTHERE ;SETUP TRAP DESTINATION
6133 035516 005771 000000      TST      @(R1)      ;ACCESS LOCATIONS
6134 035522 005771 000002      TST      @2(R1)
6135 035526 000207              RETURN      ;NO TRAP - GOOD - RETURN
6136
6137 035530 104426      1$:      READCSR
6138 035532 011137 002032      MOV      (R1),ADDRESS ;SAVE VIRTUAL ADDRESS
6139 035536 104024      ERROR   +24
6140 035540              SET      HEADER
6141 035546 000207              RETURN
6142
6143                               ;TRAP SHOULD OCCURE TEST
6144 035550 012737 035564 002264  MTPC25: MOV      #1$,PARTHERE ;SETUP TRAP DESTINATION
6145 035556 005771 000000      TST      @(R1)      ;ACCESS 1ST LOCATION
6146 035562 000405      BR      2$          ;NO TRAP - BAD NEWS - SKIP
6147 035564 012737 035614 002264  1$:      MOV      #3$,PARTHERE ;SETUP TRAP DESTINATION
6148 035572 005771 000002      TST      @2(R1)      ;ACCESS 2ND LOCATION
6149 035576 104426      2$:      READCSR      ;NO TRAP - BAD NEWS
6150 035600 011137 002032      MOV      (R1),ADDRESS ;SAVE VIRTUAL ADDRESS
6151 035604 104025      ERROR   +25
6152 035606              SET      HEADER
6153 035614 000207      3$:      RETURN
6154

```

6157 035616

MTPA26: SUBTST <<MTPA26 RANDOM DATA (WRITE)>>  
:\*\*\*\*\*  
: \*SUBTEST MTPA26 RANDOM DATA (WRITE)  
:\*\*\*\*\*

6158 035616 000137 035666  
6159 035622 010221  
6160 035624 010321  
6161 035626 077005  
6162 035630 000207  
6163  
6164 035632

1\$: JMP @#MTPC26 :V177640 GOTO V172360  
MOV R2,(R1)+ :V177644  
MOV R3,(R1)+ :V177646  
SOB RO,1\$ :V177650  
RETURN :V177652

MTPB26: SUBTST <<MTPB26 RANDOM DATA (READ)>>  
:\*\*\*\*\*  
: \*SUBTEST MTPB26 RANDOM DATA (READ)  
:\*\*\*\*\*

6165  
6166  
6167 035632 000137 035666  
6168 035636 020221  
6169 035640 001401  
6170 035642 104451  
6171 035644 005127  
6172 035646 000000  
6173 035650 020321  
6174 035652 001401  
6175 035654 104451  
6176 035656 005167 177764  
6177 035662 077015  
6178 035664 000207  
6179  
6180  
6181  
6182 035666

.DSABL AMA  
.ENABL LSB  
1\$: JMP @#MTPC26 :V177640 GOTO V172360  
CMP R2,(R1)+ :V177644  
BEQ 2\$ :V177646  
PERR25 :V177650  
2\$: COM (PC)+ :V177652  
RANODD: 0 :V177654 FOR ERROR REPORTING  
CMP R3,(R1)+ :V177656  
BEQ 3\$ :V177660  
PERR25 :V177662  
3\$: COM RANODD :V177664  
SOB RO,1\$ :V177670  
RETURN :V177672  
.DSABL LSB  
.ENABL AMA

MTPC26: SUBTST <<RANDOM NUMBER SUBPROGRAM>>  
:\*\*\*\*\*  
: \*SUBTEST RANDOM NUMBER SUBPROGRAM  
:\*\*\*\*\*

6183  
6184  
6185  
6186  
6187  
6188 035666 073427 000007  
6189 035672 060305  
6190 035674 005504  
6191 035676 060204  
6192 035700 062705 001057  
6193 035704 000240  
6194  
6195 035706

:CALLER MUST SETUP  
: MOV SEEDLO,R3  
: MOV SEEDHI,R2  
: MOV R3,R5  
: MOV R2,R4  
ASHC #7,R4 :V172360  
ADD R3,R5 :V172364  
ADC R4 :V172366  
ADD R2,R4 :V172370  
ADD #1057,R5 :V172372  
NOP :V172376 GOTO V172260

MTPD26: SUBTST <<RANDOM NUMBER SUBSUBPROGRAM>>  
:\*\*\*\*\*  
: \*SUBTEST RANDOM NUMBER SUBSUBPROGRAM  
:\*\*\*\*\*

6196 035706 005504  
6197 035710 062704 047401  
6198 035714 010503  
6199 035716 010402  
6200 035720 000137 035622

ADC R4 :V172260  
ADD #47401,R4 :V172262  
MOV R5,R3 :V172266  
MOV R4,R2 :V172270  
JMP @#MTPA26+4 :V172272 GOTO V177644

6203 035724

```

MTPO30: SUBTST <<MT0030      FLUSH OUT DBE'S>>
:*****
:*SUBTEST      MT0030  FLUSH OUT DBE'S
:*****
1$:  MOV      (R0),R2          ;V177640
     MOV      R2,(R0)+        ;V177642
     SOB      R1,1$           ;V177644
     RETURN                      ;V177646
    
```

6204 035724 011002  
6205 035726 010220  
6206 035730 077103  
6207 035732 000207  
6208  
6209 035734

```

MTPO31: SUBTST <<MTP031      SOB-A-LONG TEST>>
:*****
:*SUBTEST      MTP031  SOB-A-LONG TEST
:*****
    
```

6210  
6211 035734 000000  
6212 035736 077001  
6213 035740 005167 177777  
6214 035744 020167 177766  
6215 035750 001403  
6216 035752 104454  
6217 035754 010167 177756  
6218 035760 005167 177752  
6219 035764 010200  
6220  
6221 035766 010503  
6222 035770 005725  
6223 035772 010504  
6224 035774 020537 002472  
6225 036000 001001  
6226 036002 000207  
6227  
6228 036004 014344  
6229 036006 001376  
6230 036010 000752  
6231  
6232 000056

```

     .DSABL  AMA
     0
1$:  SOB      R0,1$           ;MOVE TERMINATOR
     COM      1$             ;SOB TILL R0 UNDERFLOWS
     CMP      R1,1$         ;WRITE COMPLEMENT OF SOB
     BEQ      2$           ;READ & CHECK FOR NOT "SOB R0, DOT"
     PERR30
     MOV      R1,1$
2$:  COM      1$             ;CORRECT SOB INSTRUCTION
     MOV      R2,R0         ;REINITIALIZE SOB CONSTANT
     ;UPDATE MOVE REGISTERS
     MOV      R5,R3
     TST      (R5)+         ;BUMP (SAFELY) BY 2
     MOV      R5,R4
     CMP      R5,@LINK1    ;DONE?
     BNE      3$           ;NO - SKIP
     RETURN                      ;YES
3$:  MOV      -(R3),-(R4)
     BNE      3$
     BR       1$
SOBLENGTH=.-MTP031
     .ENABL  AMA
    
```

6260 036012

MTP032: SUBTST <<MTP032 WRITE RECOVERY TEST>>  
 :\*\*\*\*\*  
 :\*SUBTEST MTP032 WRITE RECOVERY TEST  
 :\*\*\*\*\*

6261  
 6262  
 6263  
 6264  
 6265  
 6266 036012 012401  
 6267 036014 020102  
 6268 036016 001401  
 6269 036020 104430  
 6270 036022 077305  
 6271 036024 013703 002472  
 6272 036030 012400  
 6273 036032 020005  
 6274 036034 001401  
 6275 036036 104427  
 6276 036040 077305  
 6277 036042 000207

```

:THE TEST ACTUALLY EXECUTED ALREADY IN THE MEMORY UNDER TEST.
:THIS CODE INSURES THAT IT CHANGED MEMORY TO HAVE
:1/2 BANK OF #5141 WHICH IS A "COM -(R1)" INSTRUCTION AND
:1/2 BANK OF #110 WHICH IS A "JMP (R0)" INSTRUCTION.

1$:  MOV      (R4)+,R1      ;V177640      ;GET DATA FROM LOWER 1/2 BANK
     CMP      R1,R2      ;V177642      ;IS IT #5141?
     BEQ      2$          ;V177644      ;YES - SKIP
     PERRO2   ;V177646      ;NO - TAKE ERROR TRAP
2$:  SOB      R3,1$      ;V177650      ;LOOP FOR 1/2 BANK
     MOV      @#LINK1,R3 ;V177652      ;RESTORE LOOP SIZE
3$:  MOV      (R4)+,R0    ;V177656      ;GET DATA FROM UPPER 1/2 BANK
     CMP      R0,R5      ;V177660      ;IS IT #110?
     BEQ      4$          ;V177662      ;YES - SKIP
     PERRO1   ;V177664      ;NO- TAKE ERROR TRAP
4$:  SOB      R3,3$      ;V177666      ;LOOP FOR 1/2 BANK
     RETURN
  
```

6280 036044

MTP033: SUBTST <<MTP033 BRANCH GOBBLE TEST>>  
:\*\*\*\*\*  
:SUBTEST MTP033 BRANCH GOBBLE TEST  
:\*\*\*\*\*

6281  
6282 036044 000000  
6283 036046 000000  
6284 036050 000261  
6285 036052 105511  
6286 036054 100402  
6287 036056 105212  
6288 036060 000773  
6289  
6290  
6291 036062 102401  
6292 036064 104461  
6293  
6294 036066 000242  
6295 036070 105212  
6296 036072 103402  
6297 036074 102001  
6298 036076 100401  
6299 036100 104461  
6300  
6301  
6302  
6303 036102 010701  
6304 036104 162701 000036  
6305 036110 010102  
6306 036112 005202  
6307  
6308  
6309 036114 010503  
6310 036116 005725  
6311 036120 010504  
6312  
6313  
6314 036122 020537 002472  
6315 036126 001001  
6316 036130 000207  
6317  
6318  
6319 036132 014344  
6320 036134 001376  
6321 036136 005011  
6322 036140 000743  
6323 000076  
6324

```

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

1$: ;NOW CHECK FOR CORRECT CONDITION CODES
    BVS 2$ ;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)
    PERR35 ;NO - REPORT ERROR AND ABORT TEST
        ;COND CODES NOT EQUAL TO 1010

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

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

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

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

GBLENGTH=-MTP033
.ENABL AMA
    
```



6326 036142

6327 036142 010220  
6328 036144 077102  
6329 036146 000207  
6330 036150 012401  
6331 036152 020102  
6332 036154 001402  
6333 036156 104430  
6334 036160 000240  
6335 036162 077306  
6336 036164 000207

```
MTP034: SUBTST <<MTP034          SOFT ERROR - BACKGROUND PATTERN TEST>>
:*****
:*SUBTEST      MTP034  SOFT ERROR - BACKGROUND PATTERN TEST
:*****
1$:      MOV      R2,(R0)+      :V177640
        SOB      R1,MTP034     :V177642
        RETURN                       :V177644
2$:      MOV      (R4)+,R1      :V177646
        LMP      R1,R2         :V177650
        BEQ      3$           :V177652
        PERRO2                       :V177654
        NOP                        :V177656
3$:      SOB      R3,2$        :V177660
        RETURN                       :V177662
```

```

6338 036166      MTP035:SUBTST  <<MTP035      WORST CASE NOISE PARITY TEST>>
:*****
:*SUBTEST      MTP035  WORST CASE NOISE PARITY TEST
:*****
6339 036166  012737  000003  002074      MOV      #3,NOPAR      ;SET PARITY TRAPS TO RETURN TO 'PARTHERE''
6340
6341 036174      FOR R0 := #FIRST TO #LAST BY #4000
6342 036200  012737  000005  002144      MOV      #BIT2!BIT0,CSR ;SET WRITE WRONG PARITY & PAR. TRAPS INTO CSR
6343 036206  104425      LOADCSR
6344 036210  012737  036244  002264      MOV      #1$,PARTHERE
6345 036216  011010      MOV      (R0),(R0)      ;WWP TEST LOCATION
6346 036220  005710      TST      (R0)
6347 036222  010037  002032      MOV      R0,ADDRESS
6348 036226  104050      ERROR   +50
6349 036230  004737  054644      CALL   PERBNK
6350 036234  032763  002000  002626      BIT      #BIT10,CONFIG+2(R3)
6351 036242  001002      BNE     2$
6352 036244  104426      1$: READCSR
6353 036246  104512      ERRGEN
6354
6355 036250  104503      2$: CLR1CSR
6356 036252  011010      MOV      (R0),(R0)      ;CLEAR WRONG PARITY IN MEMORY
6357 036254  012737  000001  002144      MOV      #BIT0,CSR
6358 036262  104425      LOADCSR
6359 036264  012737  036276  002264      MOV      #3$,PARTHERE
6360 036272  005710      TST      (R0)
6361 036274  000405      BR      4$
6362 036276  010037  002032      3$: MOV      R0,ADDRESS
6363 036302  104050      ERROR   +50
6364 036304  004737  054644      CALL   PERBNK
6365 036310      4$: END; OF FOR
6366
6367 036322  005037  002074      CLR      NOPAR      ;RESET PARITY TRAP ACTION
6368 036326  000207      RETURN
  
```

6370  
6371  
6372 036330

.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
:*****
      MOV      R0,R4
      MOV      R1,R3
      MOV      R2,R5
      RETURN
```

6373 036330 010004  
6374 036332 010103  
6375 036334 010205  
6376 036336 000207  
6377  
6378 036340

```
FLIPWARN:SUBTST <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
:*****
:*SUBTEST      FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
:*****
```

6379 036340  
6380 036342 005237 002556  
6381 036346 042737 177774 002556  
6382 036354 022737 000001 002556  
6383 036362 001414  
6384 036364 022737 000002 002556  
6385 036372 001413  
6386 036374 022737 000003 002556  
6387 036402 001414  
6388 036404 005000  
6389 036406 013704 002554  
6390 036412 000414  
6391 036414  
6392 036420 000411  
6393 036422 012700 000401  
6394 036426 013704 002554  
6395 036432 000404  
6396 036434 012700 000401  
6397 036440 012704 000401  
6398 036444 010037 027116  
6399 036450 010037 027132  
6400 036454 010037 027156  
6401 036460 010037 027172  
6402 036464  
6403 036466 000207

```
      PUSH     R0
      INC     FLIPLOC
      BIC     #^C3,FLIPLOC
      CMP     #1,FLIPLOC
      BEQ     1$
      CMP     #2,FLIPLOC
      BEQ     2$
      CMP     #3,FLIPLOC
      BEQ     3$
      CLR     R0
      MOV     ONES,R4
      BR      4$
1$: CLEAR   R0,R4
      BR      4$
2$: MOV     #401,R0
      MOV     ONES,R4
      BR      4$
3$: MOV     #401,R0
      MOV     #401,R4
4$: MOV     R0,WARN2
      MOV     R0,WARN3
      MOV     R0,WARN4
      MOV     R0,WARN5
      POP     R0
      RETURN
```

6405 036470

```
BACKGND:SUBTST <<SUBR WRITE BACKGROUND>>
:*****
:SUBTEST SUBR WRITE BACKGROUND
:*****
```

```
6406 :WRITES DATA FROM R2
6407 036470 104415 SAVREG
6408 036472 012700 060000 MOV #FIRST,R0
6409 036476 012701 040000 MOV #SIZE,R1
6410 036502 022737 000001 003710 CMP #1,PROTYP
6411 036510 001415 BEQ WARN6B
6412 036512 012737 000207 027000 WARN6A: MOV #207,MTP000+4 ;WARNING PUTTING 'RETURN' AFTER WRITE
6413 036520 012737 026774 002254 MOV #MTP000,SUPDOADD
6414 036526 004737 026602 CALL SUPD03
6415 036532 012737 000240 027000 MOV #240,MTP000+4 ;RESTORE 'NOP' AFTER WRITE
6416 036540 104416 RESREG
6417 036542 000207 RETURN
6418 036544 WARN6B: BMOV MTP000
6419 036552 012737 000207 177644 WARN6: MOV #207,UIPAR2 ;WARNING PUTTING 'RETURN' INSTRUCTION AFTER WRITE
6420 036560 004737 026424 CALL SUPD01
6421 036564 104416 RESREG
6422 036566 000207 RETURN
```

6425 036570

PCONFIG:SUBTST <<SUBR PRINT CONFIGURATION MAP>>  
:\*\*\*\*\*  
:\*SUBTEST SUBR PRINT CONFIGURATION MAP  
:\*\*\*\*\*

```
6426 036570          PUSH   TKVEC,TKVEC+2,R0
6427 036602 010637 037070          MOV    SP,PCONFS           ;SAVE LAST GOOD SP
6428 036606 012737 037036 000060          MOV    #PCONF2,TKVEC
6429 036614 012737 000340 000062          MOV    #340,TKVEC+2
6430 036622 017700 143756          MOV    @STKB,R0           ;KILL ANY OLD INTERRUPT
6431 036626 042737 000200 177776          BIC    #BIT7,PSW         ;LOWER CPU PRIORITY TO 140
6432 036634 052777 000100 143740          BIS    #BIT6,@STKS      ;ENABLE KEYBOARD INTERRUPTS
6433
6434 036642          TYPE   MSG001
6435 036646          TYPE   MSG002
6436 036652          TYPE   MSG003
6437 036656 022737 000060 002526          CMP    #60,LASTBANK
6438 036664 002006          BGE    NOOJ
6439          ;IF FAT PAPER ON TERMINAL GOTO 1$
6440 036666          IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1
6441 036702 012700 000074          NOOJ: MOV    #60.,R0
6442 036706 010004          MOV    R0,R4
6443 036710          CLEAR  R1,R3
6444 036714          TYPE   MSG004
6445 036720 004737 037072          CALL  TCONF!G           ;GO TYPE CONFIGURATION (1ST HALF)
6446 036724 022737 000060 002526          CMP    #60,LASTBANK
6447 036732 002041          BGE    PCONF2
6448 036734          TYPE   $CRLF
6449 036740          TYPE   MSG017         ;PRINT SPACE(S)
6450 036744          TYPE   MSG011
6451 036750          TYPE   $CRLF
6452 036754          TYPE   MSG017         ;PRINT SPACE(S)
6453 036760          TYPE   MSG012
6454 036764 012701 000360          MOV    #60.*2+2,R1
6455 036770 010103          MOV    R1,R3
6456 036772 004737 037072          CALL  TCONFIG
6457 036776 000417          BR    PCONF2
6458
6459 037000 012700 000170          PCONF1: MOV   #120.,R0
6460 037004 010004          MOV    R0,R4
6461 037006          CLEAR  R1,R3
6462 037012          TYPE   MSG014         ;SPACE
6463 037016          TYPE   MSG011
6464 037022          TYPE   MSG004
6465 037026          TYPE   MSG012
6466 037032 004737 037072          CALL  TCONFIG
6467
6468 037036 013706 037070          PCONF2: MOV   PCONFS,SF   ;RESTORE STACK
6469 037042 042777 000100 143532          BIC    #BIT6,@STKS
6470 037050 117700 143530          MOVB  @STKB,R0         ;READ CHAR TO KILL FLAG
6471 037054          POP   R0,TKVEC+2,TKVEC
6472 037066 000207          RETURN
6473
6474 037070 000000          PCONFS: 0             ;STACK SAVED HERE
```

6477 037072

SUBTST <<SUBR TYPE CONFIGURATION>>

```

*****
*SUBTEST      SUBR      TYPE CONFIGURATION
*****
CALL:  MOV      #N,R0      ;N=NUMBER OF CHARACTERS
      MOV      R0,R4      ;BACKUP
      MOV      #K,R1      ;INDEX CONSTANT
      MOV      R1,R3      ;BACKUP
      CALL     TCONFIG     ;ACTUAL CALL
      RETURN     ;ONLY RETURN
*****
  
```

6478  
 6479  
 6480  
 6481  
 6482  
 6483  
 6484  
 6485  
 6486  
 6487  
 6488  
 6489

6490 037072  
 6491 037076 032761 000001 002624  
 6492 037104 001403  
 6493 037106  
 6494 037112 000402  
 6495 037114  
 6496 037120 062701 000004  
 6497 037124 077014  
 6498 037126 010400  
 6499 037130 010301

```

*****
** ERROR **
*****
TCONFIG: TYPE      MSG005
1$:  BIT      #BIT0,CONFIG(R1) ;ERROR ON THIS BANK?
     BEQ      2$
     TYPE     MSG013
     BR       3$
     ;PRINT 'X'
2$:  TYPE     MSG014
     ;PRINT SPACE
3$:  ADD      #4,R1
     ;BUMP POINTER
     SOB     R0,1$
     ;LOOP TILL DONE
     MOV     R4,R0
     MOV     R3,R1
  
```

6500  
 6501  
 6502  
 6503

6504 037132  
 6505 037136 016105 002624  
 6506 037142 006205  
 6507 037144 042705 177760  
 6508 037150 005705  
 6509 037152 001003  
 6510 037154 112705 000040  
 6511 037160 000402  
 6517 037162 062705 000060  
 6518 037166 110537 071234  
 6519 037172  
 6520 037176 062701 000004  
 6521 037202 077023  
 6522 037204 010400  
 6523 037206 010301

```

*****
** CPU'S **
*****
4$:  TYPE     MSG008
     MOV     CONFIG(R1),R5
     ASR     R5
     ;GET CPU BITS
     BIC     #^C17,R5
     ;CLEAR NON INTERESTING BITS
     TST     R5
     ;IS THERE ANYTHING THERE?
     BNE     8$
     ;YES - BRANCH.
     MOVB    #' ,R5
     ;NO - MOVE A BLANK INTO R5
     BR     9$
     ;BRANCH OVER NEXT INSTRUCTION
8$:  ADD     #60,R5
     ;MAKE ASCII
9$:  MOVB    R5,MSG015
     ;PLUG INTO MEMORY
     TYPE     MSG015
     ADD     #4,R1
     ;BUMP POINTER
     SOB     R0,4$
     ;LOOP TILL DONE
     MOV     R4,R0
     MOV     R3,R1
  
```

```

6526                                     ;*****
6527                                     ;** INTERLEAVE **
6528                                     ;*****
6529 037210                               TYPE      MSG007
6530                                     ;THIS IS AN ENTRY POINT FROM ERROR REPORTS
6531 037214 032761 010000 002626 TCFIG1: BIT      #BIT12,CONFIG+2(R1)
6532 037222 001014                               BNE      1$
6533 037224 032761 000002 002624       BIT      #BIT1,CONFIG(R1)           ;IS THERE ANY MEMORY HERE?
6534 037232 001004                               BNE      18$                    ;BRANCH IF MEMORY PRESENT.
6535 037234 112737 000040 071234       MOVB     #' ',MSG015           ;MOVE A BLANK IN TO BE PRINTED
6536 037242 000424                               BR       16$                    ;BRANCH TO TYPE ROUTINE
6537 037244 112737 000055 071234 18$:  MOVB     #'-',MSG015
6538 037252 000420                               BR       16$
6539 037254 016105 002624 1$:          MOV      CONFIG(R1),R5
6540 037260 042705 007777                               BIC     #^C170000,R5           ;GET CSR INTERLEAVE
6541 037264 000305                               SWAB    R5
6542 037266 072527 177774                               ASH     #-4,R5
6543 037272 022705 000012                               CMP     #10.,R5
6544 037276 100002                               BPL     2$
6545 037300 062705 000007                               ADD     #7,R5
6546 037304 062705 000060 2$:          ADD     #60,R5           ;MAKE ASCII
6547 037310 110537 071234       MOVB     R5,MSG015           ;PLUG INTO MEMORY
6548 037314 16$:          TYPE      MSG015
6549 037320                               IF NOTAB NE #0 THEN $RETURN
6550 037330 062701 000004       ADD     #4,R1           ;BUMP POINTER
6551 037334 077051                               SOB     R0,TCFIG1           ;LOOP TILL DONE
6552 037336 010400                               MOV     R4,R0
6553 037340 010301                               MOV     R3,R1
6554
6555                                     ;*****
6556                                     ;** MEMORY TYPE **
6557                                     ;*****
6558 .ENABL  LSB
6559 037342                               TYPE      MSG009
6560 037346 033761 002104 002624 TCFIG2: BIT      CPUBIT,CONFIG(R1)
6561 037354 001447                               BEQ     17$
6562 037356 016105 002626       MOV      CONFIG+2(R1),R5
6563 037362 000305                               SWAB    R5           ;GET MEMORY TYPE
6564 037364 042705 177770       BIC     #^C7,R5           ;CLEAR NON INTERESTING BITS
6565 037370 005705                               TST     R5
6566 037372 001440                               BEQ     17$
6567 037374 032705 000004       BIT      #BIT2,R5
6568 037400 001004                               BNE     4$
6569 037402 112737 000102 071234       MOVB     #'B',MSG015
6570 037410 000434                               BR      8$
6571 037412 032705 000002 4$:          BIT      #BIT1,R5
6572 037416 001013                               BNE     6$
6573 037420 032705 000001       BIT      #BIT0,R5
6574 037424 001004                               BNE     5$
6575 037426 112737 000115 071234       MOVB     #'M',MSG015
6576 037434 000422                               BR      8$
6577 037436 112737 000113 071234 5$:  MOVB     #'K',MSG015
6578 037444 000416                               BR      8$
6579 037446 032705 000001 6$:          BIT      #BIT0,R5
6580 037452 001004                               BNE     7$
6581 037454 112737 000114 071234       MOVB     #'L',MSG015
6582 037462 000407                               BR      8$

```

```

6583 037464 112737 000120 071234 7$:   MOVB  #'P,MSG015
6584 037472 000403          BR      8$
6585 037474 112737 000040 071234 17$:  MOVB  #' ,MSG015
6586 037502          8$:   TYPE  MSG015
6587 037506          IF NOTAB NE #0 THEN $RETURN
6588 037516 062701 000004          ADD   #4,R1          ;BUMP POINTER
6589 037522 077067          SOB   R0,TCFIG2      ;LOOP TILL DONE
6590 037524 010400          MOV   R4,R0
6591 037526 010301          MOV   R3,R1
6592          .DSABL  LSB
6593
6594          :*****
6595          **: CSR **
6596          :*****
6597 037530          TYPE  MSG016
6598 037534 112737 000040 071234 TCFIG3: MOVB  #' ,MSG015
6599 037542 016105 002624          MOV   CONFIG(R1),R5
6600 037546 032705 000002          BIT   #BIT1,R5
6601 037552 001414          BEQ   16$
6602 037554 042705 170377          BIC   #^C7400,R5
6603 037560 000305          SWAB  R5
6604 037562 022705 000012          CMP   #10.,R5
6605 037566 100002          BPL   10$
6606 037570 062705 000007          ADD   #7,R5
6607 037574 062705 000060          10$:  ADD   #60,R5          ;MAKE ASCII
6608 037600 110537 071234          MOVB  R5,MSG015      ;PLUG INTO MEMORY
6609 037604          16$:  TYPE  MSG015
6610 037610          IF NOTAB NE #0 THEN $RETURN
6611 037620 062701 000004          ADD   #4,R1          ;BUMP POINTER
6612 037624 077035          SOB   R0,TCFIG3      ;LOOP TILL DONE
6613 037626 010400          MOV   R4,R0
6614 037630 010301          MOV   R3,R1
6615
6616          :*****
6617          **: PROTECTED **
6618          :*****
6619 037632          TYPE  MSG010
6620 037636 105761 002624          11$:  TSTB  CONFIG(R1)      ;BANK PROTECTED?
6621 037642 100004          BPL   12$          ;NO - SKIP
6622 037644 112737 000120 071234          MOVB  #'P,MSG015
6623 037652 000407          BR      13$
6624 037654 032761 000100 002624 12$:  BIT   #BIT6,CONFIG(R1) ;PROTECTED REGION (
6625 037662 001406          BEQ   14$          ;NO - SKIP
6626 037664 112737 000111 071234          MOVB  #'I,MSG015
6627 037672          13$:  TYPE  MSG015
6628 037676 000402          BR      15$
6629 037700          14$:  TYPE  MSG014          ;PRINT SPACE
6630 037704 062701 000004          15$:  ADD   #4,R1          ;BUMP POINTER
6631 037710 077026          SOB   R0,11$        ;LOOP TILL DONE
6632 037712 010400          MOV   R4,R0
6633 037714 010301          MOV   R3,R1
6634 037716 000207          RETURN

```



6637  
6638  
6639  
6640  
6641  
6642  
6643  
6644  
6645  
6646  
6647  
6648  
6649  
6650  
6651  
6652  
6653  
6654  
6655  
6656  
6657  
6658  
6659  
6660  
6661  
6662  
6663  
6664  
6665  
6666

037720 022737 000001 002074  
037726 001003  
037730 005237 002070  
037734 000002  
037736 022737 000002 002074  
037744 001013  
037746  
037754 004737 040126  
037760 063716 002276  
037764 042766 000004 000002  
037772 000002  
037774 022737 000003 002074  
040002 001003  
040004 013716 002264  
040010 000002  
040012 004737 040126  
040016

```

.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'

PARITY: CMP #1,NOPAR ;COUNTING PARITY ERRORS?
BNE 1$ ;NO - SKIP
INC PARCNT ;PARITY ERROR COUNTER + 1
RTI

1$: CMP #2,NOPAR ;ACTION CODE = 2 ?
BNE 2$ ;NO - SKIP
SET ABORTFLAG ;YES
CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
ADD PCBUMP,(SP) ;UPDATE RETURN PC
BIC #BIT2,2(SP) ;SHOW FAILURE BY .NE.
RTI

2$: CMP #3,NOPAR ;ACTION CODE = 3 ?
BNE 3$ ;NO - SKIP
MOV PARTHERE,(SP)

3$: CAL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
FATAL 32
    
```

```

6669          .SBTTL TRAP NON-EXISTANT MEMORY (HOLES) HANDLER
6670          :*****
6671          :VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
6672          :CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
6673          : 1) IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
6674          : 2) TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
6675          :*****
6676
6677 040024 022737 000001 002076 NONEXIST:CMP #1,NONEM ;COUNTING NON-EXISTANT MEMORY ERRORS?
6678 040032 001011 BNE 2$ ;NO - SKIP
6679 040034 005237 002066 INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
6680 040040 022737 000001 002066 CMP #1,NEMCNT ;FIRST ERROP?
6681 040046 001002 BNE 1$ ;NO - SKIP
6682 040050 010037 002032 MOV RO,ADDRESS ;ASSUME RO CONTAINS THE ADDRESS ACCESSED
6683 040054 000002 1$: RTI
6684 040056 005237 002066 2$: INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
6685 040062 012701 000001 MOV #1,R1 ;DUMMY UP R1 FOR A FORCED SOB EXIT
6686 040066 000002 RTI
6687
6688          :*****
6689          .SBTTL TRAP TIMEOUT (TRAP TO 4) HANDLER
6690 040070 004737 040126 TIMEOUT:CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6691 040074 FATAL 6
6692          :*****
6693          .SBTTL TRAP MEMORY MANAGEMENT (TRAP TO 250) HANDLER
6694 040102 004737 040126 MMTRAP:CALL JADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6695 040106 FATAL 7
6696          .SBTTL TRAP RESERVED INSTRUCTION HANDLER
6697 040114 004737 040126 PDP1105:CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6698 040120 FATAL 5
6699
6705
6706 040126 BADSTACK:SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
        :*****
        :*SUBTEST FIND BAD SP, PC, & PSW FROM STACK
        :*****
6707 040126 010637 002024 MOV SP,BADSP
6708 040132 062737 000002 002024 ADD #2,BADSP
6709 040140 016637 000002 002020 MOV 2(SP),BADPC
6710 040146 016637 000004 002030 MOV 4(SP),BADPSW
6711 040154 000207 RETURN
  
```

```

6714          .SBTTL TRAP   KERNEL TRAP HANDLER
6715          ;*****
6716          ;KERNEL IS A TRAP THAT COMES HERE
6717          ;*****
6718
6719 040156 042766 140000 000002 $KERNEL:      BIC      #140000,2(SP)
6720 040164 000002
6721          ;*****
6722          .SBTTL TRAP   ENERGIZE TRAP HANDLER
6723 040166 052737 000001 177572 $ENERGIZE: BIS  #BIT0,MMRO
6724 040174 000002
6725          ;*****
6726          .SBTTL TRAP   DEENERGIZE TRAP HANDLER
6727 040176 042737 000001 177572 $DEENERGIZE: BIC #BIT0,MMRO
6728 040204 000002
6729          ;*****
6730          .SBTTL TRAP   CACHON TRAP HANDLER
6731 040206 005737 002514 $CACHN:  TST   CACHKN      ;IS THERE A CACHE
6732 040212 001406          BEQ    1$          ;NO - RETURN
6733 040214 013737 002514 177746          MOV   CACHKN,CONTRL      ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
6734 040222 052737 000001 177746          BIS   #BIT0,CONTRL      ;DISABLE TRAPS (BUT NOT ABORTS)
6735 040230 000002          1$:  RTI
6736          ;*****
6737          .SBTTL TRAP   CACHOFF TRAP HANDLER
6738 040232 005737 002514 $CACHF:  TST   CACHKN      ;IS THERE A CACHE?
6739 040236 001403          BEQ    1$          ;NO - RETURN
6740          ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
6741 040240 053737 002520 177746          BIS   CACHKF,CONTRL
6742 040246 000002          1$:  RTI

```

```

6745          .SBTTL TRAP LOAD CSR TRAP HANDLER
6746          :LOAD CORRECT CSR WITH DATA IN CSR
6747          :PROGRAM CSR'S ASSERT INHIBIT MODE POINTER WHEN LOADED
6748 040250    $LOADC: PUSH  R0,R1          ;SAVE REGISTERS
6749 040254    013700 002146          MOV   CSRNO,R0          ;CREATE CSR ADDRESS
6750 040260    IF INHECC IS TRUE THEN GOTO 3$ ;DON'T WANT INH. MODE POINTER ON
6751 040266    005737 002502          TST   PGMCSR          ;PROGRAM IN INTERLEAVED SPACE?
6752 040272    100007          BPL   1$              ;BRANCH IF NOT
6753 040274    113701 002503          MOVB  PGMCSR+1,R1      ;CHECK SECOND CSR
6754 040300    042701 177740          BIC   #^C37,R1       ;CLEAR UNNECESSARY BITS
6755 040304    020137 002146          CMP   R1,CSRNO       ;IS THIS THE CURRENT CSR?
6756 040310    001404          BEQ   2$              ;BRANCH IF IT IS
6757 040312    123737 002502 002146 1$: CMPB  PGMCSR,CSRNO     ;IS THIS THE CURRENT CSR?
6758 040320    001003          BNE   3$              ;BRANCH IF NOT
6759 040322    052737 020000 002144 2$: BIS   #BIT13,CSR      ;SET THE INHIBIT MODE POINTER TO 1ST 16K
6760 040330    013760 002144 172100 3$: MOV   CSR,CSRADD(PO)  ;LOAD THE CSR
6761 040336    POP   R1,R0          ;RESTORE REGISTERS
6762 040342    000002          RTI

6763          .SBTTL TRAP READ CSR TRAP HANDLER
6764          :READ THE CORRECT CSR INTO LOCATIONS CSR
6765          $READC: PUSH  R0
6766 040344    MOV   CSRNO,R0
6767 040346    013700 002146          MOV   CSRADD(R0),CSR ;READ IT
6768 040352    016037 172100 002144  POP   R0
6769 040360    RTI
6770 040362    000002

```

6772							.SBTTL	TRAP	TEST (R1) & READ CSR CAREFULLY
6773	040364						\$TSTRD:	PUSH	R0,R2,R3
6774	040372	012700	172100					MOV	#CSRADD,R0
6775	040376	063700	002146					ADD	CSRNO,R0
6776	040402	005002						CLR	R2
6777	040404	005737	002502					TST	PGMCSR
6778	040410	100007						BPL	1\$
6779	040412	113703	002503					MOVB	PGMCSR+1,R3
6780	040416	042703	000200					BIC	#BIT7,R3
6781	040422	020337	002146					CMP	R3,CSRNO
6782	040426	001404						BEQ	2\$
6783	040430	123737	002502	002146	1\$:			CMPB	PGMCSR,CSRNO
6784	040436	001002						BNE	3\$
6785	040440	012702	020000		2\$:			MOV	#BIT13,R2
6786	040444	022737	000001	003710	3\$:			CMP	#1,PROTYP
6787	040452	001403						BEQ	4\$
6788	040454	004737	040542					CALL	TSTRD1
6789	040460	000405						BR	5\$
6790	040462				4\$:			BMOV	TSTRD1
6791	040470	004737	177640					CALL	FASTCITY
6792									:CALL TO THE USER INSTRUCTION PAR'S
6793	040474								:IF SINGLE BIT ERROR ONLY - SET CARRY BIT
6794	040502				5\$:			POP	R3,R2,R0
6795	040522	052766	000001	000002				IF	#BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
6796	040530							BIS	#BIT0,2(SP)
6797	040532	042766	000001	000002				ELSE	
6798	040540							BIC	#BIT0,2(SP)
6799	040540	000002						END ;OF	IF #BIT4
6800								RTI	
6801	040542	010210					1STRD1:	MOV	R2,(R0)
6802	040544							TESTAREA	
6803	040552	105711						TSTB	(R1)
6804	040554	042737	140000	177776				BIC	#BIT15!BIT14,PSW
6805	040562	011037	002144					MOV	(R0),CSR
6806	040566	000207						RETURN	

6809					.SBTTL	TRAP	ECC DISABLE ALL CSR'S TRAP HANDLER
6810	040570	012737	000002	002144	\$ECCDIS:MOV	#BIT1,CSR	
6811	040576	004737	041314		CALL	CSROUT	
6812	040602	000002			RTI		
6813					.SBTTL	TRAP	ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
6814	040604	012737	000002	002144	\$ECC1DIS:MOV	#BIT1,CSR	
6815	040612	104425			LOADCSR		
6816	040614	000002			RTI		
6817					.SBTTL	TRAP	INITIALIZE ALL CSR'S TRAP HANDLER
6818	040616	012737	000001	002144	\$ECCINIT:MOV	#BIT0,CSR	
6819	040624	004737	041314		CALL	CSROUT	
6820	040630	000002			RTI		
6821					.SBTTL	TRAP	INITIALIZE 1 SELECTED CSR TRAP HANDLER
6822	040632	012737	000001	002144	\$ECC1INIT:MOV	#BIT0,CSR	
6823	040640	104425			LOADCSR		
6824	040642	000002			RTI		
6825					.SBTTL	TRAP	ENABLE SBE PARITY TRAPS ON ALL CSR'S
6826	040644	012737	000003	002144	\$ENASBE:MOV	#BIT0!BIT1,CSR	
6827	040652	004737	041314		CALL	CSROUT	
6828	040656	000002			RTI		
6829					.SBTTL	TRAP	ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
6830	040660	012737	000003	002144	\$ENA1SBE:MOV	#BIT0!BIT1,CSR	
6831	040666	104425			LOADCSR		
6832	040670	000002			RTI		
6833					.SBTTL	TRAP	WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
6834	040672	013737	002274	002144	\$CBCSR:MOV	CHECK,CSR	;BITS 11-5
6835	040700	052737	000006	002144	BIS	#BIT1!BIT2,CSR	;CHECK MODE
6836	040706	004737	041314		CALL	CSROUT	
6837	040712	000002			RTI		
6838					.SBTTL	TRAP	WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
6839	040714	013737	002274	002144	\$CB1CSR:MOV	CHECK,CSR	;BITS 11-5
6840	040722	052737	000006	002144	BIS	#BIT1!BIT2,CSR	;CHECK MODE
6841	040730	104425			LOADCSR		
6842	040732	000002			RTI		

6845  
 6846 040734  
 6847 040740 013701 002216  
 6848 040744 005004  
 6849 040746  
 6850 040746  
 6851 040752 006301  
 6852 040754  
 6853 040756 104426  
 6854 040760  
 6855 040770  
 6856 040774  
 6857 040776  
 6858 040776  
 6859 040776  
 6860 041002  
 6861 041020  
 6862 041020 006004  
 6863 041022  
 6864 041026  
 6865 041030 052766 000001 000002  
 6866 041036  
 6867 041040 042766 000001 000002  
 6868 041046  
 6869 041046 000002  
 6870  
 6871  
 6872 041050 104426  
 6873 041052 042766 000001 000002  
 6874 041060 032737 000020 002144  
 6875 041066 001403  
 6876 041070 052766 000001 000002  
 6877 041076 000002

```

.SBTTL TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER
$WASSBE: PUSH R1,R4
MOV TOTCSRS,R1 ;GET CSR'S BYTE
CLR R4
BEGIN LWSBE
FOR CSRNO := #0 TO #36 BY #2
ASL R1
ON.ERROR
READCSR
IF #BIT4 SET.IN CSR
SET R4
LEAVE LWSBE
END ;OF IF #BIT4
END ;OF ON.ERROR
IF R1 EQ #0 THEN LEAVE LWSBE
END ;OF FOR CSRNO
END LWSBE
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 SBE IN 1 SELECTED CSR TRAP HANDLER
;ON RETURN IF CARRY IS SET THERE WAS A SBE
$WAS1SBE: READCSR
BIC #BIT0,2(SP) ;CLR C BIT ON STACK
BIT #BIT4,CSR
BEQ 1$
BIS #BIT0,2(SP) ;SET C BIT ON STACK
1$: RTI
    
```

```

6880 .SBTTL TRAP WAS THERE A DBE ON ANY CSR TRAP HANDLER
6881 041100 $WASDBE: PUSH R1,R4
6882 041104 013701 002216 MOV TOTCSRS,R1 ;GET CSR'S BYTE
6883 041110 005004 CLR R4
6884 041112 BEGIN LWDBE
6885 041112 FOR CSRNO := #0 TO #36 BY #2
6886 041116 006301 ASL R1
6887 041120 ON.ERROR
6888 041122 104426 READCSR
6889 041124 IF #BIT15 SET.IN CSR
6890 041134 SET R4
6891 041140 LEAVE LWDBE
6892 041142 END ;OF IF #BIT4
6893 041142 END ;OF ON.ERROR
6894 041142 IF R1 EQ #0 THEN LEAVE LWDBE
6895 041146 END ;OF FOR CSRNO
6896 041164 END LWDBE
6897 041164 006004 ROR R4 ;SET C BIT FOR ERROR
6898 041166 POP R4,R1
6899 041172 ON.ERROR
6900 041174 052766 000001 000002 BIS #BIT0,2(SP)
6901 041202 ELSE
6902 041204 042766 000001 000002 BIC #BIT0,2(SP)
6903 041212 END ;OF ON.ERROR
6904 041212 000002 RT.
6905 .SBTTL TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
6906 ;ON RETURN IF CARRY IS SET THERE WAS A DBE
6907 041214 104426 $WAS1DBE: READCSR
6908 041216 005737 002144 TST CSR ;DBE?
6909 041222 100004 BPL 3$ ;NO - SKIP
6910 041224 052766 000001 000002 BIS #BIT0,2(SP) ;SET C BIT ON STACK
6911 041232 000002 RTI
6912 041234 042766 000001 000002 3$: BIC #BIT0,2(SP) ;CLR C BIT ON STACK
6913 041242 000002 RTI

```



```

6916 .SBTTL TRAP CLEAR ALL ECC CSR'S TRAP HANDLER
6917 041244 $CLRCSR: CLEAR CSR
6918 041250 004737 041314 CALL CSROUT
6919 041254 000002 RTI
6920 .SBTTL TRAP CLEAR 1 SELECTED CSR TRAP HANDLER
6921 041256 $CLR1CSR: CLEAR CSR
6922 041262 104425 LOADCSR
6923 041264 000002 RTI
6924 .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
6925 :CHECKBITS ALREADY IN LOC "CSR"
6926 041266 052737 000006 002144 $CHKDIS: BIS #BIT1:BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
6927 041274 004737 041314 CALL CSROUT
6928 041300 000002 RTI
6929 .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
6930 :CHECKBITS ALREADY IN LOC "CSR"
6931 041302 052737 000006 002144 $CHK1DIS: BIS #BIT1:BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
6932 041310 104425 LOADCSR
6933 041312 000002 RTI

```

6936 041314  
 6937 041314  
 6938 041316 013701 002216  
 6939 041322  
 6940 041322  
 6941 041326 006301  
 6942 041330  
 6943 041332 104425  
 6944 041334  
 6945 041334  
 6946 041340  
 6947 041356  
 6948 041356  
 6949 041360 000207  
 6950  
 6951 041362  
 6952 041362  
 6953 041366 013701 002100  
 6954 041372 006301  
 6955 041374 006301  
 6956 041376 042761 020000 002626  
 6957 041404  
 6958 041410 000002

```

CSROUT: SUBTST <<SUBR WRITE IN ALL CSR'S>>
:*****
:*SUBTEST SUBR WRITE IN ALL CSR'S
:*****
      PUSH R1
      MOV TOTCSRS,R1 ;GET CSR'S BYTE
      BEGIN LCSROUT
      FOR CSRNO := #0 TO #36 BY #2
      ASL R1
      ON.ERROR
      LOADCSR
      END ;OF ON.ERROR
      IF R1 EQ #0 THEN LEAVE LCSROUT
      END ;OF FOR CSRNO
      END LCSROUT
      POP R1
      RETURN

$INVALID: SUBTST <<TRAP INVALIDATE BACKGROUND PATTERN>>
:*****
:*SUBTEST TRAP INVALIDATE BACKGROUND PATTERN
:*****
      PUSH RO,R1
      MOV BANK,R1
      ASL R1
      ASL R1
      BIC #BIT13,CONFIG+2(R1)
      POP R1,RO
      RTI
  
```

```

6960 041412 $ERRGEN: SUBTST<<TRAP GENERATE AND TEST ERROR ADDRESS>>
:*****
:*SUBTEST TRAP GENERATE AND TEST ERROR ADDRESS
:*****
6961 041412 PUSH R0,R1,R2,R3
6962 041422 013703 002102 MOV BANKINDEX,R3
6963 041426 005737 002426 TST NOSUPER
6964 041432 001003 BNE 6$
6965 041434 013700 172246 MOV SIPAR3,R0 ;GENERATE WHAT ERROR ADDR SHOULD BE
6966 041440 000402 BR 7$
6967 041442 013700 177646 6$: MOV UIPAR3,R0
6968 041446 072027 177773 7$: ASH #-5,R0
6969 041452 005737 002130 TST EUFLAG
6970 041456 001002 BNE 1$
6971 041460 042700 177600 BIC #^C177,R0
6972 041464 000301 1$: SWAB R1 ;GET CURRENT ADDRESS BITS 11 AND 12
6973 041466 006201 ASR R1
6974 041470 006201 ASR R1
6975 041472 006201 ASR R1
6976 041474 042701 177775 BIC #^C2,R1
6977 041500 060100 ADD R1,R0 ;ADD THEM TO THE ADJUSTED PAR VALUE
6978 ;GET ERROR ADDRESS FROM CSR UNDER TEST
6979 041502 013701 002144 MOV CSR,R1
6980 041506 072127 177773 ASH #-5,R1
6981 041512 042701 177600 BIC #^C177,R1
6982 041516 005737 002424 TST NO22BIT ;IS THIS AN 11/44 OR 11/24?
6983 041522 001024 BNE 2$ ;BRANCH IF NOT NECESSARY
6984 041524 005737 002130 TST EUFLAG ;IS IT EUB?
6985 041530 001421 BEQ 2$ ;BRANCH IF NOT
6986 041532 PUSH R0 ;SAVE GENERATED ERROR ADDRESS
6987 041534 013702 002146 MOV CSRNO,R2 ;GET CSR NUMBER
6988 041540 052762 040000 172100 BIS #BIT14,CSRADD(R2) ;TURN ON EUB BIT CAREFULLY
6989 041546 016200 172100 MOV CSRADD(R2),R0 ;GET CSR CONTENTS
6990 041552 042762 040000 172100 BIC #BIT14,CSRADD(R2) ;TURN OFF EUB BIT CAREFULLY
6991 041560 042700 177037 BIC #^C740,R0 ;CLEAR EVERYTHING BUT ERROR ADDR
6992 041564 006300 ASL R0
6993 041566 006300 ASL R0 ;SHIFT ADDR BITS 18-21 INTO POSITION
6994 041570 060001 ADD R0,R1 ;ADD TO CURRENT ERROR ADDRESS
6995 041572 POP R0
6996 041574 020001 2$: CMP R0,R1 ;COMPARE REAL AND GENERATED ERR. ADDR.
6997 041576 001420 BEQ 5$ ;BRANCH IF THEY ARE THE SAME
6998 041600 005737 002134 TST INTFLAG ;INTERLEAVED?
6999 041604 001411 BEQ 3$ ;NO - WE HAVE AN ERROR
7000 041606 062700 000100 ADD #100,R0
7001 041612 005737 002136 TST INT64K ;64K INTERLEAVED MEMORY?
7002 041616 001002 BNE 4$
7003 041620 062700 000100 ADD #100,R0
7004 041624 020001 4$: CMP R0,R1
7005 041626 001404 BEQ 5$
7006 041630 005737 002064 3$: TST SKPERR ;ARE WE SUPPOSED TO SKIP ERROR P.O.?
7007 041634 001001 BNE 5$ ;YES - SKIP ERROR PRINTOUT
7008 041636 104462 FERR36 ;ELSE PRINT ERROR ADDRESS ERROR
7009 041640 010137 002430 5$: MOV R1,ERRADD ;SAVE CSR'S ERROR ADDRESS
7010 041644 005037 002064 CLR SKPERR ;ENABLE THE ERROR PRINTOUT AGAIN
7011 041650 POP R3,R2,R1,R0 ;RESTORE REGISTERS
7012 041660 000002 RTI
  
```

7015 041662

```

CHKGEN: SUBTST<<SUBR GENERATE CHECK BITS>>
:*****
:*SUBTEST SUBR GENERATE CHECK BITS
:*****
:CHECK BIT GENERATOR ROUTINE
:CALLING SEQUENCE IS:
: MOV #WORD1,SOURCE ;SOURCE = ADDRESS OF DATA
: CALL CHKGEN
:CHECK BITS RETURNED IN BITS 11-5 OF LOCATION CHECK
:
PUSH R0,R1,R2,R3,R4,R5
MOV #77,R2 ;DEFAULT CHECKBITS FOR DOUBLE WORD OF ZEROS
MOV #CHKTAB,R3 ;ADDRESS OF CHECKBIT TABLE
MOV SOURCE,R5 ;GET SOURCE ADDRESS
MOV (R5)+,R1 ;GET LSB'S
MOV (R5),R0 ;GET MSB'S
:
1$: SXT R4 ;EXTEND SIGN OF DOUBLE WORD TO R4
BICB (R3)+,R4 ;ELIMINATE BITS THAT DON'T COUNT
XOR R4,R2 ;COMPLEMENT MASKED BITS IN CHECKBITS
ASHC #1,R0 ;DOUBLE PRECISION LEFT SHIFT R0,,R1
BNE 1$ ;LOOP TILL ALL BITS ARE CHECKED
:
BIC #^C177,R2 ;KILL ALL JUNK BITS
SWAB R2 ;POSITION CHECKBITS IN BITS 11-5
ASR R2
ASR R2
ASR R2
MOV R2,CHECK
POP R5,R4,R3,R2,R1,R0
RETURN
    
```

7016  
 7017  
 7018  
 7019  
 7020  
 7021  
 7022  
 7023 041662  
 7024 041676 012702 000077  
 7025 041702 012703 041770  
 7026 041706 013705 002272  
 7027 041712 012501  
 7028 041714 011500  
 7029  
 7030 041716 006704  
 7031 041720 142304  
 7032 041722 074402  
 7033 041724 073027 000001  
 7034 041730 001372  
 7035  
 7036 041732 042702 177600  
 7037 041736 000302  
 7038 041740 006202  
 7039 041742 006202  
 7040 041744 006202  
 7041 041746 010237 002274  
 7042 041752  
 7043 041766 000207

Address	Offset	Value	Label	Bit
7046	041770		CHKTAB: .BYTE #3	
7047	041770	200	.BYTE ^C177	:BIT 31
7048	041771	301	.BYTE ^C076	:BIT 30
7049	041772	302	.BYTE ^C075	:BIT 29
7050	041773	203	.BYTE ^C174	:BIT 28
7051	041774	304	.BYTE ^C073	:BIT 27
7052	041775	205	.BYTE ^C172	:BIT 26
7053	041776	206	.BYTE ^C171	:BIT 25
7054	041777	307	.BYTE ^C070	:BIT 24
7055			.BYTE #2	
7056	042000	310	.BYTE ^C067	:BIT 23
7057	042001	211	.BYTE ^C166	:BIT 22
7058	042002	212	.BYTE ^C165	:BIT 21
7059	042003	313	.BYTE ^C064	:BIT 20
7060	042004	214	.BYTE ^C163	:BIT 19
7061	042005	315	.BYTE ^C062	:BIT 18
7062	042006	316	.BYTE ^C061	:BIT 17
7063	042007	217	.BYTE ^C160	:BIT 16
7064			.BYTE #1	
7065	042010	320	.BYTE ^C057	:BIT 15
7066	042011	221	.BYTE ^C156	:BIT 14
7067	042012	222	.BYTE ^C155	:BIT 13
7068	042013	323	.BYTE ^C054	:BIT 12
7069	042014	224	.BYTE ^C153	:BIT 11
7070	042015	325	.BYTE ^C052	:BIT 10
7071	042016	326	.BYTE ^C051	:BIT 9
7072	042017	227	.BYTE ^C150	:BIT 8
7073			.BYTE #0	
7074	042020	340	.BYTE ^C037	:BIT 7
7075	042021	241	.BYTE ^C136	:BIT 6
7076	042022	242	.BYTE ^C135	:BIT 5
7077	042023	343	.BYTE ^C034	:BIT 4
7078	042024	244	.BYTE ^C133	:BIT 3
7079	042025	345	.BYTE ^C032	:BIT 2
7080	042026	346	.BYTE ^C031	:BIT 1
7081	042027	247	.BYTE ^C130	:BIT 0

7084 042030

```

SUBTST<<SUBR  MAPPER>>
:*****
:*SUBTEST      SUBR  MAPPER
:*****

```

7085  
7086  
7087  
7088  
7089  
7090  
7091  
7092  
7093  
7094

```

: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 ARGUMENT
:      CALL   MAPPER        ;ACTUAL CALL
:      RETURN              ;ONLY RETURN

```

7095 042030  
7096 042042 012700 172340  
7097 042046 012701 172240  
7098 042052 012704 172200  
7099 042056 005737 002426  
7100 042062 001404  
7101 042064 012701 177640  
7102 042070 012704 177600  
7103 042074 012702 077406  
7104 042100 012705 000010  
7105 042104 012021  
7106 042106 010224  
7107 042110 077503  
7108 042112 012741 177600  
7109

```

:SET SUPERVISOR/USER UP FOR 1 TO 1 MAP
MAPPER: PUSH  R0,R1,R2,R4,R5
:      MOV   #KIPAR0,R0      ;FIRST AREA TO MAP TO
:      MOV   #SIPAR0,R1      ;FIRST ADDRESS REGISTER
:      MOV   #SIPDR0,R4      ;FIRST DESCRIPTOR REGISTER
:      TST   NOSUPER        ;CAN WE USE SUPERVISOR MODE?
:      BEQ   4$             ;YES, BRANCH
:      MOV   #UIPAR0,R1      ;FIRST ADDRESS REGISTER
:      MOV   #UIPDR0,R4      ;FIRST DESCRIPTOR REGISTER
4$:   MOV   #77406,R2        ;CONSTANT FOR 4K PAGE, UP, R/W
:      MOV   #8,R5           ;COUNTER
1$:   MOV   (R0)+,(R1)+      ;PUT IN SUPERVISOR ADDRESS
:      MOV   R2,(R4)+        ;PUT IN SUPERVISOR DESCRIPTOR
:      SOB   R5,1$         ;LOOP TILL DONE
:      MOV   #177600,-(R1)   ;CORRECT LAST FIELD FOR PERIPHERALS PAGE

```

7110  
7111 042116 022703 000170  
7112 042122 001516  
7113 042124 072327 000011  
7114  
7115 042130 012701 172246  
7116 042134 005737 002426  
7117 042140 001402  
7118 042142 012701 177646  
7119 042146 012702 000004  
7120 042152 010321  
7121 042154 062703 000200  
7122 042160 077204  
7123 042162 005737 002232  
7124 042166 001442  
7125 042170 162701 000010  
7126 042174 010102  
7127 042176 062702 000004  
7128 042202 022737 000001 002232  
7129 042210 001403  
7130 042212 010200  
7131 042214 010102  
7132 042216 010001  
7133 042220 012122  
7134 042222 011112  
7135 042224 013700 002102  
7136 042230 005737 002136  
7137 042234 001403

```

:SET UP SUPERVISOR/USER FOR TEST AREA
:      CMP   #120.,R3      ;MAP NOTHING (1 TO 1)?
:      BEQ   3$             ;YES - SKIP
:      ASH   #9.,R3        ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
:                               ;FOR MEMORY MANAGEMENT = 1000
:      MOV   #SIPAR3,R1     ;SETUP FOR AUTO INCREMENTING
:      TST   NOSUPER        ;DO WE HAVE SUPERVISOR MODE?
:      BEQ   5$             ;YES - BRANCH
:      MOV   #UIPAR3,R1     ;SETUP FOR AUTO INCREMENTING
5$:   MOV   #4,R2           ;COUNTER
2$:   MOV   R3,(R1)+        ;PLUG IN PAR INFO
:      ADD   #200,R3        ;BUMP ADDRESS 4K
:      SOB   R2,2$         ;LOOP TILL DONE
:      TST   SPLTCSR
:      BEQ   9$
:      SUB   #10,R1
:      MOV   R1,R2
:      ADD   #4,R2
:      CMP   #1,SPLTCSR
:      BEQ   10$
:      MOV   R2,R0
:      MOV   R1,R2
:      MOV   R0,R1
10$:  MOV   (R1)+,(R2)+
:      MOV   (R1),(R2)
:      MOV   BANKINDEX,R0
:      TST   INT64K
:      BEQ   11$

```

```

7138 042236 012700 004000      MOV    #4000,R0
7139 042242 000402              BR     12$
7140 042244 012700 010000      11$:  MOV    #10000,R0
7141 042250 005737 002426      12$:  TST    NOSUPER
7142 042254 001403              BEQ    13$
7143 042256 012701 177652      MOV    #UIPAR5,R1
7144 042262 000402              BR     14$
7145 042264 012701 172252      13$:  MOV    #SIPAR5,R1
7146 042270 060021      14$:  ADD    RO,(R1)+
7147 042272 060011      ADD    RO,(R1)
7148                          ;IF WE ONLY HAVE AN 124K SYSTEM, WE DON'T WANT TO TEST THE
7149                          ;LAST 4K, WHERE THE UNIBUS DEVICE PAGE IS. INSTEAD, THE
7150                          ;PROGRAM WILL REMAP THE LAST 4K TO 8-12K. ALSO, IF THERE
7151                          ;IS A BANK 177 ON AN 11/44, THE PROGRAM WILL REMAP THE LAST
7152                          ;4K TO 8-12K FOR THE SAME REASON.
7153 042274 022737 000007 002526 9$:  CMP    #7, LASTBANK
7154 042302 001010              BNE    7$
7155 042304 005737 002424              TST    NO22BIT                ;11/44 OR 24?
7156 042310 001423              BEQ    3$                      ;BRANCH IF SO
7157 042312 022737 000007 002100      CMP    #7, BANK                ;BANK 7?
7158 042320 001017              BNE    3$                      ;NO - BRANCH
7159 042322 000404              BR     8$
7160 042324 022737 000177 002526 7$:  CMP    #177, LASTBANK
7161 042332 001012              BNE    3$
7162 042334 005737 002426      8$:  TST    NOSUPER
7163 042340 001404              BEQ    6$
7164 042342 013737 177652 177654      MOV    UIPAR5,UIPAR6
7165 042350 000403              BR     3$
7166 042352 013737 172252 172254 6$:  MOV    SIPAR5,SIPAR6
7167 042360              3$:  POP    R5,R4,R2,R1,R0
7168 042372 000207      RETURN
7169                          .SBTTL  TRAP    MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
7170 042374              $KMAP: PUSH   RO,R1,R2,R3,R4
7171 042406 005000              CLR    RO                      ;1ST AREA TO MAP TO
7172 042410 012701 172340      MOV    #KIPAR0,R1              ;FIRST ADDRESS
7173 042414 012702 077406      MOV    #77406,R2              ;CONSTANT FOR 4k PAGE,UP,R/W
7174 042420 012703 172300      MOV    #KIPDR0,R3            ;1ST PAGE DESCRIPTOR REGISTER
7175 042424 012704 000010      MOV    #8,R4                  ;COUNTER
7176 042430 010021      1$:  MOV    RO,(R1)+                ;PUT IN KERNEL ADDRESS
7177 042432 010223      MOV    R2,(R3)+                ;PUT IN KERNEL DISCRIPTOR
7178 042434 062700 000200      ADD    #200,R0                ;ADD ADDRESS CONSTANT FOR 4K CHANGE
7179 042440 077405              SOB    R4,1$                  ;LOOP TILL DONE
7180 042442 012741 177600      MOV    #177600,-(R1)          ;THE PERIPHERALS PAGE TO KIPAR7
7181 042446 012741 177400      MOV    #177400,-(R1)          ;AND NEXT LOWER PAGE TO KIAR6
7188 042452              POP    R4,R3,R2,R1,R0
7189 042464 000002      RTI
    
```

7192 042466  
 7193 042466  
 7194 042502  
 7195 042516  
 7196 042530  
 7197 042530  
 7198 042530  
 7199 042536 004737 044240  
 7200 042542  
 7201 042564 013700 002100  
 7202 042570 010037 002402  
 7203 042574 013701 002536  
 7204 042600 004737 043710  
 7205 042604 004737 044206  
 7206 042610 013701 002102  
 7207 042614 052761 100000 002626  
 7208 042622 042761 020000 002626  
 7209 042630  
 7210 042632  
 7211 042632  
 7212 042646  
 7213 042656  
 7214 042662  
 7215 042662  
 7216 042666  
 7217 042666  
 7218 042666 013702 002526  
 7219 042672 006302  
 7220 042674 006302  
 7221 042676  
 7222 042702  
 7223 042712  
 7224 042722  
 7225 042732  
 7226 042742  
 7227  
 7228 042762  
 7229 042764  
 7230 042764  
 7231 042772  
 7232 043000 010137 002510  
 7233 043004  
 7234 043004  
 7235 043004  
 7236 043004  
 7237 043004  
 7238 043014  
 7239 043022  
 7240 043030 013701 002510  
 7241 043034 023727 002260 000030  
 7242 043042 001423  
 7243 043044  
 7244 043050 000420  
 7245 043052

```

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
MOV      LASTBANK,R2
ASL      R2
ASL      R2      ;R2 <- R2 * 4
FOR R1 := #2*2 TO R2 BY #4
IF #BIT7!BIT0 OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT PROGRAM SPACE
IF #BIT15 OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK
IF CPUBIT SET.IN CONFIG(R1) ;IF ACCESSABLE
IF #BIT9 SET.IN CONFIG+2(R1) THEN LEAVE FINDBANK ;IF PARITY
IF #BIT6 SET.IN CONFIG(R1) AND #BIT7 OFF.IN CONFIG(R1)
;IF 1ST PROTECTABLE ECC BANK
LEAVE FINDBANK
END ;OF IF #BIT6
IF INHECC IS FALSE
SET      INHECC
MOV      R1,INHBANK
END; OF IF INHECC
END ;OF IF CPUBIT
END ;OF IF #BIT15
END ;OF IF #BIT7
END ;OF FOR
IF FULLREL IS FALSE
IF INHECC IS TRUE
MOV      INHBANK,R1
CMP      REALPAT,#30      ;IS THIS PATTERN 30?
BEQ      RELENT1         ;YES - SKIP MESSAGE
TYPE     MSG123
BR       RELENT1
END; OF IF INHECC
  
```



```

7246 043052          END; OF IF FULLREL
7247 043052 0C5037 002506 CLR INHECC          ;MAKE SURE FLAG IS TURNED OFF!
7248 043056          IF #SW13 OFF.IN @SWR
7249 043066 023727 002260 000030 CMP REALPAT,#30    ;IS THIS PATTERN 30?
7250 043074 001402          BEQ SKUB                ;YES - SKIP MESSAGE
7251 043076          TYPE MSG075                ;RELOCATION NOT POSSIBLE
7252 043102          END ;OF IF #SW13
7253 043102          SKUB: $RETURN ERROR
7254 043106          END FINDBANK
7255 043106          CLEAR INHECC          ;IF WE RELOCATED PROPERLY, THIS SHOULD BE OFF!
7256 043112 042761 020000 002626 RELENT1: BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
7257 043120 005000          CLR RO
7258 043122 071027 000004          DIV #4,RO
7259 043126          RELOC1: LET NEWBANK := RO
7260 043132 013737 002502 002504 MOV PGMCSR,PGMCSR+2 ;SAVE CURRENT PGM. CSR
7261 043140 004737 044056          CALL USERMAP      ;MAP NEWBANK TO USER PAR
7262 043144          USER                    ;ENTER USER MODE
7263 043152          BMOV 0,100000,SIZE      ;MOVE PROGRAM
7264 043164 104417          KERNEL          ;ENTER KERNEL MODE
7265 043166 022737 000001 003710 CMP #1,PROTYP      ;IS THIS AN 11/44 ?
7266 043174 001021          BNE JMPRL1       ;JUMP IF NOT
7267 043176 042737 000040 172516 BIC #BIT5,MMR3    ;TURN OFF UNIBUS MAP
7268 043204 013700 002270          MOV NEWBANK,RO
7269 043210 006200          ASR RO
7270 043212          ON.ERROR
7271 043214 012737 100000 170200 MOV #BIT15,MAPLO
7272 043222          END ;OF ON.ERROR
7273 043222 010037 170202          MOV RO,MAPHO
7274 043226 004737 043644          CALL LOWMAP      ;SETUP LOWER 16K IN UNIBUS MAP
7275 043232 052737 000040 172516 BIS #BIT5,MMR3    ;ENERGIZE UNIBUS MAP
7276 043240 042737 000001 177572 JMPRL1: BIC #BIT0,MMR0 ;DEENERGIZE MEMORY MANAGEMENT
7277 043246 004737 044140          CALL NEWKERNEL
7278 043252 013700 002270          MOV NEWBANK,RO
7279 043256 006300          ASL RO
7280 043260 006300          ASL RO          ;RO <- RO * 4
7281 043262 016002 002624          MOV CONFIG(RO),R2
7282 043266 000302          SWAB R2
7283 043270 042702 177760          BIC #^C17,R2
7284 043274 006302          ASL R2
7285 043276 052737 000001 177572 BIS #BIT0,MMR0    ;ENERGIZE MEMORY MANAGEMENT
7286 043304 010237 002502          MOV R2,PGMCSR      ;PUT NEW PGM. CSR INTO PGMCSR
7287 043310 032760 010000 002626 BIT #BIT12,CONFIG+2(RO) ;IS THE NEW BANK INTERLEAVED?
7288 043316 001412          BEQ 1$          ;BRANCH IF NOT INTERLEAVED
7289 043320 016002 002624          MOV CONFIG(RO),R2
7290 043324 042702 007777          BIC #^C170000,R2
7291 043330 072227 177775          ASH #-3,R2
7292 043334 052702 100000          BIS #BIT15,R2
7293 043340 050237 002502          BIS R2,PGMCSR
7294 043344          1$: SET KLFLAG
7295 043352          $RETURN NOERROR

```

7298 043356

```
UNRELOCATE:SUBTST <<UNRELOCATE PROGRAM>>
:*****
:*SUBTEST UNRELOCATE PROGRAM
:*****
```

7299

```
7300 043356
7301 043360 013701 002402
7302 043364 013700 002536
7303 043370 004737 043710
7304 043374 004737 044206
7305 043400
7306 043404 013737 002402 002100
7307 043412 004737 044240
7308 043416 013701 002102
7309 043422 042761 100000 002626
7310 043430 013737 002536 002100
7311 043436 004737 044240
7312 043442 013701 002102
7313 043446 042761 020000 002626
7314 043454
7315 043460
7316
7317
7318 043464 042737 020000 002626
7319 043472
7320 043476 004737 044056
7321 043502
7322 043510
7323 043522 104417
7324 043524 042737 000001 177572
7325 043532 004737 044140
7326 043536 013737 002504 002502
7327 043544 052737 000001 177572
7328 043552 005037 002124
7329 043556 022737 000001 003710
7330 043564 001014
7331 043566 042737 000040 172516
7332 043574
7333 043604 004737 043644
7334 043610 052737 000040 172516
7335 043616 012700 002626
7336 043622 042710 020000
7337 043626 062700 000004
7338 043632 020027 003620
7339 043636 003771
7340 043640
7341 043642 000207
7342
7343 043644
```

```

;RESTORE LOADERS
PUSH RO
MOV LOADBANK,R1
MOV LOADHOME,RO
CALL BANKMOV
CALL NEWLOAD ;MAP NEW LOADER BANK IN KERNEL SPACE
PUSH BANK
MOV LOADBANK,BANK
CALL EXBANK
MOV BANKINDEX,R1
BIC #BIT15,CONFIG+2(R1) ;CLEAR LOADER FLAG
MOV LOADHOME,BANK
CALL EXBANK
MOV BANKINDEX,R1
BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
POP BANK
CLEAR INHECC ;MAKE SURE ECC TESTS ARE NOT INHIBITED!

;RESTORE BANK 0
BIC #BIT13,CONFIG+2 ;INVALIDATE BACKGROUND PATTERN
LET NEWBANK := #0
CALL USERMAP ;MAP NEWBANK TO USER PAR
USER ;ENTER USER MODE
BMOV 0,100000,SIZE ;MOVE PROGRAM
KERNEL ;ENTER KERNEL MODE
BIC #BIT0,MMRO ;DEENERGIZE MEMORY MANAGEMENT
CALL NEWKERNEL
MOV PGMCSR+2,PGMCSR ;RESTORE PREVIOUS PGM. CSR
BIS #BIT0,MMRO ;ENERGIZE MEMORY MANAGEMENT
CLR RLFLAG
CMP #1,PROTYP ;IS THIS AN 11/44 ?
BNE 1$
BIC #BITS5,MMR3 ;TURN OFF UNIBUS MAP
CLEAR MAPLO,MAPHO
CALL LOWMAP ;SETUP LOWER 16K OF UNIBUS MAP
BIS #BITS5,MMR3 ;ENERGIZE UNIBUS MAP
1$: MOV #CONFIG+2,RO ;MOVE 2ND WORD OF CONFIG TO RO
2$: BIC #BIT13,(RO) ;CLEAR BACKGROUND VALID BIT
ADD #4,RO ;INCREMENT TO NEXT BANK
CMP RO,#3620 ;DONE?
BLE 2$ ;NO - BRANCH
POP RO
RETURN
```

```
LOWMAP: SUBTST <<SETUP LOWER 16K OF UNIBUS MAP>>
:*****
:*SUBTEST SETUP LOWER 16K OF UNIBUS MAP
:*****
```

```
7344 043644
7345 043652 012700 170200
7346 043656 012701 170204
7347 043662 012702 000003
7348 043666 012011
```

```

PUSH RO,R1,R2
MOV #MAPLO,RO
MOV #MAPL1,R1
MOV #3,R2
1$ MOV (RO)+,(R1)
```

7349 043670 062721 020000  
7350 043674 012021  
7351 043676 077205  
7352 043700  
7353 043706 000207

ADD #BIT13,(R1)+  
MOV (R0)+,(R1)+  
SOB R2,1\$  
POP R2,R1,R0  
RETURN

7356 043710

```
BANKMOV:SUBTST <<MOVE BANKS>>
:*****
:*SUBTEST      MOVE BANKS
:*****
```

```
7357
7358
7359
7360
7361 043710 104415
7362 043712 004737 044056
7363 043716 104416
7364 043720 104415
7365 043722 072027 000011
7366 043726 072127 000011
7367 043732 012702 177650
7368 043736 012703 000200
7369
7370 043742 010122
7371 043744 060301
7372 043746 010122
7373 043750 060301
7374
7375 043752 010022
7376 043754 060300
7377 043756 010022
7378 043760 060300
7379
7380 043762
7381 043770
7382 044002 104417
7383
7384 044004 012702 177650
7385
7386 044010 010122
7387 044012 060301
7388 044014 010122
7389 044016 060301
7390
7391 044020 010022
7392 044022 060300
7393 044024 010022
7394 044026 060300
7395
7396 044030
7397 044036
7398 044050 104417
7399
7400 044052 104416
7401 044054 000207
```

```
;MOVE 3/4 OF A BANK
;CALLING SEQUENCE
;R0 = DESTINATION BANK
;R1 = SOURCE BANK
SAVREG
CALL  USERMAP
RESREG
SAVREG
ASH  #9.,R0
ASH  #9.,R1
MOV  #UIPAR4,R2
MOV  #200,R3

MOV  R1,(R2)+      ;MAP 1ST HALF BANK
ADD  R3,R1         ;BUMP BY 4K
MOV  R1,(R2)+
ADD  R3,R1

MOV  R0,(R2)+
ADD  R3,R0
MOV  R0,(R2)+
ADD  R3,R0

USER
BMOV 100000,140000,SIZE/2 ;MOV 1ST HALF BANK
KERNEL ;ENTER KERNEL MODE

MOV  #UIPAR4,R2

MOV  R1,(R2)+      ;MAP 2ND HALF BANK
ADD  R3,R1         ;BUMP BY 4K
MOV  R1,(R2)+
ADD  R3,R1

MOV  R0,(R2)+
ADD  R3,R0
MOV  R0,(R2)+
ADD  R3,R0

USER
BMOV 100000,140000,SIZE/4 ;MOV 3RD FOURTH OF BANK
KERNEL ;ENTER KERNEL MODE

RESREG
RETURN
```

7404 044056

```
USERMAP:SUBTST <<SUBR MAP USER TO NEW BANK>>
:*****
:*SUBTEST      SUBR      MAP USER TO NEW BANK
:*****
```

7405 044056 012701 177640  
7406 044062 012702 172340  
7407 044066 012703 177600  
7408 044072 012704 172300  
7409 044076 012705 000004  
7410 044102 012221  
7411 044104 011423  
7412 044106 077503  
7413  
7414 044110 013700 002270  
7415 044114 072027 000011  
7416  
7417 044120 012705 000004  
7418 044124 010021  
7419 044126 062700 000200  
7420 044132 011423  
7421 044134 077505  
7422 044136 000207  
7423  
7424 044140

```
      MOV      #UIPAR0,R1      ;COPY KERNEL PAR'S & PDR'S (0-3)
      MOV      #KIPAR0,R2
      MOV      #UIPDR0,R3
      MOV      #KIPDR0,R4
      MOV      #4,R5
1$:    MOV      (R2)+,(R1)+
      MOV      (R4),(R3)+
      SOB      R5,1$
      MOV      NEWBANK,R0
      ASH      #9.,R0          ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
                                   ;FOR MEMORY MANAGEMENT = 1000
      MOV      #4,R5
2$:    MOV      R0,(R1)+      ;SETUP UIPAR(4-7)
      ADD      #200,R0        ;BUMP ADDRESS 4K
      MOV      (R4),(R3)+    ;SETUP UIPDR(4-7)
      SOB      R5,2$
      RETURN
```

```
NEWKERNEL:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW BANK>>
:*****
:*SUBTEST      SUBR      SETUP KERNEL PAR'S FOR NEW BANK
:*****
```

7425 044140  
7426 044146 012700 172340  
7427 044152 013701 002270  
7428 044156 072127 000011  
7429  
7430 044162 012705 000004  
7431 044166 010120  
7432 044170 062701 000200  
7433 044174 077504  
7434 044176  
7435 044204 000207  
7436  
7437 044206

```
      PUSH     R0,R1,R5
      MOV      #KIPAR0,R0
      MOV      NEWBANK,R1
      ASH      #9.,R1          ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
                                   ;FOR MEMORY MANAGEMENT = 1000
      MOV      #4,R5
1$:    MOV      R1,(R0)+      ;SETUP KIPAR(0-3)
      ADD      #200,R1
      SOB      R5,1$
      POP      R5,R1,R0
      RETURN
```

```
NEWLOAD:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
:*****
:*SUBTEST      SUBR      SETUP KERNEL PAR'S FOR NEW LOADER BANK
:*****
```

7438  
7439 044206  
7440 044212 012701 172350  
7441 044216 072027 000011  
7442 044222 010021  
7443 044224 062700 000200  
7444 044230 010021  
7445 044232  
7446 044236 000207

```
      ;R0 CONTAINS THE DESTINATION BANK
      PUSH     R0,R1
      MOV      #KIPAR4,R1
      ASH      #9.,R0          ;BANK 1 STARTS AT 100000 LESS 6 LSB'S (1000)
      MOV      R0,(R1)+      ;SETUP KIPAR4
      ADD      #200,R0
      MOV      R0,(R1)+      ;SETUP KIPAR5
      POP      R1,R0
      RETURN
```

7449 044240

```

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.
    
```

7450  
7451  
7452  
7453  
7454  
7455  
7456  
7457  
7458  
7459  
7460  
7461  
7462  
7463  
7464  
7465  
7466  
7467  
7468  
7469  
7470

7471 044240  
7472 044246  
7473 044266  
7474 044274  
7475 044310  
7476 044324 013701 002100  
7477 044330 006301  
7478 044332 006301  
7479 044334 010137 002102  
7480 044340 032761 000100 002624  
7481 044346 001403  
7482 044350  
7483 044356 012700 000002 1\$:  
7487 044362  
7488 044376 005037 002114  
7489 044402  
7494 044402 005737 002114  
7495 044406 001415  
7496 044410 016102 002626  
7497 044414 000302  
7498 044416 042702 177770  
7499 044422 020227 000002  
7500 044426 003005  
7501 044430  
7502 044436 000137 044700  
7503 044442 032761 000400 002626 12\$:  
7504 044450 001003  
7505 044452  
7506 044460 032761 001000 002626 2\$:  
7507 044466 001012  
7508 044470  
7509 044476 032761 000400 002626

```

PUSH R0,R1,R2
CLEAR MKFLAG,KPFLAG,KFLAG,EUFLAG
SET ACFLAG
CLEAR PFLAG,RRFLAG,BMFLAG
CLEAR INTFLAG,INT64K,SKIPMK
MOV BANK,R1
ASL R1
ASL R1 ;R1 <- R1 * 4
MOV R1,BANKINDEX
BIT #16,CONFIG(R1) ;PROTECTED REGION OF ECC MEMORY?
BEQ $ ;NO - SKIP
SET KPFLAG
MOV #BIT1,R0
IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)
CLR ACFLAG
END :OF IF R0
TST ACFLAG ;ACTIVE MEMORY?
BEQ 12$ ;BRANCH IF NOT
MOV CONFIG+2(R1),R2
SWAB R2
BIC #*C7,R2 ;ISOLATE MEM TYPE BITS
CMP R2,#2 ;IS THIS AN ILLEGAL MEM TYPE?
BGT 12$ ;BRANCH IF NOT
SET BMFLAG ;SET BAD BANK FLAG
JMP ENEXBK ;JUMP OVER REST OF FLAG TESTS
BIT #BIT8,CONFIG+2(R1) ;IS THIS EUB?
BNE 2$ ;BRANCH IF NOT
SET EUFLAG ;YES - SET EUB FLAG
BIT #BIT9,CONFIG+2(R1) ;IS THERE ECC THERE?
BNE 3$ ;NO - SKIP
SET MKFLAG ;YES - SET MKFLAG
BIT #BIT8,CONFIG+2(R1) ;IS THIS MF11S-K MEMORY
    
```

```

7510 044504 001403      BEQ      3$      ;NO - IT'S MS11-M
7511 044506           SET      KFLAG   ;YES - SET KFLAG
7512 044514 032761 000200 002624 3$: BIT      #BIT7,CONFIG(R1) ;BANK = PROGRAM SPACE?
7513 044522 001406      BEQ      5$      ;NO - SKIP
7514 044524           SET      PFLAG,RRFLAG
7515 044540 005737 002124      5$: TST      RLFLAG   ;IS PROGRAM RELOCATED?
7516 044544 001402      BEQ      6$      ;NO - SKIP
7517 044546 005137 002122      COM      RRFLAG   ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
7518 044552 032761 000001 002624 6$: BIT      #BIT0,CONFIG(R1) ;ERRORS PRESENT IN THIS BANK?
7519 044560 001403      BEQ      8$      ;NO - SKIP
7520 044562           SET      BMFLAG
7521 044570 005737 002540      8$: TST      WORST   ;IS THIS A WORST FIRST PASS?
7522 044574 001002      BNE      9$      ;YES - SKIP
7523 044576 005137 002126      COM      BMFLAG   ;NO - COMPLEMENT BAD MEMORY FLAG
7524 044602           9$: IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
7525 044620           SET      RRFLAG
7526 044626           END ;OF IF SELONLY
7527 044626 032761 010000 002626      BIT      #BIT12,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED?
7528 044634 001421      BEQ      ENEXBK  ;BRANCH IF IT IS NOT
7529 044636           SET      INTFLAG
7530 044644 032761 004000 002626      BIT      #BIT11,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED WITH 64K BOARDS?
7531 044652 001403      BEQ      10$     ;BRANCH IF IT IS NOT
7532 044654           SET      INT64K
7533 044662 032761 000040 002624 10$: BIT      #BIT5,CONFIG(R1) ;SHOULD THIS BANK BE TESTFD?
7534 044670 001403      BEQ      ENEXBK  ;BRANCH IF IT SHOULD
7535 044672           SET      SKIPMK
7536 044700           ENEXBK: POP      R2,R1,RO ;RESTORE REGISTERS
7537 044706 000207           RETURN
    
```

7540 044710

BANKOK: SUBTST <<SUBR BANK OK?>>  
:\*\*\*\*\*  
:\*SUBTEST SUBR BANK OK?  
:\*\*\*\*\*

7541

:TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK

7542

:IS OF THE TYPE WE ARE TESTING 'TMFLAG'.

7543

:RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).

7544 044710 013700 002132

MOV TMFLAG,RO

7545 044714 005100

COM RO

7546 044716 013701 002116

MOV MKFLAG,R1

7547 044722 074001

XOR RO,R1

7548 044724 000207

RETURN

;OK = (=OK)

7549

7550 044726

INCRPT:

7551 044726

INCPAT: SUBTST <<SUBR INCREMENT PATTERN TESTING >>  
:\*\*\*\*\*  
:\*SUBTEST SUBR INCREMENT PATTERN TESTING  
:\*\*\*\*\*

7552

:INCREMENT THE PATTERN & SET UP THE CONDITION CODES

7553

:RESULT - Z BIT SET INDICATES OVERFLOW

7554 044726 005237 002110

INC PATTERN

7555 044732 022737 000030 002110

CMP #30,PATTERN

;SET UP CONDITION CODES

7556 044740 000207

RETURN

;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)

7557

7558 044742

SETPAT:

7559 044742

HIPAT: SUBTST <<SUBR SET HIGHEST PATTERN TESTING TYPE>>  
:\*\*\*\*\*  
:\*SUBTEST SUBR SET HIGHEST PATTERN TESTING TYPE  
:\*\*\*\*\*

7560 044742 012737 000027 002110

MOV #27,PATTERN

;SET HIGHEST PATTERN

7561 044750 000207

RETURN

7562

7563 044752

INCBNK: SUBTST <<SUBR INCREMENT BANK & TEST>>  
:\*\*\*\*\*  
:\*SUBTEST SUBR INCREMENT BANK & TEST  
:\*\*\*\*\*

7564

:RESULTS RETURNED IN CONDITION CODES

7565 044752 005237 002100

INC BANK

7566 044756 023737 002526 002100

CMP LASTBANK,BANK

;TOO FAR?

7567 044764 000207

RETURN



7570 044766

```
BOOT: SUBTST <<BOOTSTRAP ROUTINE>>
:*****
:*SUBTEST BOOTSTRAP ROUTINE
:*****
7571 ;INITIALIZE ALL CSR'S
7572 ;UNRELOCATE IF NECESSARY
7573 ;FLUSH OUT ANY DBE'S
7574 ;TURN OFF MEMORY MANAGEMENT
7575 ;TURN OFF THE UNIBUS MAP
7576 ;BOOT RKO OR RK1
7577 044766 104472 ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7578 044770 SET4 #BOOT1 ;TRAPS TO 4 GOTO BOOT1
7579 044776 IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
7580 045010 004737 024656 CALL MT0030 ;FLUSH OUT DBE'S
7581 045014 104421 DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
7582 045016 005737 002424 TST NO22BIT ;IS THIS AN 11/44 OR 11/24?
7583 045022 001003 BNE BOOT1
7584 045024 042737 000040 172516 BIC #BIT5,MMR3 ;TURN OFF THE UNIBUS MAP
7585 045032 005001 BOOT1: CLR R1
7586 045034 000005 1$: RESET
7587 045036 012700 177406 MOV #177406,R0
7588 045042 010160 000004 MOV R1,4(R0)
7589 045046 012710 177400 MOV #177400,(R0)
7590 045052 012740 000005 MOV #5,-(R0)
7591 045056 105710 2$: TSTB (R0)
7592 045060 100376 BPL 2$
7593 045062 062701 020000 ADD #BIT13,R1
7594 045066 005710 TST (R0)
7595 045070 100761 BMI 1$
7596 045072 005007 CLR PC
```

7599 045074

```
EXIT:  SUBTST <<HALT PROGRAM>>
:*****
:*SUBTEST  HALT PROGRAM
:*****
```

7600 045074 004737 045126  
7601 045100  
7602 045114 000777  
7603 045116  
7604 045120 000000  
7605 045122 000137 003630  
7606 045126  
7607  
7608 045126

```
CALL SHUTUP
EXIT2: IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
      BR .
      ELSE
$EXHALT: HALT
        JMP START
        END ;OF IF APTFLAG
```

```
SHUTUP: SUBTST <<SHUTDOWN DIAGNOSTIC>>
:*****
:*SUBTEST  SHUTDOWN DIAGNOSTIC
:*****
```

7609  
7610  
7611  
7612  
7613  
7614  
7618 045126 104472  
7619 045130  
7620 045142  
7621 045150 004737 024656  
7622 045154  
7623 045154 012700 000001  
7624 045160 013701 002536  
7625 045164 004737 043710  
7626 045170 104421  
7627 045172 005737 002424  
7628 045176 001003  
7629 045200 042737 000040 172516  
7633 045206 000207  
7634  
7635 045210

```
;INITIALIZE ALL CSR'S
;UNRELOCATE
;FLUSH OUT DBE'S
;RESTORE LOADERS
;TURN OFF MEMORY MANAGEMENT
;UNMAP THE UNIBUS MAP
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
IF QUICK IS FALSE
  CALL MTO030 ;FLUSH OUT DBE'S
END ;OF IF QUICK
MOV #1,R0 ;DESTINATION BANK
MOV LOADHOME,R1 ;SOURCE BANK
CALL BANKMOV
DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
TST NO22BIT ;DOES THIS PDP-11 HAVE 22-BIT ADDR?
BNE 1$ ;BRANCH IF NOT
BIC #BITS,MMR3 ;TURN OFF UNIBUS MAP
1$: RETURN
```

```
APTDOWN: SUBTST <<APT SHUTDOWN SEQUENCE>>
:*****
:*SUBTEST  APT SHUTDOWN SEQUENCE
:*****
```

7636 045210  
7637 045224  
7638 045232 012737 045210 060024  
7639 045240 012737 000340 060026  
7640 045246 012737 000000 125210  
7641 045254 104417  
7642 045256 000000

```
MAP #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
TESTAREA ;ENTER TEST MODE
MOV #APTDOWN,FIRST+24
MOV #340,FIRST+26
MOV #0,FIRST+APTDOWN
KERNEL ;ENTER KERNEL MODE
APTHLT: HALT
```

7645 045260

```
      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
BLOCK1: PUSH   R0,R1,R2
      MOV     #FASTCITY,R2
      MOV     #16.,R1
      BR      3$
BLOCK2: PUSH   R0,R1,R2
      MOV     #16.,R1
      BR      2$
BLOCK3: PUSH   R0,R1,R2
      MOV     (R5)+,R1
2$:    MOV     (R5)+,R2
3$:    MOV     (R5)+,R0
1$:    MOV     (R0)+,(R2)+
      SOB     R1,1$
      POP     R2,R1,R0
      RTS     R5
      .DSABL  LSB
```

7646  
7647  
7648  
7649  
7650  
7651  
7652 045260  
7653 045266 012702 177640  
7654 045272 012701 000020  
7655 045276 000413  
7656  
7657 045300  
7658 045306 012701 000020  
7659 045312 000404  
7660  
7661 045314  
7662 045322 012501  
7663 045324 012502  
7664 045326 012500  
7665  
7666 045330 012022  
7667 045332 077102  
7668 045334  
7669 045342 000205  
7670

7672  
7673  
7674 045344  
  
7675 045344 104415  
7676 045346  
7677  
7678 045352  
7679 045366  
7680 045372 104416  
7681 045374 000207  
7682 045376  
7683 045376 005737 002514  
7684 045402 001402  
7685 045404  
7686 045410  
7687 045420 104424  
7688 045422  
7689 045430  
7690 045434 104414  
7691 045436  
7692 045440 020027 000022  
7693 045444 101403  
7694 045446  
7695 045452 000766  
7696 045454  
7697 045464 045542  
7698 045466 045644  
7699 045470 045754  
7700 045472 046122  
7701 045474 046376  
7702 045476 046716  
7703 045500 047540  
7704 045502 047546  
7705 045504 050040  
7706 045506 050244  
7707 045510 050536  
7708 045512 050564  
7709 045514 050606  
7710 045516 050626  
7711 045520 050650  
7716 045522 050666  
7717 045524 050752  
7718 045526 051014  
7719 045530 051030  
7720 045532  
7721 045540 000733

```
.SBTTL FIELD SERVICE MODE
FIELDSERVICE:SUBTST <<SUBR FIELD SERVICE COMMAND MODE>>
:*****
:*SUBTEST SUBR FIELD SERVICE COMMAND MODE
:*****
SAVREG
TYPE MSG020 ;FIELD SERVICE COMMAND MODE
IF RLFLAG IS TRUE OR NOFSMODE IS TRUE
TYPE MSG048 ;NOT AVAILABLE NOW - TRY LATER!
RESREG
RETURN
END ;OF IF RLFLAG
TST CACHKN
BEQ 1$
PUSH CONTRL ;SAVE CACHE STATUS
1$: PUSH CSRNO,KAMIKAZE ;SAVE CSR & KAMIKAZE STATUS
CACHOFF ;TURN CACHE OFF
FS1: SET KAMIKAZE
TYPE MSG026 ;COMMAND:
RDDEC ;READ A DECIMAL NUMBER
POP RO ;COMMAND --> RO
CMP RO,#18.
BLOS 1$
TYPE MSG021
BR FS1
1$: CASE RO
FSCMD0 ;EXIT FIELD SERVICE COMMANDS
FSCMD1 ;READ CSR
FSCMD2 ;LOAD CSR
FSCMD3 ;EXAMINE MEMORY
FSCMD4 ;MODIFY MEMORY
FSCMD5 ;SELECT BANK & PATTERN
FSCMD6 ;TYPE CONFIGURATION MAP
FSCMD7 ;SOB-A-LONG TEST
FSCMD8 ;ERROR SUMMARY
FSCMD9 ;REFRESH TEST
FCMD10 ;SET FILL COUNT
FCMD11 ;ENTER KAMIKAZE MODE
FCMD12 ;EXIT KAMIKAZE MODE
FCMD13 ;TURN CACHE OFF
FCMD14 ;TURN CACHE ON
FCMD15 ;TEST ONLY SELECTED BANKS
FCMD16 ;RESUME TESTING ALL BANKS
FCMD17 ;ENABLE TRACE
FCMD18 ;DISABLE TRACE
END ;OF CASE
BR FS1
```

7724 045542

```
FSCMD0: SUBTST <<COMMAND 0 EXIT>>
:*****
:*SUBTEST COMMAND 0 EXIT
:*****
```

7725 045542  
 7726 045546 062706 000002  
 7727 045552  
 7728 045560 062706 000002  
 7729 045564 005037 002006  
 7730 045570  
 7731 045572  
 7732 045576  
 7733 045576  
 7734 045602 005737 002514  
 7735 045606 001414  
 7736 045610  
 7737 045620 062706 000002  
 7738 045624  
 7739 045626 005737 002514  
 7740 045632 001402  
 7741 045634  
 7742 045640  
 7743 045640 104416  
 7744 045642 000207  
 7745  
 7746 045644

```
TYPE MSG103 ;LEAVING FIELD SERVICE MODE
ADD #2,SP
IF SKIPKAMI IS TRUE
ADD #2,SP ;THROW AWAY OLD KAMIKAZE FLAG
CLR SKIPKAMI
ELSE
POP KAMIKAZE ;RESTORE OLD KAMIKAZE FLAG
END ;OF IF SKIPKAMI
POP CSRNO
TST CACHKN
BEQ RES0
IF CACHKN EQ CACHKF ;IF CACHE IS OFF
ADD #2,SP ;THROW AWAY CACHE STATUS
ELSE
TST CACHKN
BEQ RES0
POP CONTRL ;RESTORE CACHE STATUS
END ;OF IF CACHKN
RES0: RESREG
RETURN
```

```
FSCMD1: SUBTST <<FS COMMAND 1 READ CSR>>
:*****
:*SUBTEST FS COMMAND 1 READ CSR
:*****
```

7747 045644 004737 051042  
 7748 045650 010637 002266  
 7749 045654  
 7750 045662 104426  
 7751 045664  
 7752 045672 104026  
 7753 045674  
 7754 045716 000207  
 7755 045720  
 7756 045724 013706 002266  
 7757 045730  
 7758 045752 000207

```
CALL WHICHCSR
MOV SP,FSSTACK
SET4 #RES1 ;TRAPS TO 4 GOTO RES1
READCSR
SET NOERROR
ERROR +26 ;USE ERROR ROUTINE FOR PRINTOUT
RES4 ;RESET TRAPS TO 4 TO DEFAULT
RETURN
RES1: TYPE MSG025 ;THIS CSR DOES NOT EXIST
MOV FSSTACK,SP
RES4 ;RESET TRAPS TO 4 TO DEFAULT
RETURN
```

7761 045754

```
FSCMD2: SUBTST <<FS COMMAND 2 LOAD CSR>>
:*****
:*SUBTEST FS COMMAND 2 LOAD CSR
:*****
```

```
7762 045754 004737 051042
7763 045760 010637 002266
7764 045764
7765 045772 104426
7766 045774
7767 046000
7768 046006 104026
7769 046010
7770 046032
7771 046036 104413
7772 046040
7773 046044 104425
7774 046046 104426
7775 046050
7776 046054
7777 046062 104026
7778 046064 000207
7779 046066
7780 046072 013706 002266
7781 046076
7782 046120 000207
```

```
CALL WHICHCSR
MOV SP,FSSTACK
SET4 #RES2 ;TRAPS TO 4 GOTO RES2
READCSR
TYPE MSG027
SET NOERROR
ERROR +26 ;USE ERROR ROUTINE FOR PRINTOUT
RES4 ;RESET TRAPS TO 4 TO DEFAULT
TYPE MSG023 ;FIRST CSR WORD
RDOCT ;READ AN OCTAL NUMBER
POP CSR ;PUT IN IN LOC 'CSR'
LOADCSR
READCSR
TYPE MSG028
SET NOERROR
ERROR +26 ;USE FOR PRINTOUT - NOT AN ERROR
RETURN
RES2: TYPE MSG025 ;THIS CSR DOES NOT EXIST
MOV FSSTACK,SP
RES4 ;RESET TRAPS TO 4 TO DEFAULT
RETURN
```

7785 046122

```

FSCMD3: SUBTST <<FS COMMAND 3 EXAMINE MEMORY>>
:*****
:*SUBTEST FS COMMAND 3 EXAMINE MEMORY
:*****
7786 046122 PUSH BANK,NOPAR,PARTHERE,4
7787 046142 012737 000002 002074 MOV #2,NOPAR ;INDICATE PARITY ACTION
7788 046150 TYPE MSG029 ;EXAMINE MEMORY
7789 046154 1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-17775776)??
7790 046160 104413 RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
7791 046162 013737 061766 002100 MOV $HIOCT,BANK ;PUT MSB'S IN BANK
7792 046170 POP RO ;PUT LSB'S IN RO
7793 046172 000241 CLC
7794 046174 006100 ROL RO
7795 046176 006137 002100 ROL BANK
7796 046202 000241 CLC
7797 046204 006000 ROR RO
7798 046206 023737 002100 002526 CMP BANK,LASTBANK ;CHECK FOR BANK TOO HIGH
7799 046214 003357 BGT 1$ ;BRANCH IF TRUE
7800 046216 062700 060000 ADD #FIRST,RO
7801 046222 032700 000001 BIT #BIT0,RO ;CHECK FOR ODD ADDRESS
7802 046226 001352 BNE 1$ ;BRANCH IF ODD ADDRESS
7803 046230 020027 157776 CMP RO,#LAST ;CHECK FOR ADDRESS OVER 16K
7804 046234 101347 BHI 1$ ;BRANCH IF OVER 16K
7805 046236 012737 046310 002264 MOV #3$,PARTHERE ;INCASE OF ABORTS
7806 046244 SET4 #4$ ;TRAPS TO 4 GOTO 4$
7807 046252 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7808 046266 TESTAREA ;ENTER TEST MODE
7809 046274 011001 MOV (RO),R1
7810 046276 104417 KERNEL ;ENTER KERNEL MODE
7811 046300 TYPOCS R1
7812 046306 000410 BR EXCMD3
7813
7814 046310 3$: TYPE MSG032 ;PARITY ABORT
7815 046314 000405 BR EXCMD3
7816
7817 046316 062706 000004 4$: ADD #4,SP ;FIX STACK
7818 046322 TYPE MSG033 ;TIMEOUT TRAP
7819 046326 000400 BR EXCMD3
7820
7821 046330 104417 EXCMD3: KERNEL ;ENTER KERNEL MODE
7822 046332 POP 4,PARTHERE,NOPAR,BANK
7823 046352 RES4 ;RESET TRAPS TO 4 TO DEFAULT
7824 046374 000207 RETURN
  
```

7827 046376

FSCMD4: SUBTST <<FS COMMAND 4 MODIFY MEMORY>>  
 ;\*\*\*\*\*  
 ;\*SUBTEST FS COMMAND 4 MODIFY MEMORY  
 ;\*\*\*\*\*

```

7828 046376
7829 046416 012737 000003 002074
7830 046424
7831 046430
7832 046434 104413
7833 046436 013737 061766 002100
7834 046444
7835 046446 000241
7836 046450 006100
7837 046452 006137 002100
7838 046456 000241
7839 046460 006000
7840 046462
7841 046472 062700 060000
7842 046476
7843 046504
7844 046512 012737 046560 002264
7845 046520
7846 046526
7847 046542 104511
7848 046544
7849 046552 011001
7850
7851 046554 104417
7852 046556 000410
7853
7854 046560 3$:
7855 046564 000431
7856
7857 046566 062706 000004 4$:
7858 046572
7859 046576 000424
7860
7861 046600 5$:
7862 046604
7863 046612
7864 046616 104413
7865 046620
7866 046622
7867 046630 010110
7868 046632 011001
7869 046634 104417
7870 046636
7871 046642
7872
7873 046650 104417
7874 046652
7875 046672
7876 046714 000207
  
```

```

PUSH BANK,NOPAR,PARTHERE,4
MOV #3,NOPAR ;INDICATE PARITY ACTION
TYPE MSG036 ;MODIFY MEMORY
1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-17775776)??
RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
MOV $HIOCT,BANK ;PUT MSB'S IN BANK
POP RO ;PUT LSB'S IN RO
CLC
ROL RO
ROL BANK
CLC
ROR RO
IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
ADD #FIRST,RO
IF #BIT0 SET IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
IF RO HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER 16K
MOV #3$,PARTHERE ;INCASE OF ABORTS
SET4 #4$ ;TRAPS TO 4 GOTO 4$
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
INVALIDATE
TESTAREA ;ENTER TEST MODE
MOV (RO),R1
;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
KERNEL ;ENTER KERNEL MODE
BR 5$

3$: TYPE MSG032 ;PARITY ABORT
BR EXCMD4 ;EXIT

4$: ADD #4,SP ;FIX STACK
TYPE MSG033 ;TIMEOUT TRAP
BR EXCMD4 ;EXIT

5$: TYPE MSG037 ;OLD DATA WAS
TYPOCS R1 ;PRINT IT
TYPE MSG039 ;INPUT NEW DATA
RDOCT ;READ ON OCTAL NUMBER ONTO THE STACK
POP R1 ;GET NEW NUMBER
TESTAREA ;ENTER TEST MODE
MOV R1,(RO) ;PUT IT IN MEMORY
MOV (RO),R1 ;READ IT AGAIN
KERNEL ;ENTER KERNEL MODE
TYPE MSG038 ;DATA IS NOW
TYPOCS R1 ;PRINT IT

EXCMD4: KERNEL ;ENTER KERNEL MODE
POP 4,PARTHERE,NOPAR,BANK
RES4 ;RESET TRAPS TO 4 TO DEFAULT
RETURN
  
```



C 7

7879 046716  
 7880 046716  
 7881 046746 010637 002266  
 7882 046752  
 7883 046756  
 7884 046762 104413  
 7885 046764  
 7886 046770  
 7887  
 7888 047000 013701 002100  
 7889 047004 006301  
 7890 047006 006301  
 7891 047010  
 7892 047020  
 7893 047024  
 7894 047026  
 7895  
 7896 047026 104413  
 7897 047032  
 7898 047034  
 7899 047040  
 7900 047050  
 7901 047056  
 7902 047062 104413  
 7903 047064  
 7904 047066  
 7905  
 7906  
 7907 047066 104511  
 7908 047102 004737 044240  
 7909 047104  
 7910 047110  
 7911 047116  
 7912 047122  
 7913 047124  
 7914 047124  
 7915 047130 012737 047450 000060  
 7916 047136 012737 000340 000062  
 7917 047144 017700 133434  
 7918 047150 042737 000200 177776  
 7919 047156 052777 000100 133416  
 7920  
 7921  
 7922 047164 013701 002100  
 7923 047200 006301  
 7924 047204 006301  
 7925 047206 006301  
 7926 047210 005037 002232  
 7927 047214 005037 002256  
 7928 047220 012737 060000 002362  
 7929 047226 012737 060002 002364  
 7930 047234  
 7931 047244 005237 002232  
 7932 047250

```

FSCMD5: SUBTST <<FS COMMAND 5 SELECT BANK & PATTERN>>
:*****
:*SUBTEST FS COMMAND 5 SELECT BANK & PATTERN
:*****
PUSH BANK,PATTERN,TESTADD,PCBUMP,TKVEC,TKVEC+2
MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
TYPE MSG040 ;SELECT BANK & PATTERN TEST
1$: TYPE MSG030 ;BANK(0-17)?
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP BANK ;PUT IT IN BANK
IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH

MOV BANK,R1
ASL R1
ASL R1
IF (PUBIT OFF.IN CONFIG(R1)
TYPE MSG041 ;BANK NOT ACCESSABLE
GOTO 1$
END ;OF IF

2$: TYPE MSG042 ;PATTERN(0-35)?
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP PATTERN ;PUT IT IN PATTERN
IF PATTERN GT #35 THEN GOTO 2$ ;CHECK FOR PATTERN TOO HIGH
IF PATTERN EQ #0
TYPE MSG043 ;PATTERN 0 DATA IS?
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP R2 ;PUT IT IN R2
END ;OF IF

MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
INVALIDATE
CALL EXBANK ;SET NEW MARGINS
IF RRFLAG IS TRUE
TYPE MSG049 ;BANK REQUIRES RELOCATION
GOTO CMD5C
END ;OF IF RRFLAG
TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
MOV #CMD5C,TKVEC
MOV #340,TKVEC+2
MOV @STKB,R0 ;KILL ANY OLD INTERRUPT
BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
BIS #BIT6,@STKS ;ENABLE KEYBOARD INTERRUPTS

CMD5B: SET HEADER,MUT
MOV BANK,R1
ASL R1
ASL R1
CLR SPLTCSR
CLR PASFLG
MOV #FIRST,TESTADD
MOV #FIRST+2,TESTADD+2
IF #BIT12 SET.IN CONFIG+2(R1)
INC SPLTCSR
MAP BANK
  
```

```

7933 047264 012737 120000 002364      MOV #120000,TESTADD+2
7934 047272      END: OF IF #BIT12
7935 047272      IF #SWO SET.IN @SWR
7936 047302 104470      ECCDIS ;DISABLE ERROR CORRECTION
7937 047304      ELSE
7938 047306      PUSH CSRNO
7939 047312 104502      CLRCSR ;CLEAR CSRS
7940 047314      POP CSRNO
7941 047320      END :OF IF
7942 047320 012737 000002 002074      MOV #2,NOPAR ;PARITY ACTION
7943 047326 012737 000002 002276      MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
7944 047334 013700 002110      MOV PATTERN,RO
7945 047340 006300      ASL RO
7946 047342 004770 047354      CALL @FSPAT(RO)
7947 047346 005037 002074      CLR NOPAR
7948 047352 000712      BR CMD5B ;LOOP TILL KEYBOARD INTERRUPT
7949
7950 047354 020062      FSPAT: MT0000 ;<1 SEC DATA PATTERN TEST
7951 047356 020142      MT0001 ;<1 SEC ADDRESS TEST
7952 047360 020262      MT0002 ;<1 SEC COMPLEMENT ADDRESS TEST
7953 047362 020422      MT0003 : 1 SEC 3 XOR 9 WORST CASE NOISE TEST
7954 047364 020654      MT0004 : 1 SEC ROTATING ZEROS TEST
7955 047366 020776      MT0005 : 1 SEC ROTATING ONES TEST
7956 047370 021132      MT0006 ;<1 SEC INITIAL DATA TEST
7957 047372 021166      MT0007 ;<1 SEC ADDRESS BIT TEST
7958 047374 021230      MT0010 ;<1 SEC BYTE ADDRESSING TEST
7959 047376 021264      MT0011 ;<2 SEC CREATE SINGLE BIT ERROR TEST
7960 047400 021332      MT0012 ;<1 SEC WRITE BYTE CLEARS SBE TEST
7961 047402 021426      MT0013 : 1 SEC CREATE DOUBLE BIT ERROR TEST
7962 047404 021502      MT0014 : 1 SEC WRITE INHIBIT DURING DATIP WITH DBE
7963 047406 021560      MT0015 : 1 SEC WRITE INHIBIT OF BYTE WITH DBE
7964 047410 021626      MT0016 ;<1 SEC WRITE INHIBIT OF WORD WITH DBE
7965 047412 021674      MT0017 ;<1 SEC HOLDING 1'S & 0'S TEST
7966 047414 021716      MT0020 ;<1 SEC MARCHING 1'S & 0'S IN CHECK BITS
7967 047416 023006      MT0021 : 1 SEC MARCHING 0'S & 1'S TEST
7968 047420 023260      MT0022 :10 SEC REFRESH & SHIFTING DIAGONAL TEST
7969 047422 023312      MT0023 :10 SEC SHIFTING DIAGONAL TEST
7970 047424 023356      MT0024 :20 SEC FAST GALLOPING PATTERN TEST
7971 047426 023622      MT0025 ;<1 SEC INTERRUPT ENABLE TEST
7972 047430 023670      MT0026 ;<1 SEC RANDOM DATA TEST
7973 047432 024172      MT0027 : 1 SEC UNIQUE BANK TEST
7974 047434 024656      MT0030 : 1 SEC FLUSH OUT DBE'S TEST
7975 047436 025160      MT0031 : 3 SEC SOB-A-LONG TEST
7976 047440 025350      MT0032 ;<1 SEC WRITE RECOVERY TEST
7977 047442 025702      MT0033 :35 SEC BRANCH GOBBLE TEST
7978 047444 026070      MT0034 : 1 SEC SOFT ERROR TEST
7979 047446 026242      MT0035 ;<1 SEC WORST CASE NOISE PARITY TEST
7980
7981 047450 013706 002266      CMD5C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
7982 047454 042777 000100 133120      BIC #BIT6,@STKS
7983 047462      POP TKVEC+2,TKVEC
7984 047472 117700 133106      MOV @STKB,RO ;GET CHARACTER TO GET RID OF FLAG
7985 047476      POP PCBUMP,TESTADD
7986 047506      POP PATTERN,BANK
7987 047516      MAP BANK ;REMAP OLD BANK
7988 047532 004737 044240      CALL EXBANK
7989 047536 000207      RETURN
    
```

7991 047540

FSCMD6: SUBTST <<FS COMMAND 6 TYPE CONFIGURATION MAP>>  
:.....  
:\*SUBTEST FS COMMAND 6 TYPE CONFIGURATION MAP  
:.....

7992 047540 004737 036570

7993 047544 000207

7994

CALL PCONFIG  
RETURN

7997 047546

FSCMD7: SUBTST <<FS COMMAND 7 SOB-A-LONG TEST>>  
 :\*\*\*\*\*  
 :\*SUBTEST FS COMMAND 7 SOB-A-LONG TEST  
 :\*\*\*\*\*

7998 047546  
 7999 047572 010637 002266  
 8000 047576  
 8001  
 8002 047602  
 8003 047612 104470  
 8004 047614  
 8005 047616 104502  
 8006 047620  
 8007 047620  
 8008  
 8009 047624  
 8010 047630 012737 047754 000060  
 8011 047636 012737 000340 000062  
 8012 047644 017700 132734  
 8013 047650 042737 000200 177776  
 8014 047656 052777 000100 132716  
 8015  
 8016  
 8017 047664  
 8018  
 8019 047700  
 8020 047704 004737 044240  
 8021 047710  
 8022 047724 10451;  
 8023 047726 004737 025160  
 8024 047732  
 8025 047732  
 8026 047746  
 8027 047752  
 8028  
 8029 047754 013706 002266  
 8030 047760 042777 000100 132614  
 8031 047766 117700 132612  
 8032 047772  
 8033 050016  
 8034 050032 004737 044240  
 8035 050036 000207

PUSH BANK,PATTERN,TKVEC,TKVEC+2,NOPAR  
 MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER  
 TYPE MSG055 ;SOB-A-LONG TEST  
  
 IF #SWO 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 #CMD7C,TKVEC  
 MOV #340,TKVEC+2  
 MOV @STKB,RO ;KILL ANY OLD INTERRUPT  
 BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140  
 BIS #BIT6,@STKS ;ENABLE KEYBOARD INTERRUPTS  
  
 SET HEADER,MUT  
  
 CMD7B: FOR BANK := #0 TO LASTBANK  
 CALL EXBANK  
 IF ACFLAG IS TRUE AND RRFLAG IS FALSE  
 INVALIDATE  
 CALL MT0031  
 END ;OF IF ACFLAG  
 END ;OF FOR BANK  
 TYPE \$BELL ;RING BELL  
 GOTO CMD7B  
  
 CMD7C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER  
 BIC #BIT6,@STKS  
 MOV @STKB,RO ;READ CHAR TO KILL FLAG  
 POP NOPAR,TKVEC+2,TKVEC,PATTERN,BANK  
 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK  
 CALL EXBANK  
 RETURN

8038 050040

FSCMD8: SUBTST <<FS COMMAND 8 ERROR SUMMARY>>  
 :\*\*\*\*\*  
 :\*SUBTEST FS COMMAND 8 ERROR SUMMARY  
 :\*\*\*\*\*

8039 050040  
 8040 050052 013737 062474 002404  
 8041 050060 005337 002404  
 8042 050064  
 8043 050072  
 8044 050076  
 8045 050104  
 8046 050110  
 8047 050116 005037 002304  
 8048 050122  
 8049 050126 013703 002100  
 8050 050132 070327 000004  
 8051 050136  
 8052 050144  
 8053 050152  
 8054 050156  
 8055 050164  
 8056 050164  
 8057 050174 116300 002626  
 8058 050200 042700 177400  
 8059 050204  
 8060 050210  
 8061 050214  
 8062 050214  
 8063 050230  
 8064 050230  
 8065 050242 000207

```

PUSH  R0,R2,R3,BANK
MOV   $PASS,TEMP
DEC   TEMP
TYPDEC TEMP
TYPE  MSG125           ;PASSES COMPLETED
TYPDEC $ERTTL
TYPE  MSG079           ;ERROR(S) DETECTED
IF $ERTTL NE #0
CLR   SUCCESS
FOR BANK := #0 TO LASTBANK
MOV  BANK,R3
MUL  #4,R3
IFB CONFIG+2(R3) NE #0
IF SUCCESS IS FALSE
TYPE  MSG076           ;BANK ERRORS
SET SUCCESS
END ;OF IF SUCCESS
TYPCS BANK,3
MOVB  CONFIG+2(R3),R0
BIC   #^C377,R0
TYPDEC R0
TYPE  $CRLF
END ;OF IFB CONFIG(R3)
END ;OF FOR BANK
END ;OF IF $ERTTL
POP   BANK,R3,R2,R0
RETURN
    
```

```

8068 050244 FSCMD9: SUBTST <<FS COMMAND 9 REFRESH TEST>>
:*****
:+SUBTEST FS COMMAND 9 REFRESH TEST
:*****
8069 050244 PUSH BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
8070 050270 010637 002266 MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
8071 050274 TYPE MSG073 ;REFRESH TEST
8072
8073 050300 IF #SWO SET.IN @SWR
8074 050310 104470 ECCDIS ;DISABLE ERROR CORRECTION
8075 050312 ELSE
8076 050314 104502 CLRCSR ;CLEAR CSRS
8077 050316 END ;OF IF
8078 050316 TYPE MSG056 ;BELL = EACH PASS COMPLETE
8079
8080 050322 TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
8081 050326 012737 050452 000060 MOV #CMD9C,TKVEC
8082 050334 012737 000340 000062 MOV #340,TKVEC+2
8083 050342 017700 132236 MOV @TKB,RO ;KILL ANY OLD INTERRUPT
8084 050346 042737 000200 177776 BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
8085 050354 052777 000100 132220 BIS #BIT6,@TKS ;ENABLE KEYBOARD INTERRUPTS
8086
8087 050362 SET HEADER,MUT
8088
8089 050376 CMD9B: FOR BANK := #0 TO LASTBANK
8090 050402 004737 044240 CALL EXBANK
8091 050406 IF ACFLAG IS TRUE AND RRFLAG IS FALSE
8092 050422 104511 INVALIDATE
8093 050424 004737 023260 CALL MTO022
8094 050430 END ;OF IF ACFLAG
8095 050430 END ;OF FOR BANK
8096 050444 TYPE $BELL ;RING BELL
8097 050450 GOTO CMD9B
8098
8099 050452 013706 002266 CMD9C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
8100 050456 042777 000100 132116 BIC #BIT6,@TKS
8101 050464 117700 132114 MOV @TKB,RO ;READ CHAR TO KILL FLAG
8102 050470 POP NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
8103 050514 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
8104 050530 004737 044240 CALL EXBANK
8105 050534 000207 RETURN
8106
  
```

```

8109 050536 FCMD10: SUBTST <<FS COMMAND 10 SET FILL COUNT>>
:*****
:*SUBTEST FS COMMAND 10 SET FILL COUNT
:*****
8110 050536 PUSH RO
8111 050540 TYPE MSG085 ;FILL COUNT(OCTAL)?
8112 050544 104413 RDOCT
8113 050546 POP RO
8114 050550 042700 177760 BIC #^C17,RO
8115 050554 110037 002327 MOVB RO,$FILLS
8116 050560 POP RO
8117 050562 000207 RETURN
8118
8119 050564 FCMD11: SUBTST <<FS COMMAND 11 ENTER KAMIKAZE MODE>>
:*****
:*SUBTEST FS COMMAND 11 ENTER KAMIKAZE MODE
:*****
8120 050564 TYPE MSG101 ;ENTERING KAMIKAZE MODE
8121 050570 SET KAMIKAZE,SKIPKAMI
8122 050604 000207 RETURN
8123
8124 050606 FCMD12: SUBTST <<FS COMMAND 12 EXIT KAMIKAZE MODE>>
:*****
:*SUBTEST FS COMMAND 12 EXIT KAMIKAZE MODE
:*****
8125 050606 TYPE MSG102 ;LEAVING KAMIKAZE MODE
8126 050612 005037 002004 CLR KAMIKAZE
8127 050616 SET SKIPKAMI
8128 050624 000207 RETURN
8129
8130 050626 FCMD13: SUBTST <<FS COMMAND 13 TURN CACHE OFF>>
:*****
:*SUBTEST FS COMMAND 13 TURN CACHE OFF
:*****
8131 050626 TYPE MSG106 ;CACHE IS OFF
8132 050632 104424 CACHOFF ;TURN CACHE OFF
8133 050634 013737 002514 002516 MOV CACHKN,CACHKN+2 ;SAVE OLD CACHE ON STATE
8134 050642 005037 002514 CLR CACHKN ;KEEP CACHE OFF
8135 050646 000207 RETURN
8136
8137 050650 FCMD14: SUBTST <<FS COMMAND 14 TURN CACHE ON>>
:*****
:*SUBTEST FS COMMAND 14 TURN CACHE ON
:*****
8138 050650 TYPE MSG107 ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
8139 050654 013737 002516 002514 MOV CACHKN+2,CACHKN ;RESTORE OLD CACHE ON STATE
8140 050662 104423 CACHON ;TURN CACHE ON
8141 050664 000207 RETURN
8142
  
```

8155  
 8156 050666  
 8157 050666  
 8158 050672 004737 050762  
 8159 050676  
 8160 050676  
 8161 050676  
 8162 050702 104413  
 8163 050704  
 8164 050706  
 8165 050720  
 8166 050722  
 8167 050722 006301  
 8168 050724 006301  
 8169 050726 052761 040000 002626  
 8170 050734  
 8171 050736  
 8172 050736  
 8173 050742  
 8174 050750 000207  
 8175  
 8176 050752

```

FCMD15: SUBTST <<FS COMMAND 15 TEST ONLY SELECTED BANKS>>
:*****
:*SUBTEST FS COMMAND 15 TEST ONLY SELECTED BANKS
:*****
TYPE MSG105 ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
CALL CMD16A ;ERASE OLD SELECTIONS
BEGIN CMD16LOOP
REPEAT
TYPE MSG030 ;BANK(0-177)?
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP R1 ;PUT IT IN R1
IF R1 GT #177 OR R1 LT #0
LEAVE CMD16LOOP
END ;OF IF R1
ASL R1
ASL R1 ;R1 <- R1 * 4
BIS #BIT14,CONFIG+2(R1)
END ;OF REPEAT
END CMD16LOOP
TYPE MSG110 ;ONLY SELECTED BANKS WILL BE TESTED
SET SELONLY
RETURN
  
```

8177 050752  
 8178 050756 005037 002000  
 8179  
 8180  
 8181 050762 013702 002526  
 8182 050766 006302  
 8183 050770 006302  
 8184 050772  
 8185 050774 042761 040000 002626  
 8186 051002  
 8187 051012 000207

```

FCMD16: SUBTST <<FS COMMAND 16 RESUME TESTING ALL BANKS>>
:*****
:*SUBTEST FS COMMAND 16 RESUME TESTING ALL BANKS
:*****
TYPE MSG111 ;ALL BANKS WILL BE TESTED
CLR SELONLY

;ENTRY POINT FROM CMD15
CMD16A: MOV LASTBANK,R2
ASL R2
ASL R2
FOR R1 := #0 TO R2 BY #4
BIC #BIT14,CONFIG+2(R1)
END ;OF FOR R1
RETURN
  
```



8190 051014

FCMD17: SUBTST <<FS COMMAND 17 ENABLE TRACE>>  
:\*\*\*\*\*  
:\*SUBTEST FS COMMAND 17 ENABLE TRACE  
:\*\*\*\*\*

8191 051014  
8192 051020 012737 177777 006'04  
8193 051026 000207

TYPE MSG127  
MOV #-1,TRACE  
RETURN

8196 051030

FCMD18: SUBTST <<FS COMMAND 18 DISABLE TRACE>>  
:\*\*\*\*\*  
:\*SUBTEST FS COMMAND 18 DISABLE TRACE  
:\*\*\*\*\*

8197 051030  
8198 051034 005037 006104  
8199 051040 000207

TYPE MSG128  
CLR TRACE  
RETURN

8202 051042

WHICHCSR:SUBTST <<SUBR DETERMINE CORRECT CSR>>

\*\*\*\*\*  
 :\*SUBTEST SUBR DETERMINE CORRECT CSR  
 :\*\*\*\*\*

8203 051042 013700 002216  
 8204 051046 022700 100000  
 8205 051052 001003  
 8206 051054 005037 002146  
 8207 051060 000207  
 8208  
 8209 051062  
 8210 051066 104412  
 8211 051070  
 8212 051072 011000  
 8213 051074 020027 000106  
 8214 051100 101370  
 8215 051102 022700 000101  
 8216 051106 103002  
 8217 051110 162700 000007  
 8218 051114 162700 000060  
 8219 051120 006300  
 8220 051122 010037 002146  
 8221 051126 000207

```

MOV TOTCSRS,RO ;GET CSR'S FLAG
CMP #BIT15,RO ;CSR 0?
BNE 1$ ;NO - SKIP
CLR CSRNO ;YES - SET IT UP
RETURN

1$: TYPE MSG022 ;WHICH CSR(0-F)
RDLIN ;GET CHARACTER
POP RO ;PUT IN RO
MOV (RO),RO ;PUT CHAR IN RO
CMP RO,#106 ;CHECK LIMIT
BHI 1$ ;IF BAD LOOP TILL HE TYPES IT RIGHT
CMP #'A,RO
BHIS 2$
SUB #7,RO
2$: SUB #60,RO
ASL RO
MOV RO,CSRNO
RETURN
  
```

8770  
 8771 051130  
 8772 051142  
 8773 051150  
 8774 051156  
 8775 051164 104417  
 8776 051166  
 8777 051166  
 8778 051174  
 8779 051170  
 8780 051202  
 8781 051206  
 8782 051206 000137 054410  
 8783  
 8784 051212  
  
 8785 051212  
 8786 051224  
 8787 051226 005037 002144  
 8788 051232 104505  
 8789 051234  
 8790 051242 005711  
 8791 051244 104417  
 8792 051246 104426  
 8793 051250 013700 002144  
 8794 051254 104503  
 8795 051256 072027 177773  
 8796 051262 042700 177600  
 8797 051266  
 8798 051272 005037 002042  
 8799 051276  
 8800 051304 011137 002050  
 8801 051310 011437 002052  
 8802 051314 104417  
 8803 051316 110037 002054  
 8804 051322 105037 002055  
 8805 051326 004737 054644  
 8806 051332 104033  
 8807 051334  
 8808 051336  
 8809 051346 104506  
 8810 051350  
 8811 051352 104472  
 8812 051354  
 8813 051354 000002

```

.SBTTL ERROR DATA (SUPERVISOR) SETUP STUFF
$PER25: LET ADDRESS := R1 - #2
        IF ABORTFLAG IS FALSE
            TESTAREA ;ENTER TEST MODE
            LET BAD := -2(R1)
            KERNEL ;ENTER KERNEL MODE
        END ;OF IF ABORTFLAG
        IF 177654 EQ #0
            LET GOOD := R2
        ELSE
            LET GOOD := R3
        END ;OF IF
        JMP PERRAW

PERRA3: SUBTST <<DATA WAS 3 WORDS>>
;*****
;*SUBTEST DATA WAS 3 WORDS
;*****
        IF BADPC EQ #0 THEN $CALL BADSTACK
        PUSH RO
        CLR CSR ;MAKE SURE CSR BIT HOLDER IS CLEAR
        CHK1DIS ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
        TESTAREA
        TST (R1) ;READ LOCATION TO READ CHECKBITS INTO CSR
        KERNEL
        READCSR ;GET CSR CONTENTS
        MOV CSR,RO ;SAVE CSR CONTENTS IN RO
        CLR1CSR ;RETURN CSR TO NORMAL MODE
        ASH #-5,RO ;MOVE CHECK BITS TO BOTTOM OF WORD
        BIC #^C177,RO ;CLEAR OFF EXTRANEIOUS GARBAGE
        LET ADDRESS := R1 ;SAVE VIRTUAL ADDRESS FOR PRINTOUT
        CLR GOOD ;FIRST TEST WORD WRITTEN SHOULD ALWAYS BE ZERO
        TESTAREA ;ENTER TEST MODE
        MOV (R1),BAD ;GET BAD DATA FROM MUT - FIRST WORD
        MOV (R4),BAD2 ;AND SECOND WORD
        KERNEL ;ENTER KERNEL MODE
        MOVB RO,BAD3 ;MOVE BAD CHECKBITS FOR PRINTOUT
        CLRB BAD3+1 ;CLEAR OFF THE OTHER UNUSED BITS
        CALL PERBNK ;MARK BANK AS BAD IN CONFIG TABLE
        ERROR +33
        POP RO ;RESTORE RO
        IF #SWO SET.IN @SWR
            ENASBE ;TRAP ON SINGLE BIT ERRORS
        ELSE
            ECCINIT ;TRAP ON UNCORRECTABLE ERRORS
        END; OF IF #SWO
        RTI
    
```

8816 051356  
8817 051362  
8818 051374  
8819 051402  
8820 051410  
8821 051416 104417  
8822 051420  
8823 051420 000137 054410  
8824  
8825 051424

```
$PER30: LET GOOD := R1
        LET ADDRESS := (SP) - 16
        IF ABORTFLAG IS FALSE
          TESTAREA ;ENTER TEST MODE
          LET BAD := @ADDRESS
          KERNEL ;ENTER KERNEL MODE
        END ;OF IF ABORTFLAG
        JMP PERRAW
```

```
GETDATA:SUBTST <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
:*****
:*SUBTEST GET DATA FROM ABORTED AREA IF POSSIBLE
:*****
```

8826 051424  
8827 051436 010637 051522  
8828 051442 012737 051502 000004  
8829 051450 012737 051502 000114  
8830 051456 013700 002032  
8831 051462  
8832 051470 011037 002050  
8833 051474 104417  
8834 051476 005037 002140  
8835 051502 013706 051522  
8836 051506  
8837 051520 000207  
8838 051522 000000

```
        PUSH RO,4,114
        MOV SP,GETDA1
        MOV #1$,4
        MOV #1$,114
        MOV ADDRESS,R0
        TESTAREA
        MOV (R0),BAD
        KERNEL
        CLR ABORTFLAG
1$: MOV GETDA1,SP ;RESTORE KNOWN GOOD STACK POINTER
    POP 114,4,R0
    RETURN
GETDA1: 0
```

```

8841                                     .SBTTL POWER FAIL AUTO RESTART
8842                                     .SBTTL ROUTINE POWER DOWN AND UP
8843 :*****
8844 :POWER DOWN ROUTINE
8845 $PWRDN:
8853 ;SAVE CACHE STATUS
8854 051524 005737 002514 TST CACHKN
8855 051530 001403 BEQ 5$
8856 051532 PUSH CONTRL
8857 051536 104423 CACHON ;TURN CACHE ON
8858 051540 012737 052476 000024 5$: MOV #5ILLUP,PWRVEC ;SET FOR FAST UP
8859 051546 012737 000340 000026 MOV #340,PWRVEC+2 ;:PRIO:7
8860 051554 PUSH RO,R1,R2,R3,R4,R5,CSRNO
8861 ;SAVE USER PAR'S & PDR?
8862 051574 012700 177700 MOV #177700,R0
8863 051600 012701 000021 MOV #17.,R1
8864 051604 1$: PUSH -(R0)
8865 051606 077102 SOB R1,1$
8866 ;SAVE SUPERVISOR PAR'S
8867 051610 005737 002426 TST NOSUPER
8868 051614 001013 BNE PD1
8869 051616 012700 172300 MOV #172300,R0
8870 051622 012701 000020 MOV #16.,R1
8871 051626 2$: PUSH -(R0)
8872 051630 077102 SOB R1,2$
8873 051632 IF RLFLAG IS TRUE THEN $CALL WOOPS
8874 ;COPY KERNEL MAP TO USER & SUPERVISOR
8875 051644 012700 172300 PD1: MOV #KIPDR0,R0
8876 051650 012701 177600 MOV #UIPDR0,R1
8877 051654 012702 172200 MOV #SIPDR0,R2
8878 051660 012703 000040 MOV #32.,R3
8879 051664 011021 3$: MOV (R0),(R1)+
8880 051666 012022 MOV (R0)+,(R2)+
8881 051670 077303 SOB R3,3$

```

```

8883 ;SAVE USER & SUPERVISOR STACK POINTERS
8884 051672 USER
8885 051700 010600 MOV USP,R0
8886 051702 104417 KERNEL ;ENTER KERNEL MODE
8887 051704 PUSH RO
8888 051706 005737 002426 TST NOSUPER
8889 051712 001006 BNE 7$
8890 051714 SUPERVISOR ;ENTER SUPERVISOR MODE
8891 051722 010600 MOV SSP,R0
8892 051724 104417 KERNEL ;ENTER KERNEL MODE
8893 051726 PUSH RO
8894 ;SAVE ECC REGISTERS
8895 051730 013701 002216 7$: MOV TOTCSRS,R1 ;GET CSR'S
8896 051734 BEGIN LCSRSAVE
8897 051734 FOR CSRNO := #0 TO #36 BY #2
8898 051740 006301 ASL R1
8899 051742 ON.ERROR
8900 051744 104426 READCSR
8901 051746 PUSH CSR
8902 051752 END ;OF ON.ERROR
8903 051752 IF R1 EQ #0 THEN LEAVE LCSRSAVE
8904 051756 END ;OF FOR CSRNO
8905 051774 END LCSRSAVE
8906 ;SAVE MMRO,1,2,3
8907 051774 PUSH MMRO,MMR1,MMR2
8908 052010 005737 002426 TST NOSUPER
8909 052014 001002 BNE 8$
8910 052016 PUSH MMR3
8911 ;SAVE KERNEL PAR'S
8912 052022 012700 172400 8$: MOV #172400,R0
8913 052026 012701 000020 MOV #16.,R1
8914 052032 4$: PUSH -(R0)
8915 052034 077102 SOB R1,4$
8916 ;SAVE UNIBUS MAP REGISTERS
8917 052036 022737 000001 003710 CMP #1,PROTYP ;IS THIS AN 11/44?
8918 052044 001004 BNE 9$ ;BRANCH IF NOT
8919 052046 PUSH MAPH0,MAPL0
8920 ;SAVE POSSIBLE SOFTWARE SWITCH REGISTER
8921 052056 9$: PUSH @SWR
8922 ;SAVE STACK POINTER
8923 052062 010637 052502 MOV SP,$SAVR6 ;;SAVE SP
8924 ;NOW SET UP REAL VECTOR
8925 052066 012737 052100 000024 MOV #$PWRUP,PWRVEC ;;SET UP VECTOR
8926 052074 000000 $DOWN: HALT
8927 052076 000776 BR $DOWN ;;HANG UP

```

```

8930 .....
8931 :POWER JP ROUTINE
8932 052100 $PWRUP:
8936 052100 012737 052476 000024 MOV #SILLUP,PWRVEC ;;SET FOR FAST DOWN
8937 :RESTORE STACK POINTER
8938 052106 013706 052502 MOV $SAVR6,SP ;;GET SP
8939 052112 005037 052502 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
8940 052116 005237 052502 1$: INC $SAVR6 ;;WAIT FOR THE INC
8941 052122 001375 BNE 1$ ;;OF A WORD
8942 :RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
8943 052124 POP @SWR
8944 :RESTORE UNIBUS MAP
8945 052130 022737 000001 003710 CMP #1,PROTYP ;IS THIS AN 11/44?
8946 052136 001006 BNE 10$
8947 052140 POP MAPLO,MAPHO
8948 052150 004737 043644 CALL LOWMAP ;SETUP LOWER 16K OF UNIBUS MAP
8949 :RESTORE KERNEL PAR'S & PGR'S
8950 052154 012700 172340 10$: MOV #172340,R0
8951 052160 012702 172300 MOV #KIPDRO,R2
8952 052164 012701 000020 MOV #16.,R1
8953 052170 6$: POP (R0)+
8954 052172 012722 077406 MOV #77406,(R2)+
8955 052176 077104 SOB R1,6$
8956 :RESTORE MMR3,2,1,0
8957 052200 005737 002426 TST NOSUPER
8958 052204 001002 BNE 11$
8959 052206 POP MMR3
8960 052212 11$: POP MMR2,MMR1,MMR0
8961 :RESTORE ECC REGISTERS
8962 052226 013701 002216 MOV TOTCSRS,R1 ;GET CSR'S
8963 052232 042701 177400 BIC #177400,R1
8964 052236 BEGIN LCSRRESTORE
8965 052236 FOR CSRNO := #36 DOWNT0 #0 BY #2
8966 052244 006201 ASR R1
8967 052246 ON.ERROR
8968 052250 POP CSR
8969 052254 104425 LOADCSR
8970 052256 END ;OF ON.ERROR
8971 052256 IF R1 EQ #0 THEN LEAVE LCSRRESTORE
8972 052262 END ;OF FOR CSRNO
8973 052300 END LCSRRESTORE
8974 :COPY KERNEL MAP TO USER & SUPERVISOR
8975 052300 012700 172300 MOV #KIPDRO,R0
8976 052304 012701 177600 MOV #UIPDRO,R1
8977 052310 012702 172200 MOV #SIPDRO,R2
8978 052314 012703 000040 MOV #32.,R3
8979 052320 011021 3$: MOV (R0),(R1)+
8980 052322 012022 MOV (R0)+,(R2)+
8981 052324 077303 SOB R3,3$

```



```

8983                                     ;RESTORE SUPERVISOR & USER STACK POINTERS
8984 052326 005737 002426               TST     NCSUPER
8985 052332 001006                       BNE     13$
8986 052334                               POP     R0
8987 052336                               SUPERVISOR ;ENTER SUPERVISOR MODE
8988 052344 010006                       MOV     R0,SSP
8989 052346 104417                       KERNEL ;ENTER KERNEL MODE
8990 052350                               POP     R0
8991 052352                               USER
8992 052360 010006                       MOV     R0,USP
8993 052362 104417                       KERNEL ;ENTER KERNEL MODE
8994                                     ;RESTORE SUPERVISOR PAR'S
8995 052364 012700 172240               MOV     #172240,R0
8996 052370 012701 000020               MOV     #16.,R1
8997 052374                               POP     (R0)+
8998 052376 077102                       SOB     R1,7$
8999                                     ;RESTORE USER PAR'S & PDR7
9000 052400 012700 177636               MOV     #177636,R0
9001 052404 012701 000021               MOV     #17.,R1
9002 052410                               POP     (R0)+
9003 052412 077102                       SOB     R1,8$
9004                                     ;RESTORE POSSIBLE SOFTWARE DISPLAY REGISTER
9005 052414 013777 002010 130156       MOV     $PATMAR,@DISPLAY
9006 052422 013737 002010 000174       MOV     $PATMAR,DISPREG
9007 052430                               POP     CSRNO,R5,R4,R3,R2,R1,R0
9008 052450 012737 051524 000024       MOV     #$PWRDN,PWRVEC ;:SET UP THE POWER DOWN VECTOR
9009 052456                               TYPE     MSG051 ;:REPORT THE POWER FAILURE
9010                                     ;RESTORE CACHE STATUS
9011 052462 005737 002514               TST     CACHKN
9012 052466 001402                       BEQ     9$
9013 052470                               POP     CONTRL
9014 052474 000002                       9$: RTI
9015 052476 000000                       $ILLUP: HALT ;:THE POWER UP SEQUENCE WAS STARTED
9016 052500 000776                       BR      $ILLUP ;: BEFORE THE POWER DOWN WAS COMPLETE
9017 052502 000000                       $SAVR6: 0 ;:PUT THE SP HERE
9018                                     .EVEN
  
```

9030 052504

WOOPS: SUBTST <<POWER FAIL WHILE RELOCATED>>  
:\*\*\*\*\*  
:\*SUBTEST POWER FAIL WHILE RELOCATED  
:\*\*\*\*\*

9031 052504  
9032 052510 005037 002100  
9033 052514  
9034 052530  
9035 052536 013737 060024 053102  
9036 052544 013737 060026 053104  
9037 052552  
9038 052564 012737 052670 060024  
9039 052572 012737 000340 060026  
9040 052600  
9041 052612 012700 172340  
9042 052616 012701 133052  
9043 052622 012702 000010  
9044 052626 012021  
9045 052630 077202  
9046 052632 005737 002426  
9047 052636 001002  
9048 052640 013721 172516  
9049 052644 013721 177576  
9050 052650 013721 177574  
9051 052654 013721 177572  
9052 052660 104417  
9053 052662  
9054 052666 000207

PUSH BANK  
CLR BANK  
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK  
SUPERVISOR ;ENTER SUPERVISOR MODE  
MOV FIRST+PWRVEC,WOOPSAV  
MOV FIRST+PWRVEC+2,WOOPSAV+2  
BMOV FIRST+WOOPUP,WOOPSAV+4,WOOPEN-WOOPUP/2+12.  
MOV #WOOPUP,FIRST+PWRVEC  
MOV #340,FIRST+PWRVEC+2  
BMOV WOOPUP,FIRST+WOOPUP,WOOPEN-WOOPUP/2  
MOV #KIPAR0,R0  
MOV #FIRST+WOOPEN,R1  
MOV #8,R2  
1\$: MOV (R0)+,(R1)+  
SUB R2,1\$  
TST NOSUPER  
BNE 2\$  
MOV MMR3,(R1)+  
2\$: MOV MMR2,(R1)+  
MOV MMR1,(R1)+  
MOV MMRO,(R1)+  
KERNEL ;ENTER KERNEL MODE  
POP BANK  
RETURN

9057 052670

WOOPUP: SUBTST <<POWER UP FROM BANK 0 TO RELOCATION>>  
:\*\*\*\*\*  
: \*SUBTEST POWER UP FROM BANK 0 TO RELOCATION  
:\*\*\*\*\*

9058 052670 012700 053052  
9059 052674 012701 172340  
9060 052700 012703 172300  
9061 052704 012702 000010  
9062 052710 012021  
9063 052712 012723 077406  
9064 052716 077204  
9065 052720 005737 002426  
9066 052724 001002  
9067 052726 012037 172516  
9068 052732 012037 177576  
9069 052736 012037 177574  
9070 052742 012037 177572  
9071 052746 013706 052502  
9072 052752  
9073 052756 005037 002100  
9074 052762  
9075 052776  
9076 053004 013737 053102 060024  
9077 053012 013737 053104 060026  
9078  
9079  
9080 053020 012700 053106  
9081 053024 012701 000105  
9082 053030 012702 132670  
9083 053034 012022  
9084 053036 077102  
9085  
9086 053040 104417  
9087 053042  
9088 053046 000137 052100  
9089 053052 000014  
9092 053102 000107

```

MOV #WOOPEND,R0
MOV #KIPAR0,R1
MOV #KIPDR0,R3
MOV #8,R2
1$: MOV (R0)+,(R1)+
    MCV #77406,(R3)+
    SCB R2,1$
    TST NOSUPER
    BNE 3$
3$: MOV (R0)+,MMR3
    MOV (R0)+,MMR2
    MOV (R0)+,MMR1
    MOV (R0)+,MMR0
    MOV $$SAVR6,SP
    PUSH BANK
    CLR BANK
    MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
    SUPERVISOR ;ENTER SUPERVISOR MODE
MOV WOOPSAV,FIRST+PWRVEC
MOV WOOPSAV+2,FIRST+PWRVEC+2
;SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
;BMOV WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
MOV #WOOPSAV+4,R0
MOV #WOOPEND-WOOPUP/2+12.,R1
MOV #FIRST+WOOPUP,R2
2$: MOV (R0)+,(R2)+
    SOB R1,2$

    KERNEL ;ENTER KERNEL MODE
    POP BANK
    JMP $PWRUP
WOOPEND: .REPT 12.
WOOPSAV: .REPT WOOPEND-WOOPUP/2+12.+2
    
```

```

9097          .SBTTL IO SUBROUTINES
9098
9099          .SBTTL ROUTINE TYPE
9100
9101          ;*****
9102          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
9103          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
9104          ;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
9105          ;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
9106          ;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
9107          ;*
9108          ;*CALL:
9109          ;*1) USING A TRAP INSTRUCTION
9110          ;*          TYPE      MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
9111          ;*OR
9112          ;*          TYPE
9113          ;*          MESADR
9114          ;*
9115
9116 053320 105737 002330 $TYPE: TSTB $TPFLG          ;;IS THERE A TERMINAL?
9117 053324 100407          BMI 6$          ;;BR IF NO
9118 053326 010046          1$: MOV RO,-(SP)          ;;SAVE RO
9119 053330 017600 000002          MOV @2(SP),RO          ;;GET ADDRESS OF ASCIZ STRING
9120 053334 112046          4$: MOV (RO)+,-(SP)          ;;PUSH CHARACTER TO BE TYPED ONTO STACK
9121 053336 001005          BNE 7$          ;;BR IF IT ISN'T THE TERMINATOR
9122 053340 005726          TST (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
9123 053342 012600          5$: MOV (SP)+,RO          ;;RESTORE RO
9124 053344 062716 000002          6$: ADD #2,(SP)          ;;ADJUST RETURN PC
9125 053350 000002          RTI          ;;RETURN
9126 053352 122716 000011          7$: CMPB #HT,(SP)          ;;BRANCH IF NOT <HT>
9127 053356 001002          BNE 11$
9128 053360 112716 000040          MOVB #' ,(SP)          ;;REPLACE TAB WITH SPACE
9129 053364 122716 000200          11$: CMPB #CRLF,(SP)          ;;BRANCH IF NOT <CRLF>
9130 053370 001006          BNE 8$
9131 053372 005726          TST (SP)+          ;;POP <CR><LF> EQUIV
9132 053374          TYPE          ;;TYPE A CR AND LF
9133 053376 002620          $CRLF
9134 053400 105037 053632          CLRB $CHARCNT          ;;CLEAR CHARACTER COUNT
9135 053404 000753          BR 4$          ;;GET NEXT CHARACTER
9136 053406 004737 053446          8$: CALL $TYPEC          ;;GO TYPE THIS CHARACTER
9137 053412 123726 002612          9$: CMPB $FILLC,(SP)+          ;;IS IT TIME FOR FILLER CHARS.?
9138 053416 001346          BNE 4$          ;;IF NO GO GET NEXT CHAR.
9139 053420 013746 002326          MOV $NULL,-(SP)          ;;GET # OF FILLER CHARS. NEEDED
9140          ;;AND THE NULL CHAR.
9141 053424 105366 000001          10$: DECB 1(SP)          ;;DOES A NULL NEED TO BE TYPED?
9142 053430 002770          BLT 9$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
9143 053432 004737 053446          CALL $TYPEC          ;;GO TYPE A NULL
9144 053436 105337 053632          DECB $CHARCNT          ;;DO NOT COUNT AS A COUNT
9145 053442 000770          BR 10$          ;;LOOP
9146 053444 000000          XOCHAR: .WORD 0
9147 053446          $TYPEC: PUSH R1
9148 053450 116601 000004          MOV 4(SP),R1
9149 053454 005737 002514          TST CACHKN
9150 053460 001402          BEQ 2$
9151 053462          PUSH CONTRL
9152 053466          2$: PUSH RO
9153 053470 104424          CACHOFF          ;;TURN CACHE OFF

```

```

9178 053472 105777 127110          3$:   TSTB  @STPS          ;;WAIT UNTIL PRINTER IS READY
9179 053476 100375                    BPL   3$
9180 053500 005037 053444          CLR   XOCHAR
9181 053504 105777 127072          TSTB  @STKS          ;;CHECK FOR XOFF
9182 053510 100032                    BPL   NC              ;;SKIP IF NO CHARACTER
9183 053512 117737 127066 053444        MOVB  @STKB,XOCHAR    ;;SAVE THE CHARACTER
9184 053520 042737 177600 053444        BIC   #^C177,XOCHAR  ;;STRIP OFF ASCII
9185 053526 023727 053444 000023        CMP   XOCHAR,#023    ;;WAS IT A CONTROL S?
9186 053534 001020                    BNE   NC              ;BRANCH IF NOT
9187 053536 105777 127040          CONTS3: TSTB @STKS          ;;WAIT FOR CHARACTER
9188 053542 100375                    BPL   CONTS3
9189 053544 117737 127034 053444        MOVB  @STKB,XOCHAR    ;;GET CHARACTER
9190 053552 042737 177600 053444        BIC   #^C177,XOCHAR  ;;STRIP OFF ASCII
9191 053560                    IF XOCHAR EQ #21    ;; IF IT IS A ^Q
9192 053570 000402                    BR    NC
9193 053572                    ELSE
9194 053574 000760                    BR    CONTS3
9195 053576                    END ;OF IF XOCHAR
9196 053576 110177 127006          NC:   MOVB  R1,@STPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
9200 053602 122766 000015 000002        CMPB  #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
9201 053610 001003                    BNE   1$             ;;BRANCH IF NO
9202 053612 105037 053632          CLRB  $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
9203 053616 000406                    BR    $TYPEX        ;;EXIT
9204 053620 122766 000012 000002        1$:  CMPB  #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
9205 053626 001402                    BEQ   $TYPEX        ;;BRANCH IF YES
9206 053630 105227                    INCB  (PC)+          ;;COUNT THE CHARACTER
9207 053632 000000          $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
9208 053634          $TYPEX: POP   R0
9209 053636 005737 002514          TST   CACHKN        ;IS THERE A CACHE?
9210 053642 001402                    BEQ   2$             ;BRANCH IF NOT
9211 053644                    POP   CONTRL      ;POP CACHE STATUS
9212 053650          2$:  POP   R1
9213 053652 000207          RETURN
9214 053654          SUPLIMIT:;.!!!!!!!.THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE

```

9892  
9893  
9894  
9895  
9896  
9897  
9898  
9899  
9900  
9901  
9902  
9903  
9904  
9905  
9906  
9907  
9908  
9909  
9910  
9911  
9912  
9913  
9914  
9915  
9916  
9917  
9918  
9919  
9920  
9921  
9922  
9923  
9924  
9925  
9926  
9927  
9928  
9929  
9930  
9931  
9932  
9933  
9934  
9935  
9936  
9937  
9938  
9939  
9940  
9941  
9942  
9943  
9944  
9945  
9946  
9947  
9948

```
.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
```

```
9949          THEN BAD DATA IS POINTED TO BY (R1)
9950          GOOD DATA IS ZERO
9951
9952          PERR14  TRAP
9953          BAD DATA IN R0 UNLESS ABORTED
9954          THEN BAD DATA IS POINTED TO BY (R1)
9955          GOOD DATA IS ONES
9956
9957          PERR15  TRAP
9958          BAD DATA IN R0 UNLESS ABORTED
9959          THEN BAD DATA IS POINTED TO BY (R1)
9960          GOOD DATA IN TSTDAT
9961
9962          PERR16  TRAP
9963          BAD DATA IN R0 UNLESS ABORTED
9964          THEN BAD DATA IS POINTED TO BY (R1)
9965          GOOD DATA IN TSTDAT+2
9966
9967          PERR17  TRAP
9968          BAD DATA IN R0 UNLESS ABORTED
9969          THEN BAD DATA IS POINTED TO BY (R1)
9970          GOOD DATA IN R2
9971
9972          PERR20  TRAP
9973          BAD DATA IN R0 UNLESS ABORTED
9974          THEN BAD DATA IS POINTED TO BY (R1)
9975          GOOD DATA IN R3
9976
9977          PERR21  TRAP
9978          7 BIT BYTE TEST
9979          BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
9980          THEN BAD DATA IS POINTED TO BY (R1)
9981          GOOD DATA IS A 7 BIT BYTE ON ONES
9982
9983          PERR22  TRAP
9984          BAD DATA IN R2 UNLESS ABORTED
9985          THEN BAD DATA IS POINTED TO BY (R1)
9986          GOOD DATA IN R0
9987
9988          PERR23  TRAP
9989          BAD DATA IN R0 UNLESS ABORTED
9990          THEN BAD DATA IS POINTED TO BY (R1)
9991          GOOD DATA IN R4
9992
9993          PERR24  TRAP
9994          BAD DATA IN R0 UNLESS ABORTED
9995          THEN BAD DATA IS POINTED TO BY (R2)
9996          GOOD DATA IN R3
9997
9998          PERR25  TRAP
9999          BAD DATA POINTED TO BY -(R1)
10000         GOOD DATA IN R2 UNLESS LOC V177654 IS SET
10001         THEN GOOD DATA IS IN R3
10002
10003         PERR26  TRAP
10004         BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
10005         GOOD DATA IS 000000,,100000,,100
```

10006  
10007  
10008  
10009  
10010  
10011  
10012  
10013  
10014  
10015  
10016  
10017  
10018  
10019  
10020  
10021  
10022  
10023  
10024  
10025  
10026  
10027  
10028  
10029  
10030  
10031  
10032  
10033

```
:  
: PERR27 TRAP  
: BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0  
: GOOD DATA IS 000000,,000000,,077  
:  
: PERR30 TRAP  
: BAD DATA IS POINTED TO BY -16(SP)  
: GOOD DATA IS IN R1  
:  
: PERR31 TRAP  
: SPECIAL ECC FAILURE HANDLER  
:  
: PERR32 TRAP  
: SPECIAL ECC FAILURE HANDLER  
:  
: PERR33 TRAP  
: SPECIAL ECC FAILURE HANDLER  
:  
: PERR34 TRAP  
: SPECIAL ECC FAILURE HANDLER  
:  
: PERR35 TRAP  
: SPECIAL BRANCH GOBBLE FAILURE HANDLER.  
:  
: CALLING SEQUENCE FOR TRAP TYPES  
: BEQ 2$ ;NO - ERROR,BRANCH FOR CARD  
: PERRXX ;TRAP TO ERROR ROUTINE  
:2$: NEXT INSTRUCTION ;CONTINUE TESTING
```



10036	053654	010437	002032		\$PER01: MOV R4,ADDRESS	
10037	053660	162737	000002	002032	SUB #2,ADDRESS	
10038	053666	010037	002050		MOV R0,BAD	
10039	053672	010537	002042		MOV R5,GOOD	
10040	053676	000137	054410		JMP PERRAW	
10041						
10042	053702	010437	002032		\$PER02: MOV R4,ADDRESS	
10043	053706	162737	000002	002032	SUB #2,ADDRESS	
10044	053714	010137	002050		MOV R1,BAD	
10045	053720	010237	002042		MOV R2,GOOD	
10046	053724	000137	054410		JMP PERRAW	
10047						
10048	053730	010137	002032		\$PER03: MOV R1,ADDRESS	
10049	053734	162737	000002	002032	SUB #2,ADDRESS	
10050	053742	010437	002042		MOV R4,GOOD	
10051	053746	016137	177776	002050	MOV -2(R1),BAD	
10052	053754	000137	054410		JMP PERRAW	
10053						
10054	053760	010037	002032		\$PER04: MOV R0,ADDRESS	
10055	053764	162737	000002	002032	SUB #2,ADDRESS	
10056	053772	010437	002050		MOV R4,BAD	
10057	053776	010237	002042		MOV R2,GOOD	
10058	054002	000137	054410		JMP PERRAW	
10059						
10060	054006	010237	002042		PERR05: MOV R2,GOOD	
10061	054012	014037	002050		PERA05: MOV -(R0),BAD	
10062	054016	010037	002032		MOV R0,ADDRESS	
10063	054022	062700	000002		ADD #2,R0	;RESTORE R0
10064	054026	004737	040126		CALL BADSTACK	
10065	054032	000207			RETURN	
10066						
10067	054034	005037	002042		PERR06: CLR GOOD	
10068	054040	000764			BR PERA05	
10069						
10070	054042	010137	002032		\$PER07: MOV R1,ADDRESS	
10071	054046	010237	002050		MOV R2,BAD	
10072	054052	013737	002234	002042	MOV DATBUF,GOOD	
10073	054060	000137	054410		JMP PERRAW	
10074						
10075	054064				\$PER10: LET ADDRESS := R1 + #2	
10076	054076				LET BAD := R2	
10077	054102				LET GOOD := DATBUF+2	
10078	054110	000137	054410		JMP PERRAW	
10079						
10080	054114				\$PER11: LET ADDRESS := R1	
10081	054120				LET BAD := R0	
10082	054124				LET GOOD := #0	
10083	054130	000137	054462		JMP PERRAB	
10084						
10085	054134				\$PER12: LET ADDRESS := R1	
10086	054140				LET BAD := R0	
10087	054144				LET GOOD := #377	
10088	054152	000137	054462		JMP PERRAB	

10091	054156		\$PER13: LET ADDRESS := R1
10092	054162		LET BAD := R0
10093	054166		LET GOOD := #0
10094	054172	000137 054410	JMP PERRAW
10095			
10096	054176		\$PER14: LET ADDRESS := R1
10097	054202		LET BAD := R0
10098	054206		LET GOOD := ONES
10099	054214	000137 054410	JMP PERRAW
10100			
10101	054220		\$PER15: LET ADDRESS := R1
10102	054224		LET BAD := R0
10103	054230		LET GOOD := TSTDAT
10104	054236	000137 054410	JMP PERRAW
10105			
10106	054242		\$PER16: LET ADDRESS := R1
10107	054246		LET BAD := R0
10108	054252		LET GOOD := TSTDAT+2
10109	054260	000453	BR PERRAW
10110			
10111	054262		\$PER17: LET ADDRESS := R1
10112	054266		LET BAD := R0
10113	054272		LET GOOD := R2
10114	054276	000444	BR PERRAW
10115			
10116	054300		\$PER20: LET ADDRESS := R1
10117	054304		LET BAD := R0
10118	054310		LET GOOD := R3
10119	054314	000435	BR PERRAW
10120			
10121	054316		\$PER21: LET ADDRESS := R1
10122	054322		LET BAD := R0
10123	054326		LET GOOD := #177
10124	054334	000477	BR PERRAW
10125			
10126	054336		\$PER22: LET ADDRESS := R1
10127	054342		LET BAD := R2
10128	054346		LET GOOD := R0
10129	054352	000416	BR PERRAW
10130			
10131	054354		\$PER23: LET ADDRESS := R1
10132	054360		LET BAD := R0
10133	054364		LET GOOD := R4
10134	054370	000407	BR PERRAW
10135			
10136	054372		\$PER24: LET ADDRESS := R2
10137	054376		LET BAD := R0
10138	054402		LET GOOD := R3
10139	054406	000400	BR PERRAW

10141 054410

PERRAW: SUBTST <<DATA WAS A WORD>>  
:\*\*\*\*\*  
:\*SUBTEST DATA WAS A WORD  
:\*\*\*\*\*

10142 054410 004737 054644

10143 054414

10144 054426

10145 054440 004737 054620

10146 054444

10147 054452 104011

10148 054454

10149 054456 104012

10150 054460

10151 054460 000002

10152

10153 054462

CALL PERBNK  
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA  
IF BADPC EQ #0 THEN \$CALL BADSTACK  
CALL PERXOR  
IF ABORTFLAG IS FALSE  
ERROR +11  
ELSE  
ERROR +12  
END ;OF IF ABORTFLAG  
RTI

PERRAB: SUBTST <<DATA WAS A BYTE>>  
:\*\*\*\*\*  
:\*SUBTEST DATA WAS A BYTE  
:\*\*\*\*\*

10154 054462 004737 054644

10155 054466

10156 054500

10157 054512 004737 054620

10158 054516

10159 054524 104014

10160 054526

10161 054530 104015

10162 054532

10163 054532 000002

CALL PERBNK  
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA  
IF BADPC EQ #0 THEN \$CALL BADSTACK  
CALL PERXOR  
IF ABORTFLAG IS FALSE  
ERROR +14  
ELSE  
ERROR +15  
END ;OF IF ABORTFLAG  
RTI

10166 054534

```
PERRA7: SUBTST <<DATA WAS A 7 BIT BYTE>>  
:*****  
:*SUBTEST DATA WAS A 7 BIT BYTE  
:*****
```

10167 054534  
10168 054546 004737 054620  
10169 054552 004737 054644  
10170 054556 104022  
10171 054560 000002

```
IF BADPC EQ #0 THEN $CALL BADSTACK  
CALL PERXOR  
CALL PERBNK  
ERROR +22  
RTI
```

10172  
10173 054562  
10174 054570  
10175 054576 000137 051212  
10176  
10177 054602 005037 002044  
10178 054606  
10179 054614 000137 051212  
10180  
10181 054620

```
$PER26: LET GOOD2 := #100000  
LET GOOD3 := #100  
JMP PERRA3  
  
$PER27: CLR GOOD2  
LET GOOD3 := #077  
JMP PERRA3
```

```
PERXOR: SUBTST <<DETERMINE XOR OF GOOD & BAD>>  
:*****  
:*SUBTEST DETERMINE XOR OF GOOD & BAD  
:*****
```

10182 054620  
10183 054622 013700 002042  
10184 054626 013737 002050 002056  
10185 054634 074037 002056  
10186 054640  
10187 054642 000207

```
PUSH RO  
MOV GOOD,RO  
MOV BAD,BAD XOR  
XOR RO,BAD XOR  
POP RO  
RETURN
```

10190 054644

```
PERBNK: SUBTST<<LOG ERROR ON BAD BANK>>
:.....
:SUBTEST LOG ERROR ON BAD BANK
:.....
```

```
10191
10192 054644
10193 054650 013701 002100
10194 054654 006301
10195 054656 006301
10196 054660 052761 000001 002624
10197 054666 105261 002626
10198 054672 001002
10199 054674 105361 002626
10200 054700 126137 002626 002524 12$:
10201 054706 101403
10202 054710
10203 054716
10204 054722 000207
10205
10206 054724 010037 002050
10207 054730
10208 054740 013737 002240 002042
10209 054746
10210 054750 013737 002242 002042
10211 054756
10212 054756 004737 054620
10213 054762
10214 054770 000207
10215
10216 054772
10217 055004 004737 054724
10218 055010
10219 055020 104037
10220 055022
10221 055024
10222 055032 104042
10223 055034
10224 055034
10225 055044 104043
10226 055046
10227 055046
10228 055056 104044
10229 055060
10230 055060
10231 055066 000002
```

```
:WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE
PUSH RO,R1
MOV BANK,R1
ASL R1
ASL R1
BIS #BIT0,CONFIG(R1)
INCB CONFIG+2(R1) ;BUMP BANK COUNTER
BNE 12$ ;NO OVERFLOW - SKIP
DECB CONFIG+2(R1) ;SET BACK TO 255.
CMPB CONFIG+2(R1),ERRMAX ;IS IT PAST MAX?
BLOS 11$ ;NO - SKIP
SET TOOMANY ;YES
POP R1,RO
RETURN

PERECC: MOV RO,BAD
IF ADDRESS EQ TESTADD
MOV TSTDAT,GOOD
ELSE
MOV TSTDAT+2,GOOD
END :OF IF (R1)
CALL PERXOR
SET HEADER
RETURN

$PER31: IF BADPC EQ #0 THEN $CALL BADSTACK
CALL PERECC
IF REALPAT EQ #11
ERROR +37
END :OF IF REALPAT
IF REALPAT EQ #14
ERROR +42
END :OF IF REALPAT
IF REALPAT EQ #15
ERROR +43
END :OF IF REALPAT
IF REALPAT EQ #16
ERROR +44
END :OF IF REALPAT
SET HEADER
RTI
```

10234	055070				\$PER32: IF BADPC EQ #0 THEN \$CALL BADSTACK
10235	055102	010137	002032		MOV R1,ADDRESS
10236	055106	010037	002050		MOV R0,BAD
10237	055112	010237	002042		MOV R2,GOOD
10238	055116				SET HEADER
10239	055124	104040			ERROR +40
10240	055126				SET HEADER
10241	055134	000002			RTI
10242					
10243	055136				\$PER33: IF BADPC EQ #0 THEN \$CALL BADSTACK
10244	055150	010137	002032		MOV R1,ADDRESS
10245	055154	010037	002050		MOV R0,BAD
10246	055160	105037	002051		CLRB BAD+1
10247	055164	012737	000377	002042	MOV #377,GOOD
10248	055172	004737	054620		CALL PERXOR
10249	055176				SET HEADER
10250	055204	104041			ERROR +41
10251	055206				SET HEADER
10252	055214	000002			RTI
10253					
10254	055216				\$PER34: IF BADPC EQ #0 THEN \$CALL BADSTACK
10255	055230				IF #BIT15:BIT4 OFF.IN CSR
10256	055240	104016			ERROR +16 ;NO SBE OR DBE
10257	055242				ELSE
10258	055244	104001			ERROR +1 ;EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
10259	055246				END ;OF IF #BIT15.BIT4
10260	055246	000002			RTI
10261					
10262					;DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG
10263	055250	004737	054644		\$PER35: CALL PERBNK
10264	055254	004737	040126		CALL BADSTACK
10265	055260	013737	002030	002050	MOV BADPSW,BAD
10266	055266	012737	000012	002042	MOV #12,GOOD
10267	055274	104047			ERROR +47
10268	055276	062706	000004		ADD #4,SP ;FIX STACK FROM TRAP
10269	055302	000207			RETURN ;ABORTING TEST
10270					
10271	055304	010037	002042		\$PER36: MOV R0,GOOD
10272	055310	010137	002050		MOV R1,BAD
10273	055314				SET HEADER
10274	055322	104023			ERROR +23
10275	055324				SET HEADER
10276	055332	000002			RTI

10279  
 10280  
 10281  
 10282  
 10283  
 10284  
 10285  
 10286  
 10287  
 10288 055334 005237 062476  
 10289 055340  
 10290 055342 005037 062476  
 10291 055346 105237 062500  
 10292 055352  
 10293 055352 104410  
 10294 055354 005737 006104  
 10295 055360 001402  
 10296 055362 004737 061002  
 10297 055366  
 10306 055366  
 10307 055404 005037 002370  
 10308 055410 000137 045074  
 10309 055414  
 10310 055414  
 10311 055422 000002  
 10312 055424  
 10313 055424  
 10314  
 10315 055434 000425  
 10316  
 10317 055436 013746 000004  
 10318 055442 012737 055462 000004  
 10319 055450 005737 177060  
 10320 055454 012637 000004  
 10321 055460 000430  
 10322 055462 062706 000004  
 10323 055466 022737 000001 003710  
 10324 055474 001002  
 10325 055476 005037 177766  
 10326 055502 012637 000004  
 10327 055506 000407  
 10328 055510  
 10329 055510 105737 002012  
 10330 055514 001412  
 10331 055516 032777 001000 125052  
 10332 055524 001404  
 10333 055526 013737 002564 002562  
 10334 055534 000410  
 10335 055536 105037 002012  
 10336 055542 011637 002562  
 10337 055546 011637 002564  
 10338 055552 005037 002332  
 10339 055556 004737 055570  
 10340 055562 013716 002562  
 10341 055566 000002

```

.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=ICT
$SCOPE: INC    $DEVCT      ;TELL APT WE ARE ALIVE
          IF RESULT IS LT
          CLR    $DEVCT
          INCB  $UNIT
          END ;OF IF RESULT
          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
          TST    TRACE
          BEQ    NOTRCE
          CALL   CONTT    ;TRACE
NOTRCE:
          IF STOPOK IS TRUE AND #SW8 SET.IN @SWR
          CLR    STOPOK
          JMP    EXIT
          END ;OF IF STOPOK
          IF NOSCOPE IS TRUE
          RTI
          END ;OF IF NOSCOPE
1$:      IF #SW14 SET.IN @SWR THEN GOTO $OVER
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR    2$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                          ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
                          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
                          ;;SET FOR TIMEOUT
                          ;;TIME OUT ON XOR?
                          ;;RESTORE THE ERROR VECTOR
                          ;;GO TO THE NEXT TEST
                          ;;FIX STACK FROM TRAP
                          ;;IS THIS AN 11/44?
                          ;;BRANCH IF NOT
                          ;;RESET CPU ERROR REGISTER
                          ;;RESTORE THE ERROR VECTOR
                          ;;LOOP ON THE PRESENT TEST
2$:;*****END OF CODE FOR THE XOR TESTER*****
3$:      1STB  $ERFLG      ;;HAS AN ERROR OCCURRED?
          BEQ  $$VLAD     ;;BR IF NO
          BIT  #SW9,@SWR  ;;LOOP ON ERROR?
          BEQ  5$         ;;BR IF NO
4$:      MOV  $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
          BR  $OVER
5$:      CLRB $ERFLG      ;;ZERO THE ERROR FLAG
          MOV (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
          MOV (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
          CLR  $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
$OVER:   CALL GETDIS
          MOV  $LPADR,(SP) ;;FUDGE RETURN ADDRESS
          RTI              ;;FIXES PS

```

10343 055570

GETDIS: SUBTST <<SUBR DISPLAY>>

\*\*\*\*\*  
: \*SUBTEST            SUBR    DISPLAY  
\*\*\*\*\*

10344 055570 113737 002100 002011  
10345 055576 113737 002260 002010  
10346 055604  
10347 055606 005737 002124  
10348 055612 001403  
10349 055614 052737 100000 002010  
10350 055622  
10354 055622 013777 002010 124750  
10355 055630 013737 002010 000174  
10356 055636  
10357 055640 000207

1\$:  
      MOVB    BANK,\$BANK  
      MOVB    REALPAT,\$PATMAR  
      PUSH    R0  
      TST     RLFLAG            ;ARE WE RELOCATED?  
      BEQ     1\$                ;NO - SKIP  
      BIS     #BIT15,\$PATMAR    ;YES - SET MSB  
  
      MOV     \$PATMAR,@DISPLAY  
      MOV     \$PATMAR,DISPREG    ;SOFTWARE DISPLAY REGISTER  
      POP     R0  
      RETURN



```

10360
10361
10362
10363
10364
10365
10366
10367
10368
10369
10370
10371
10372
10373
10374 055642
10375 055650 104410
10376 055652 105237 002012
10377 055656 001775
10378 055660 004737 055570
10379 055664 013737 002010 062472
10380 055672 032777 002000 124676
10381 055700 001404
10382 055702
10383 055706
10384 055712 005237 002570
10385 055716
10386 055720 012737 077777 002570
10387 055726
10388 055726
10389 055726 011637 002016
10390 055732 162737 000002 002016
10391 055740 010637 002022
10392 055744 016637 000002 002026
10393 055752 117737 124040 002013
10394 055760
10395 055766
10396 055774 013737 002020 002016
10397 056002 162737 000002 002016
10398 056010 013737 002024 002022
10399 056016 013737 002030 002026
10400 056024 005037 002020
10401 056030
10402 056030 013737 002016 062470
10403 056036
10404 056046 000412
10405 056050
10411 056050
10412 056066
10413 056070
10414 056070
10415 056070 004737 056306
    
```

```

.SBTTL ROUTINE ERROR HANDLER
*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO $ERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1 HALT ON ERROR
*SW13=1 INHIBIT ERROR TYPEGUTS
*SW10=1 BELL ON ERROR
*SW9=1 LOOP ON ERROR
*CALL
*
ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

.ENABL LSB
$ERROR: IF NOERROR IS FALSE
1$: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 1$ ;;DON'T LET THE FLAG GO TO ZERO
CALL GETDIS ;;SETUP DISPLAY STUFF
MOV $PATMAR,$TESTN ;FOR APT
BIT #SW10,@SWR ;;BELL ON ERROR?
BEQ 2$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
TYPE MSG014 ;;CONTROL Z
2$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
IF RESULT IS MI
MOV #77777,$ERTTL
END ;OF IF RESULT
END ;OF IF NOERROR
MOV (SP),ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,ERRPC
MOV SP,ERRSP
MOV 2(SP),ERRPSW
MOVB @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
IF NOERROR IS FALSE
IF BADPC NE #0
MOV BADPC,ERRPC
SUB #2,ERRPC
MOV BADSP,ERRSP
MOV BADPSW,ERRPSW
CLR BADPC
END ;IF
MOV ERRPC,$FATAL ;FOR APT
IF #SW13 SET.IN @SWR
BR 3$
END ;OF IF #SW13
IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE
GOTO 3$
END ;OF IF #SW5
END ;OF IF NOERROR
CALL $ERRTYP ;;GO TO USER ERROR ROUTINE
    
```

10417 056074  
 10418 056102 005777 124470  
 10419 056106 100002  
 10420 056110 000000  
 10421 056112 104410  
 10422 056114  
 10423 056132 013716 002564  
 10424 056136  
 10425 056136 005737 002332  
 10426 056142 001402  
 10427 056144 013716 002332  
 10428 056150  
 10429 056156 022737 000001 003710  
 10430 056164 001002  
 10431 056166 005037 177766  
 10432 056172  
 10433 056214 012737 000001 062466  
 10434 056222 000137 045074  
 10435 056226  
 10436 056226  
 10437 056244  
 10438 056250 013700 000042  
 10439 056254 005037 000042  
 10440 056260 000137 014410  
 10441 056264  
 10442 056264  
 10443 056264  
 10444 056266  
 10445 056274  
 10446 056274  
 10447 056304 000002  
 10448

```

3$: IF NOERROR IS FALSE
    TST @SWR ;;HALT ON ERROR
    BPL 7$ ;;SKIP IF CONTINUE
$HALT: HALT ;;HALT ON ERROR!
    CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
7$: IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR
    MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
    END ;OF IF NOSCOPE
    TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
    BEQ 9$ ;;BR IF NONE
    MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
9$: IF DETFLAG IS FALSE
    CMP #1,PROTYP ;IS THIS AN 11/44?
    BNE 11$
    CLR CPUERR
11$: IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL$ IS TRUE
    MOV #1,$MSGTY ;FOR APT
    JMP EXIT
    END ;OF IF ACTFLAG
    IF XXDPCHAIN IS TRUE AND $ERTTL HI #20
    TYPE MSG066 ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN
    MOV 42,R0
    CLR 42
    JMP $ZAP42
    END ;OF IF XXDPCHAIN
    END ;OF IF DETFLAG
ELSE
    SET HEADER
    END ;OF IF NOERROR
10$: CLEAR TOOMANY,NOERROR
    RTI ;;RETURN
    .DSABL LSB
  
```

10451  
 10452  
 10453  
 10454  
 10455  
 10456  
 10457  
 10458 056306 104415  
 10459 056310  
 10460 056314 005000  
 10461 056316 153700 002013  
 10462 056322 001004  
 10463  
 10464 056324  
 10465 056332 000503  
 10466 056334 005300  
 10467 056336 006300  
 10468 056340 006300  
 10469 056342 006300  
 10470 056344 062700 063110  
 10471 056350 012037 056406  
 10472 056354 001417  
 10473 056356 005737 002400  
 10474 056362 001003  
 10475 056364 005737 002552  
 10476 056370 100011  
 10477 056372 005737 002062  
 10478 056376 001402  
 10479 056400  
 10480 056404  
 10481 056406 000000  
 10482 056410  
 10483 056414 012037 056440  
 10484 056420 001412  
 10485 056422 005737 002400  
 10486 056426 001003  
 10487 056430 005737 002552  
 10488 056434 100004  
 10489 056436  
 10490 056440 000000  
 10491 056442  
 10492 056446 012001  
 10493 056450 001427  
 10494 056452 012002

.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.

```

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

10497	056454	112203		7\$:	MOVB	(R2)+,R3	
10498	056456	006303			ASL	R3	;MAKE IT A WORD ADDRESS
10499	056460	004773	056466		CALL	@8\$(R3)	
10500	056464	000412			BR	9\$	
10501	056466	056602		8\$:	TAG70\$		
10502	056470	056612			TAG71\$		
10503	056472	056622			TAG72\$		
10504	056474	056672			TAG73\$		
10505	056476	056732			TAG74\$		
10506	056500	056744			TAG75\$		
10507	056502	056756			TAG76\$		
10508	056504	057022			TAG77\$		
10509	056506	057030			TAG78\$		
10510	056510	057110			TAG79\$		
10515	056512	062701	000002	9\$:	ADD	#2,R1	:UPDATE DATA TABLE POINTER
10516	056516	005711			TST	(R1)	::IS THERE ANOTHER NUMBER?
10517	056520	001403			BEQ	10\$	::BR IF NO
10518	056522				TYPE	MSG018	:TYPE 2 SPACES
10519	056526	000752			BR	7\$	::LOOP
10520							
10521	056530	005737	002106	10\$:	TST	MUT	:IS THERE A MEMORY UNDER TEST
10522	056534	001402			BEQ	11\$	:NO - SKIP
10523	056536	005237	002552		INC	HEADER	:YES - BUMP HEADER FLAG
10524	056542	104416		11\$:	RESREG		
10525	056544				IF #SW7	SET.IN @SWR AND DETFLAG IS FALSE AND NCERROR IS FALSE	
10526	056570	004737	057132		CALL	DETAIL	
10527	056574				END ;OF	IF #SW7	
10528	056574				TYPE	MSG104	:CONTROL Z
10529	056600	000207			RETURN		

10532  
10533  
10534  
10535 056602  
10536 056610 000207  
10537  
10538  
10539  
10540  
10541 056612  
10542 056620 000207  
10543  
10544  
10545  
10546  
10547 056622  
10548 056626 013701 002100  
10549 056632 070127 000004  
10550 056636  
10551 056644  
10552 056650 004737 037214  
10553 056654 005037 002342  
10554 056660  
10555 056664  
10556 056670 000207  
10557  
10558  
10559  
10560  
10561 056672  
10562 056676 013701 002100  
10563 056702 070127 000004  
10564 056706  
10565 056714 004737 037534  
10566 056720 005037 002342  
10567 056724  
10568 056730 000207  
10569  
10570  
10571  
10572  
10573 056732  
10574 056742 000207  
10575  
10576  
10577  
10578  
10579 056744  
10580 056754 000207

```
*****  
:*** OCTAL ***  
*****  
TAG70$: TYPOCT @ (R1) ;:TYPE AN OCTAL NUMBER  
RETURN  
  
*****  
:*** DECIMAL ***  
*****  
TAG71$: TYPDEC @ (R1) ;:TYPE A DECIMAL NUMBER  
RETURN  
  
*****  
:*** INTERLEAVE ***  
*****  
TAG72$: PUSH R1,R5  
MOV BANK,R1  
MUL #4,R1  
SET NOTAB ;INDICATE NO TABLE TO BE PRINTED - NOW  
TYPE MSG014  
CALL TCFIG1  
CLR NOTAB  
POP R5,R1  
TYPE MSG014 ;1 SPACE  
RETURN  
  
*****  
:*** CSR ***  
*****  
TAG73$: PUSH R1,R5  
MOV BANK,R1  
MUL #4,R1  
SET NOTAB  
CALL TCFIG3  
CLR NOTAB  
POP R5,R1  
RETURN  
  
*****  
:*** PATTERN ***  
*****  
TAG74$: TYPOCS REALPAT,<TYPE (0-77)>,2,2  
RETURN  
  
*****  
:*** BANK ***  
*****  
TAG75$: TYPOCS BANK,<TYPE (0-167)>,3  
RETURN
```

10582  
10583  
10584  
10585 056756  
10586 056762 013701 002100  
10587 056766 070127 000004  
10588 056772  
10589 057000  
10590 057004 004737 037346  
10591 057010 005037 002342  
10592 057014  
10593 057020 000207  
10594  
10595  
10596  
10597  
10598 057022  
10599 057026 000207  
10600  
10601  
10602  
10603  
10604 057030 013737 002032 002036  
10605 057036 162737 060000 002036  
10606 057044 013737 002100 002040  
10607 057052 006237 002040  
10608 057056 103003  
10609 057060 052737 100000 002036  
10610 057066 012746 002036  
10611 057072 004737 062346  
10612 057076 062706 000002  
10613 057102  
10614 057106 000207  
10615  
10616  
10617  
10618  
10619 057110  
10620 057114  
10621 057124  
10622 057130 000207

```
*****  
*** MTYPE ***  
*****  
TAG76$: PUSH R1,R5  
MOV BANK,R1  
MUL #4,R1  
SET NOTAB  
TYPE MSG019  
CALL TCFIG2  
CLR NOTAB  
POP R5,R1  
RETURN  
  
*****  
*** UNKNOWN DATA ***  
*****  
TAG77$: TYPE MSG061  
RETURN  
  
*****  
*** PHYSICAL ADDRESS ***  
*****  
TAG78$: MOV ADDRESS,PHYADD  
SUB #FIRST,PHYADD  
MOV BANK,PHYADD+2  
ASR PHYADD+2  
BCC 1$  
BIS #BIT15,PHYADD  
1$: MOV #PHYADD,-(SP) ; POINTER TO DOUBLE WORD ON STACK  
CALL $DB20 ; CALL DOUBLE PRECISION CONVERSION ROUTINE  
ADD #2,SP ; FIX STACK  
TYPE $OCT8  
RETURN  
  
*****  
*** OCTAL BYTE ***  
*****  
TAG79$: TYPE MSG018 ;2 SPACES  
TYPOCS @(R1),<TYPE BYTE>,3,Z  
TYPE MSG014 ;SPACE  
RETURN
```

10666 057132

DETAIL: SUBTS: <<SUBR DETAILED ERROR REPORT>>

10667 057132 005237 002212  
10668 057136 022737 000003 002212  
10669 057144 101473  
10670 057146 022737 000002 002212  
10671 057154 001435  
10672 057156  
10673 057166  
10674 057174 005037 002106  
10675 057200 010037 002172  
10676 057204 012700 002174  
10677 057210 010120  
10678 057212 010220  
10679 057214 010320  
10680 057216 010420  
10681 057220 010520  
10682 057222 013720 002022  
10683 057226 013720 002026  
10684 057232 013700 002172  
10685 057236  
10686 057244 104013  
10687 057246 000423  
10688 057250  
10689 057260  
10690 057266 005037 002106  
10691 057272  
10692 057300 104031  
10693 057302 022737 000001 003710  
10694 057310 001002  
10695 057312 005037 177766  
10696 057316  
10697  
10698 057326 004737 057132  
10699 057332 000207

\*\*\*\*\*  
: \*SUBTEST SUBR DETAILED ERROR REPORT  
\*\*\*\*\*  
INC DETFLAG  
CMP #3,DETFLAG  
BLOS 4\$  
CMP #2,DETFLAG  
BEQ 2\$  
PUSH HEADER,MUT  
SET HEADER  
CLR MUT  
MOV R0,DETRO  
MOV #DETR1,R0  
MOV R1,(R0)+  
MOV R2,(R0)+  
MOV R3,(R0)+  
MOV R4,(R0)+  
MOV R5,(R0)+  
MOV ERRSP,(R0)+  
MOV ERRPSW,(R0)+  
MOV DETRO,R0  
SET NOERROR  
ERROR +13  
BR 1\$  
2\$: PUSH HEADER,MUT  
SET HEADER  
CLR MUT  
SET NOERROR  
ERROR +31  
CMP #1,PROTYP ;IS THIS AN 11/44?  
BNE 1\$  
CLR CPUERR  
1\$: POP MUT,HEADER  
;WARNING RECURSIVE  
CALL DETAIL  
RETURN

```

10702 ;SIMULATE CONTROL 'T'
10703 057334 004737 061002 4$: CALL CONTT ;DISPLAY 'DISPLAY' INFO
10704
10705 ;TYPE CONTENTS OF ALL CSR'S
10706 057340 PUSH CSR,CSRNO,R1
10707 057352 TYPE MSG058
10708 057356 TYPE $CRLF
10709 057362 013701 002216 MOV TOTCSRS,R1
10710 057366 BEGIN DUMPCSRLOOP
10711 057366 FOR CSRNO := #0 TO #36 BY #2
10712 057372 006301 ASL R1
10713 057374 ON.ERROR
10714 057376 104426 READCSR
10715 057400 TYPOCT CSR
10716 057406 TYPE MSG018 ;2 SPACES
10717 057412 END ;OF ON.ERROR
10718 057412 IF R1 EQ #0 THEN LEAVE DUMPCSRLOOP
10719 057416 END ;OF FOR CSRNO
10720 057434 END DUMPCSRLOOP
10721 057434 POP R1,CSRNO,CSR
10722
10723 ;TYPE STACKS
10724 057446 PUSH RO,R1
10725 057452 TYPE MSG088 ;KERNEL STACK
10726 057456 013701 002534 MOV KSTACK,R1
10727 057462 162701 000002 SUB #2,R1
10728 057466 FOR RO := SP TO R1 BY #2
10729 057470 TYPE $CRLF
10730 057474 TYPOCT RO
10731 057500 TYPE MSG018 ;2 SPACES
10732 057504 TYPOCT (RO)
10733 057510 END ;OF FOR RO
10734 ;SET PREVIOUS MODE TO SUPERVISOR
10735 057520 005737 002426 TST NOSUPER
10736 057524 001036 BNE DET1
10737 057526 042737 030000 177776 BIC #BIT13!BIT12,PSW
10738 057534 052737 010000 177776 BIS #BIT12,PSW
10739 057542 006506 MFPI SSP
10740 057544 POP R1,RO
10741 057550 TYPE MSG089 ;SUPERVISOR STACK
10742 057554 IF RO LT #SUPSTK
10743 057562 FOR RO := RO TO #SUPSTK-2 BY #2
10744 057562 TYPE $CRLF
10745 057566 TYPOCT RO
10746 057572 TYPE MSG018 ;2 SPACES
10747 057576 TYPOCT (RO)
10748 057602 END ;OF FOR RO
10749 057614 ELSE
10750 057616 TYPE MSG091 ;IS EMPTY
10751 057622 END ;OF IF RO
10752 ;SET PREVIOUS MODE TO USER
10753 057622 052737 030000 177776 DET1: BIS #BIT13!BIT12,PSW
10754 057630 006506 MFPI USP
10755 057632 POP RO
10756 057634 TYPE MSG090 ;USER STACK
10757 057640 IF RO LT #USESTK
10758 057646 FOR RO := RO TO #USESTK-2 BY #2

```



10759 057646  
10760 057652  
10761 057656  
10762 057662  
10763 057666  
10764 057700  
10765 057702  
10766 057706  
10767 057706  
10768 057712 005037 002212  
10769 057716  
10770 057720 000207

TYPE \$CRLF  
TYPOCT RO  
TYPE MSG018 ;2 SPACES  
TYPOCT (RO)  
END ;OF FOR RO  
ELSE  
TYPE MSG091 ;IS EMPTY  
END ;OF IF RO  
TYPE \$CRLF  
CLR DETFLAG  
POP RO  
RETURN

10808  
 10809  
 10810  
 10811  
 10812  
 10813  
 10814  
 10815  
 10816  
 10817  
 10818  
 10819  
 10820  
 10821  
 10822  
 10823  
 10824  
 10825  
 10826  
 10827  
 10828  
 10829  
 10830  
 10831  
 10831  
 10832  
 10833  
 10834  
 10835  
 10836  
 10837  
 10838  
 10839  
 10840  
 10841  
 10842  
 10843  
 10844  
 10845  
 10846  
 10847  
 10848  
 10849  
 10850  
 10851  
 10852  
 10853  
 10854  
 10855  
 10856  
 10857  
 10858  
 10859  
 10860  
 10861  
 10862  
 10863  
 10864

057722 017646 000000  
 057726 116637 000001 060145  
 057734 112637 060147  
 057740 062716 000002  
 057744 000406  
 057746 112737 000001 060145  
 057754 112737 000006 060147  
 057762 112737 000005 060144  
 057770 010346  
 057772 010446  
 057774 010546  
 057776 113704 060147  
 060002 005404  
 060004 062704 000006  
 060010 110437 060146  
 060014 113704 060145  
 060020 016605 000012  
 060024 005003  
 060026 006105  
 060030 000404  
 060032 006105  
 060034 006105  
 060036 006105  
 060040 010503  
 060042 006103  
 060044 105337 060146  
 060050 100016  
 060052 042703 177770  
 060056 001002  
 060060 005704  
 060062 001403  
 060064 005204

```

.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
*$TYPOS: MOV     @ (SP),-(SP)  ;;PICKUP THE MODE
        MOV     1(SP), $OFILL  ;;LOAD ZERO FILL SWITCH
        MOV     (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD     #2, (SP)      ;;ADJUST RETURN ADDRESS
        BR     $TYPON
*$TYPOC: MOV     #1, $OFILL    ;;SET THE ZERO FILL SWITCH
        MOV     #6, $OMODE+1  ;;SET FOR SIX(6) DIGITS
*$TYPON: MOV     #5, $OCNT     ;;SET THE ITERATION COUNT
        MOV     R3,-(SP)      ;;SAVE R3
        MOV     R4,-(SP)      ;;SAVE R4
        MOV     R5,-(SP)      ;;SAVE R5
        MOV     $OMODE+1, R4  ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG     R4
        ADD     #6, R4        ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV     R4, $OMODE    ;;SAVE IT FOR USE
        MOV     $OFILL, R4    ;;GET THE ZERO FILL SWITCH
        MOV     12(SP), R5    ;;PICKUP THE INPUT NUMBER
        CLR     R3           ;;CLEAR THE OUTPUT WORD
1$:     ROL     R5           ;;ROTATE MSB INTO 'C'
        BR     3$           ;;GO DO MSB
2$:     ROL     R5           ;;FORM THIS DIGIT
        ROL     R5
        ROL     R5
        MOV     R5, R3
3$:     ROL     R3           ;;GET LSB OF THIS DIGIT
        DECB   $OMODE        ;;TYPE THIS DIGIT?
        BPL    6$           ;;BR IF NO
        BIC   #177770, R3    ;;GET RID OF JUNK
        BNE   4$           ;;TEST FOR 0
        TST   R4           ;;SUPPRESS THIS 0?
        BEQ   5$           ;;BR IF YES
4$:     INC    R4           ;;DON'T SUPPRESS ANYMORE 0'S
  
```

```

10865 060066 052703 000060          BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
10866 060072 052703 000040      5$:    BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
10867 060076 110337 060142          MOVB    R3,8$      ;;SAVE FOR TYPING
10868 060102          TYPE    8$      ;;GO TYPE THIS DIGIT
10869 060106 105337 060144      6$:    DECB    $OCNT      ;;COUNT BY 1
10870 060112 003347          BGT     2$      ;;BR IF MORE TO DO
10871 060114 002402          BLT     7$      ;;BR IF DONE
10872 060116 005204          INC     R4      ;;INSURE LAST DIGIT ISN'T A BLANK
10873 060120 000744          BR     2$      ;;GO DO THE LAST DIGIT
10874 060122 012605      7$:    MOV     (SP)+,R5      ;;RESTORE R5
10875 060124 012604          MOV     (SP)+,R4      ;;RESTORE R4
10876 060126 012603          MOV     (SP)+,R3      ;;RESTORE R3
10877 060130 016666 000002 000004      MOV     2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
10878 060136 012616          MOV     (SP)+,(SP)
10879 060140 000002          RTI                    ;;RETURN
10880 060142          000      8$:    .BYTE 0      ;;STORAGE FOR ASCII DIGIT
10881 060143          000      .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
10882 060144          000      $OCNT: .BYTE 0      ;;OCTAL DIGIT COUNTER
10883 060145          000      $OFILL: .BYTE 0      ;;ZERO FILL SWITCH
10884 060146          000000      $OMODE: .WORD 0      ;;NUMBER OF DIGITS TO TYPE

```

```

10886 .SBTTL ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
10887 :*****
10888 :*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
10889 :*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
10890 :*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
10891 :*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
10892 :*REPLACED WITH SPACES.
10893 :*CALL:
10894 :*
10895 :*
10896 060150 $TYPDS: PUSH RO,R1,R2,R3,R5 ;;PUT THE BINARY NUMBER ON THE STACK
10897 060162 012746 020200 MOV #20200,-(SP) ;;GO TO THE ROUTINE
10898 060166 016605 000020 MOV 20(SP),R5 ;;SET BLANK SWITCH AND SIGN
10899 060172 100004 BPL 1$ ;;GET THE INPUT NUMBER
10900 060174 005405 NEG R5 ;;BR IF INPUT IS POS.
10901 060176 112766 000055 000001 MOVB #'-,1(SP) ;;MAKE THE BINARY NUMBER POS.
10902 060204 005000 1$: CLR RO ;;MAKE THE ASCII NUMBER NEG.
10903 060206 012703 060364 MOV #SDBLK,R3 ;;ZERO THE CONSTANTS INDEX
10904 060212 112723 000040 MOVB #' ,(R3)+ ;;SETUP THE OUTPUT POINTER
10905 060216 005002 2$: CLR R2 ;;SET THE FIRST CHARACTER TO A BLANK
10906 060220 016001 060354 MOV $DTBL(RO),R1 ;;CLEAR THE BCD NUMBER
10907 060224 160105 3$: SUB R1,R5 ;;GET THE CONSTANT
10908 060226 002402 BLT 4$ ;;FORM THIS BCD DIGIT
10909 060230 005202 INC R2 ;;BR IF DONE
10910 060232 000774 BR 3$ ;;INCREASE THE BCD DIGIT BY 1
10911 060234 060105 4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
10912 060236 005702 TST R2 ;;CHECK IF BCD DIGIT=0
10913 060240 001002 BNE 5$ ;;FALL THROUGH IF 0
10914 060242 105716 TSTB (SP) ;;STILL DOING LEADING 0'S?
10915 060244 100407 BMI 7$ ;;BR IF YES
10916 060246 106316 5$: ASLB (SP) ;;MSD?
10917 060250 103003 BCC 6$ ;;BR IF NO
10918 060252 116663 000001 177777 MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
10919 060260 052702 000060 6$: BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII
10920 060264 052702 000040 7$: BIS #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
10921 060270 110223 MOVB R2,(R3)+ ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
10922 060272 005720 TST (R0)+ ;;JUST INCREMENTING
10923 060274 020027 000010 CMP RO,#10 ;;CHECK THE TABLE INDEX
10924 060300 002746 BLT 2$ ;;GO DO THE NEXT DIGIT
10925 060302 003002 BGT 8$ ;;GO TO EXIT
10926 060304 010502 MOV R5,R2 ;;GET THE LSD
10927 060306 000764 BR 6$ ;;GO CHANGE TO ASCII
10928 060310 105726 8$: TSTB (SP)+ ;;WAS THE LSD THE FIRST NON-ZERO?
10929 060312 100003 BPL 9$ ;;BR IF NO
10930 060314 116663 177777 177776 9$: MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
10931 060322 105013 CLR (R3) ;;SET THE TERMINATOR
10932 060324 POP R5,R3,R2,R1,RO
10933 060336 TYPE $DBLK ;;NOW TYPE THE NUMBER
10934 060342 016666 000002 000004 MOV 2(SP),4(SP) ;;ADJUST THE STACK
10935 060350 012616 MOV (SP)+,(SP)
10936 060352 000002 RTI ;;RETURN TO USER
10937 060354 023420 $DTBL: 10000.
10938 060356 001750 1000.
10939 060360 000144 100.
10940 060362 000012 10.
10941 060364 000000 000000 000000 $DBLK: .WORD 0,0,0,0
060372 000000

```

10943  
10944  
10945  
10946  
10947  
10948  
10949  
10950 060374  
10956 060374 005737 053444  
10957 060400 001406  
10958 060402 013746 053444  
10959 060406 005037 053444  
10960 060412 000137 060434  
10961 060416 105777 122160  
10962 060422 100130  
10963 060424 117746 122154  
10964 060430 042716 177600  
10965 060434 022716 000006  
10966 060440 001002  
10967 060442 004737 045344  
10968 060446 022716 000024  
10969 060452 001002  
10970 060454 004737 061002  
10971 060460 022716 000003  
10972 060464 001454  
10973 060466 022716 000023  
10974 060472 001002  
10975 060474 004737 061056  
10976 060500 022716 000013  
10977 060504 001005  
10978 060506  
10979 060512 013706 002142  
10980 060516 000207  
10981 060520 022737 000176 002576 6\$:  
10982 060526 001067  
10983 060530 022716 000007  
10984 060534 001064  
10985 060536 005737 002060  
10986 060542 001061  
10987 060544  
10988 060550  
10989 060554  
10990 060562  
10991 060566 005046 3\$:  
10992 060570 005046  
10993 060572 105777 122004 4\$:  
10994 060576 100375  
10995 060600 117746 122000  
10996 060604 042716 177600  
10997 060610 021627 000003  
10998 060614 001006  
10999 060616 5\$:  
11000 060622 062706 000006  
11001 060626 000137 044766  
11002 060632 021627 000025 7\$:  
11003 060636 001005  
11004 060640

```
.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
$CKSWR:
TST      XOCHAR      ;;SOMETHING THERE?
BEQ      NOCH        ;; GO ON IF NOT
MOV      XOCHAR,-(SP) ;; USE IT
CLR      XOCHAR
JMP      CONTS1
NOCH:    TSTB        @STKS      ;;CHAR THERE?
BPL      12$         ;;IF NO, DON'T WAIT AROUND
MOVB     @STKB,-(SP) ;;SAVE THE CHAR
BIC      #^C177,(SP) ;;STRIP-OFF THE ASCII
CONTS1:  CMP         #6,(SP)    ;;IS IT CONTROL F?
BNE      1$         ;;NO SKIP
CALL     FIELDSERVICE
1$:      CMP         #24,(SP)   ;;IS IT CONTROL T?
BNE      16$        ;;NO - SKIP
CALL     CONTT       ;;YES - CALL CONTROL T ROUTINE
16$:     CMP         #3,(SP)    ;;IS IT CONTROL C?
BEQ      5$         ;;YES EXIT *****NOTE***** STACK IS SCREWED UP.
2$:      CMP         #23,(SP)   ;;IS IT CONTROL S?
BNE      17$        ;;NO - SKIP
CALL     CONTS       ;;YES - CALL CONTROL S ROUTINE
17$:     CMP         #13,(SP)   ;;IS IT CONTROL K?
BNE      6$         ;;NO - SKIP
TYPE     $CNTLK     ;;TYPE A ^K
MOV      CTLKVEC,SP ;;RESET KSP TO AFTER PATTERN EXEC ROUTINE
RETURN   ;;RETURN TO PATTERN EXEC ROUTINE
6$:      CMP         #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
BNE      CKEND      ;;BRANCH IF NO
CMP      #7,(SP)    ;;IS IT A CONTROL G?
BNE      CKEND      ;;NO, RETURN TO USER
TST      $AUTO      ;;ARE WE RUNNING IN AUTO-MODE?
BNE      CKEND      ;;BRANCH IF YES
TYPE     $CNTLG     ;;ECHO THE CONTROL-G (^G)
$GTSWR:  TYPE     $MSWR  ;;TYPE CURRENT CONTENTS
TYOCT   @SWR        ;;OF THE SWR
TYPE     $MNEW      ;;PROMPT FOR NEW SWR
3$:      CLR         -(SP)     ;;CLEAR COUNTER
CLR      -(SP)      ;;THE NEW SWR
4$:      TSTB        @STKS      ;;CHAR THERE?
BPL      4$         ;;IF NOT TRY AGAIN
MOVB     @STKB,-(SP) ;;PICK UP CHAR
BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
CMP      (SP),#3    ;;IS IT A CONTROL-C?
BNE      7$         ;;BRANCH IF NOT
5$:      TYPE     $CNTLC     ;;YES, ECHO CONTROL-C (^C)
ADD      #6,SP      ;;CLEAN UP STACK
JMP      BOOT       ;;CONTROL-C RESTART
7$:      CMP         (SP),#25  ;;IS IT A CONTROL-U?
BNE      9$         ;;BRANCH IF NOT
TYPE     $CNTLU     ;;YES, ECHO CONTROL-U (^U)
```

11005	060644	062706	000006		8\$:	ADD	#6,SP	::	IGNORE PREVIOUS INPUT
11006	060650	000746				BR	3\$	::	LET'S TRY IT AGAIN
11007	060652	021627	000015		9\$:	CMP	(SP),#15	::	IS IT A <CR>?
11008	060656	001016				BNE	13\$	::	BRANCH IF NO
11009	060660	005766	000004			TST	4(SP)	::	YES, IS IT THE FIRST CHAR?
11010	060664	001403				BEQ	10\$	::	BRANCH IF YES
11011	060666	016677	000002	121702		MOV	2(SP),@SWR	::	SAVE NEW SWR
11012	060674	062706	000006		10\$:	ADD	#6,SP	::	CLEAR UP STACK
11013	060700					TYPE	\$CRLF	::	ECHO <CR> AND <LF>
11014	060704	000002			12\$:	RTI		::	RETURN
11015	060706	062706	000002		CKEND:	ADD	#2,SP	::	FIX STACK
11016	060712	000002				RTI		::	RETURN
11017	060714	004737	053446		13\$:	CALL	\$TYPEC	::	ECHO CHAR
11018	060720	021627	000060			CMP	(SP),#60	::	CHAR < 0?
11019	060724	002420				BLT	15\$	::	BRANCH IF YES
11020	060726	021627	000067			CMP	(SP),#67	::	CHAR > 7?
11021	060732	003015				BGT	15\$	::	BRANCH IF YES
11022	060734	042726	000060			BIC	#60,(SP)+	::	STRIP-OFF ASCII
11023	060740	005766	000002			TST	2(SP)	::	IS THIS THE FIRST CHAR
11024	060744	001403				BEQ	14\$	::	BRANCH IF YES
11025	060746	006316				ASL	(SP)	::	NO, SHIFT PRESENT
11026	060750	006316				ASL	(SP)	::	CHAR OVER TO MAKE
11027	060752	006316				ASL	(SP)	::	ROOM FOR NEW ONE.
11028	060754	005266	000002		14\$:	INC	2(SP)	::	KEEP COUNT OF CHAR
11029	060760	056616	177776			BIS	-2(SP),(SP)	::	SET IN NEW CHAR
11030	060764	000702				BR	4\$	::	GET THE NEXT ONE
11031	060766				15\$:	TYPE	\$QUES	::	TYPE ?<CR><LF>
11032	060772	000724				BR	8\$	::	SIMULATE CONTROL-U
11033	060774	136	113	015	\$CNTLK:	.ASCIZ	/'^k/<15><12>	::	CONTROL K ASCII STRING
	060777	012	000						
11034						.EVEN			
11035						.DSABL	LSB		

11038 061002

```
CONTT: SUBTST <<CONTROL T>>  
:*****  
:*SUBTEST CONTROL T  
:*****
```

11039 061002

11040 061004

11050 061010

11051 061016

11052 061022

11053 061022

11054 061026

11055 061036

11056 061042

11060 061052

11061 061054 000207

11062

11063 061056

```
PUSH RO  
TYPE $CRLF  
IF RLFLAG IS TRUE  
TYPE MSG092 ;RELOCATED  
END ;OF IF RLFLAG  
TYPE MSG093 ;BANK=  
TYPOCS BANK,,3 ;TYPE 3 DIGITS  
TYPE MSG095 ;PAT=  
TYPOCS REAL.PAT,,2 ;TYPE 2 DIGITS  
POP RO  
RETURN
```

```
CONTS: SUBTST <<CONTROL S & CONTROL Q>>  
:*****  
:*SUBTEST CONTROL S & CONTROL Q  
:*****
```

11064 061056

11065 061060 105777 121516

11066 061064 100375

11067 061066 117716 121512

11068 061072 042716 177600

11069 061076

11070 061104 000137 060434

11071 061110

11072 061112 000762

11073 061114

```
CONTS2: POP RO ;GET RID OF RETURN ADDRESS FROM STACK  
TSTB @$TKS ;WAIT FOR CHARACTER  
BPL CONTS2  
MOVB @$TKB,(SP) ;REPLACE OVER OLD CHARACTER ON STACK  
BIC #^C177,(SP) ;STRIP ALL BUT ASCII  
IF (SP) EQ #21 ;IF IT IS A CONTROL Q  
JMP CONTS1  
ELSE  
BR CONTS2  
END ;OF IF (SP)
```

```

11075
11076
11077
11078
11079
11080
11081
11082
11083 061114 011646
11084 061116 016666 000004 000002
11085 061124 105777 121452
11086 061130 100375
11087 061132 117766 121446 000004
11088 061140 042766 177600 000004
11089 061146 026627 000004 000023
11090 061154 001013
11091 061156 105777 121420
11092 061162 100375
11093 061164 117746 121414
11094 061170 042716 177600
11095 061174 022627 000021
11096 061200 001366
11097 061202 000750
11098 061204 026627 000004 000140
11099 061212 002407
11100 061214 026627 000004 000175
11101 061222 003003
11102 061224 042766 000040 000004
11103 061232 000002
11104
11105
11106
11107
11108
11109
11110 061234 010346
11111 061236 005046
11112 061240 012703 061532
11113 061244 022703 061556
11114 061250 101477
11115 061252 104411
11116 061254 112613
11117 061256 122713 000003
11118 061262 001016
11119 061264
11120 061270 005726
11121 061272 012603
11122 061274 032777 000400 121274
11123 061302 001404
11124 061304 005037 002370
11125 061310 000137 045074
11126 061314 000137 044766
11127 061320 122713 000177
11128 061324 001022
11129 061326 005716
11130 061330 001007
11131 061332 112737 000134 061530

```

```

*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
* RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
* RETURN HERE   ;; CHARACTER IS ON THE STACK
*               ;; WITH PARITY BIT STRIPPED OFF
:
$RDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC
        MOV      4(SP),2(SP)    ;; SAVE THE PS
1$:     TSTB     @$TKS          ;; WAIT FOR
        BPL      1$            ;; A CHARACTER
        MOVB     @$TKB,4(SP)    ;; READ THE TTY
        BIC      #^C<177>,4(SP) ;; GET RID OF JUNK IF ANY
        CMP      4(SP),#2$     ;; IS IT A CONTROL-S?
        BNE      3$            ;; BRANCH IF NO
2$:     TSTB     @$TKS          ;; WAIT FOR A CHARACTER
        BPL      2$            ;; LOOP UNTIL ITS THERE
        MOVB     @$TKB,-(SP)    ;; GET CHARACTER
        BIC      #^C177,(SP)   ;; MAKE IT 7-BIT ASCII
        CMP      (SP)+,#21     ;; IS IT A CONTROL-Q?
        BNE      2$            ;; IF NOT DISCARD IT
        BR       1$            ;; YES, RESUME
3$:     CMP      4(SP),#140     ;; IS IT UPPER CASE?
        BLT      4$            ;; BRANCH IF YES
        CMP      4(SP),#175     ;; IS IT A SPECIAL CHAR?
        BGT      4$            ;; BRANCH IF YES
        BIC      #40,4(SP)     ;; MAKE IT UPPER CASE
4$:     RTI                    ;; GO BACK TO USER
*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
* RDLIN
* RETURN HERE   ;; INPUT A STRING FROM THE TTY
*               ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*               ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
$RDLIN: MOV      R3,-(SP)      ;; SAVE R3
        CLR      -(SP)         ;; CLEAR THE RUBOUT KEY
1$:     MOV      #$TTYIN,R3    ;; GET ADDRESS
2$:     CMP      #$TTYIN+20.,R3 ;; BUFFER FULL?
        BLOS     8$            ;; BR IF YES
        RDCHR    ;; GO READ ONE CHARACTER FROM THE TTY
        MOVB     (SP)+,(R3)    ;; GET CHARACTER
        CMPB     #3,(R3)      ;; IS IT A CONTROL-C?
        BNE      3$            ;; BRANCH IF NO
        TYPE     $CNTLC        ;; TYPE A CONTROL-C (^C)
        TST      (SP)+        ;; CLEAN RUBOUT KEY OFF OF THE STACK
        MOV      (SP)+,R3     ;; RESTORE R3
        BIT      #BIT8,@SWR    ;; IS THERE A HALT FLAG SET IN THE SWR?
        BEQ      11$          ;; BRANCH IF NOT TO BOOT ROUTINE
        CLR      STOPOK       ;; GET READY TO HALT PROGRAM
        JMP      EXIT         ;; GO HALT PROGRAM
11$:    JMP      BOOT         ;; GOTO CONTROL-C RESTART
3$:     CMPB     #177,(R3)    ;; IS IT A RUBOUT
        BNE      5$            ;; BR IF NO
        TST      (SP)         ;; IS THIS THE FIRST RUBOUT?
        BNE      4$            ;; BR IF NO
        MOVB     #'\\,10$     ;; TYPE A BACK SLASH

```



```

11132 061340          TYPE 10$
11133 061344 012716 177777      MOV #-1,(SP)      ;;SET THE RUBOUT KEY
11134 061350          4$: DEC R3                ;;BACKUP BY ONE
11135 061352 005303          CMP R3,#$TTYIN    ;;STACK EMPTY?
11136 061356 103434          BLO 8$                ;;BR IF YES
11137 061360 111337 061530      MOVB (R3),10$     ;;SETUP TO TYPEOUT THE DELETED CHAR.
11138 061364          TYPE 10$      ;;GO TYPE
11139 061370 000725          BR 2$                ;;GO READ ANOTHER CHAR.
11140 061372 005716          5$: TST (SP)            ;;RUBOUT KEY SET?
11141 061374 001406          BEQ 6$                ;;BR IF NO
11142 061376 112737 000134 061530  MOVB #' \,10$    ;;TYPE A BACK SLASH
11143 061404          TYPE 10$
11144 061410 005016          CLR (SP)            ;;CLEAR THE RUBOUT KEY
11145 061412 122713 000025      6$: CMPB #25,(R3)   ;;IS CHARACTER A CTRL U?
11146 061416 001003          BNE 7$                ;;BR IF NO
11147 061420          TYPE $CNTLU      ;;TYPE A CONTROL 'U'
11148 061424 000705          BR 1$                ;;GO START OVER
11149 061426 122713 000022      7$: CMPB #22,(R3)   ;;IS CHARACTER A '^R'?
11150 061432 001011          BNE 9$                ;;BRANCH IF NO
11151 061434 105013          CLRB (R3)         ;;CLEAR THE CHARACTER
11152 061436          TYPE $CRLF      ;;TYPE A 'CR' & 'LF'
11153 061442          TYPE $TTYIN     ;;TYPE THE INPUT STRING
11154 061446 000676          BR 2$                ;;GO PICKUP ANOTHER CHACTER
11155 061450          8$: TYPE $QUES      ;;TYPE A '?'
11156 061454 000671          BR 1$                ;;CLEAR THE BUFFER AND LOOP
11157 061456 111337 061530      9$: MOVB (R3),10$    ;;ECHO THE CHARACTER
11158 061462          TYPE 10$
11159 061466 122723 000015      CMPB #15,(R3)+   ;;CHECK FOR RETURN
11160 061472 001264          BNE 2$                ;;LOOP IF NOT RETURN
11161 061474 105063 177777      CLRB -1(R3)     ;;CLEAR RETURN (THE 15)
11162 061500          TYPE $LF        ;;TYPE A LINE FEED
11163 061504 005726          TST (SP)+         ;;CLEAN RUBOUT KEY FROM THE STACK
11164 061506 012603          MOV (SP)+,R3     ;;RESTORE R3
11165 061510 011646          MOV (SP),-(SP)   ;;ADJUST THE STACK AND PUT ADDRESS OF THE
11166 061512 016666 000004 000002  MOV 4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
11167 061520 012766 061532 000004  MOV #$TTYIN,4(SP)
11168 061526 000002          RTI                ;;RETURN
11169 061530          10$: .BYTE 0      ;;STORAGE FOR ASCII CHAR. TO TYPE
11170 061531          .BYTE 0      ;;TERMINATOR
11171 061532 000024          $TTYIN: .REPT 20.      ;;RESERVE SIZE BYTES FOR TTY INPUT
11174 061556          136 103 015 $CNTLC: .ASCIZ /^C/<15><12>  ;;CONTROL 'C'
11175 061561          012 000
11175 061563          136 125 015 $CNTLU: .ASCIZ /^U/<15><12>  ;;CONTROL 'U'
11176 061566          012 000
11176 061570          136 107 015 $CNTLG: .ASCIZ /^G/<15><12>  ;;CONTROL 'G'
11177 061573          012 000
11177 061575          015 012 123 $MSWR: .ASCIZ <15><12>/SWR = /
11178 061600          127 122 040
11178 061603          075 040 000
11178 061606          040 040 116 $MNEW: .ASCIZ / NEW = /
11178 061611          105 127 040
11178 061614          075 040 000
11179          .EVEN

```

```

11181
11182
11183
11184
11185
11186
11187
11188
11189
11190
11191
11192
11193 061620 011646
11194 061622 016666 000004 000002
11195 061630
11196 061636 104412
11197 061640 012600
11198 061642 010037 061746
11199 061646 005001
11200 061650 005002
11201 061652 112046
11202 061654 001420
11203 061656 122716 000060
11204 061662 003026
11205 061664 122716 000067
11206 061670 002423
11207 061672 006301
11208 061674 006102
11209 061676 006301
11210 061700 006102
11211 061702 006301
11212 061704 006102
11213 061706 042716 177770
11214 061712 062601
11215 061714 000756
11216 061716 005726
11217 061720 010166 000012
11218 061724 010237 061766
11219 061730
11220 061736 000002
11221 061740 005726
11222 061742 105010
11223 061744
11224 061746 000000
11225 061750
11226 061754
11227 061760
11228 061764 000724
11229 061766 000000
11230
11231
11232
11233
11234
11235
11236
11237

```

```

.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
$RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
MOV      4(SP),2(SP)          ;;INPUT NUMBER
PUSH     R0,R1,R2
1$:  RDLIN          ;;READ AN ASCII LINE
MOV      (SP)+,R0            ;;GET ADDRESS OF 1ST CHARACTER
MOV      R0,5$              ;;AND SAVE IT
CLR      R1                 ;;CLEAR DATA WORD
CLR      R2
2$:  MOVB      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
BEQ      3$                 ;;IF ZERO GET OUT
CMPB     #'0,(SP)           ;;MAKE SURE THIS CHARACTER
BGT      4$                 ;;IS AN OCTAL DIGIT
CMPB     #'7,(SP)
BLT      4$
ASL      R1                 ;;*2
ROL      R2
ASL      R1                 ;;*4
ROL      R2
ASL      R1                 ;;*8
ROL      R2
BIC      #'C7,(SP)          ;;STRIP THE ASCII JUNK
ADD      (SP)+,R1           ;;ADD IN THIS DIGIT
BR       2$                 ;;LOOP
3$:  TST      (SP)+           ;;CLEAN TERMINATOR FROM STACK
MOV      R1,12(SP)          ;;SAVE THE RESULT
MOV      R2,$HIOCT
POP      R2,R1,R0
RTI
4$:  TST      (SP)+           ;;CLEAN PARTIAL FROM STACK
CLRB     (R0)               ;;SET A TERMINATOR
TYPE     ;;TYPE UP THRU THE BAD CHAR.
5$:  .WORD     0
TYPE     MSG062             ;;INPUT MUST BE A
TYPE     MSG063             ;;N OCTAL
TYPE     MSG064             ;;NUMBER
BR       1$                 ;;TRY AGAIN
$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

```

```

11238 ;*POSITIVE 32767 TO NEGATIVE 32768.
11239 ;*CALL:
11240 ;* RDDEC ;:READ A DECIMAL NUMBER
11241 ;* RETURN HERE ;:NUMBER IS ON TOP OF THE STACK
11242 ;
11243 ;
11244 061770 011646 $RDDEC: MOV (SP),-(SP) ;:PROVIDE SPACE FOR
11245 061772 016666 000004 000002 MOV 4(SP),2(SP) ;:THE INPUT NUMBER
11246 062000 PUSH RO,R1,R2
11247 062006 104412 1$: RDLIN ;:READ AN ASCIZ LINE
11248 062010 012600 MOV (SP)+,RO ;:ADDRESS OF 1ST CHAR.
11249 062012 010037 062136 MOV RO,6$ ;:SAVE INCASE OF BAD INPUT
11250 062016 005046 CLR -(SP) ;:CLEAR DATA WORD
11251 062020 005002 CLR R2 ;:SIGN SET POSITIVE
11252 062022 122710 000055 CMPB #'-(,RO) ;:SEE IF A MINUS SIGN WAS TYPED
11253 062026 001001 BNE 2$ ;:BR IF NO MINUS SIGN
11254 062030 112002 MOVB (RO)+,R2 ;:SAVE FOR LATER USE
11255 062032 112001 2$: MOVB (RO)+,R1 ;:PICKUP THIS CHARACTER
11256 062034 001424 BEQ 3$ ;:GET OUT IF ZERO
11257 062036 122701 000060 CMPB #'0,R1 ;:MAKE SURE THIS CHARACTER
11258 062042 003032 BGT 5$ ;:IS A DIGIT BETWEEN 0 & 9
11259 062044 122701 000071 CMPB #'9,R1
11260 062050 002427 BLT 5$
11261 062052 032716 170000 BIT #'C7777,(SP) ;:DON'T LET NUMBER GET TO BIG
11262 062056 001024 BNE 5$ ;:BR IF NUMBER WOULD OVERFLOW
11263 062060 006316 ASL (SP) ;:*2
11264 062062 011646 MOV (SP),-(SP) ;:SAVE FOR LATER
11265 062064 006316 ASL (SP) ;:*4
11266 062066 006316 ASL (SP) ;:*8.
11267 062070 062616 ADD (SP)+,(SP) ;:*10.
11268 062072 102416 BVS 5$ ;:OVERFLOW ISN'T ALLOWED
11269 062074 162701 000060 SUB #'0,R1 ;:STRIP AWAY THE ASCII JUNK
11270 062100 060116 ADD R1,(SP) ;:ADD IN THIS DIGIT
11271 062102 102412 BVS 5$ ;:OVERFLOW ISN'T ALLOWED
11272 062104 000752 BR 2$ ;:LOOP
11273 062106 005702 3$: TST R2 ;:CHECK IF NUMBER IS NEG
11274 062110 001401 BEQ 4$ ;:BR IF NO
11275 062112 005416 NEG (SP) ;:YES--NEGATE THE NUMBER
11276 062114 012666 000012 4$: MOV (SP)+,12(SP) ;:SAVE THE RESULT
11277 062120 POP R2,R1,RO
11278 062126 000002 RTI ;:RETURN
11279 ;
11280 062130 005726 5$: TST (SP)+ ;:CLEAN PARTIAL NUMBER FROM STACK
11281 062132 105010 CLRB (RO) ;:SET A TERMINATOR
11282 062134 TYPE ;:TYPE THE INPUT UP TO BAD CHAR.
11283 062136 000000 6$: .WORD 0 ;:POINTER GOES HERE
11284 062140 TYPE MSG062 ;:INPUT MUST BE A
11285 062144 TYPE MSG065 ;:DECIMAL
11286 062150 TYPE MSG064 ;:NUMBER
11287 062154 000714 BR 1$ ;:TRY AGAIN

```

11289  
11290  
11291  
11292  
11293  
11294  
11295  
11296  
11297  
11298  
11299  
11300  
11301  
11302  
11303  
11304  
11305  
11306 062156  
11307 062156  
11308 062172 016646 000022  
11309 062176 016646 000022  
11310 062202 016646 000022  
11311 062206 016646 000022  
11312 062212 000002  
11313  
11314  
11315  
11316  
11317 062214  
11318 062214 012666 000022  
11319 062220 012666 000022  
11320 062224 012666 000022  
11321 062230 012666 000022  
11322 062234  
11323 062250 000002

.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  
  
$SAVREG:  
PUSH R0,R1,R2,R3,R4,R5  
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW  
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW  
MOV 22(SP),-(SP) ;;SAVE PS OF CALL  
MOV 22(SP),-(SP) ;;SAVE PC OF CALL  
RTI  
  
;*RESTORE R0-R5  
;*CALL:  
;* RESREG  
$RESREG:  
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL  
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL  
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW  
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW  
POP R5,R4,R3,R2,R1,R0  
RTI
```

```

11325
11326
11327
11328
11329
11330
11331
11332
11333
11334
11335
11336 062252
11337 062260 013700 002544
11338 062264 013701 002542
11339 062270 012702 000007
11340 062274 006300
11341 062276 006101
11342 062300 077203
11343 062302 063700 002544
11344 062306 005501
11345 062310 063701 002542
11346 062314 062700 001057
11347 062320 005501
11348 062322 062701 047401
11349 062326 010037 002544
11350 062332 010137 002542
11351 062336
11352 062344 000207
  
```

.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
$RAND:  PUSH   R0,R1,R2
        MOV   SEEDLO,R0       ;;SET R0 WITH LOW
        MOV   SEEDHI,R1       ;;SET R1 WITH HIGH
        MOV   #7,R2           ;;SET SHIFT COUNT
1$:     ASL   R0               ;;SHIFT R0 LEFT AND
        ROL   R1               ;;ROTATE CARRY INTO R1 AND
        SOB   R2,1$
        ADD   SEEDLO,R0       ;;ADD NUMBER TO MAKE X 129
        ADC   R1               ;;PROPOGATE CARRY
        ADD   SEEDHI,R1       ;;ADD NUMBER TO MAKE X 129
        ADD   #1057,R0        ;;ADD LOW CONSTANT
        ADC   R1               ;;PROPOGATE CARRY
        ADD   #47401,R1       ;;ADD HIGH CONSTANT
        MOV   R0,SEEDLO       ;;SAVE R0
        MOV   R1,SEEDHI       ;;SAVE R1
        POP   R2,R1,R0
        RETURN
  
```

```

11355          .SBTTL  ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
11356          :*****
11357          :*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
11358          :*UNSIGNED OCTAL ASCII NUMBER.
11359          :*CALL
11360          :*      MOV      #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
11361          :*      CALL     $DB20          ;; CALL THE ROUTINE
11362          :*      RETURN    ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
11363
11364          $DB20: SAVREG          ;; SAVE ALL REGISTERS
11365 062346 104415      MOV      2(SP),R1    ;; PICKUP THE POINTER TO LOW WORD
11366 062350 016601 000002      MOV      #$OCTVL+13.,R5 ;; POINTER TO DATA TABLE
11367 062354 012705 062465      MOV      #12.,R4      ;; DO ELEVEN CHARACTERS
11368 062360 012704 000014      MOV      #^C7,R3     ;; MASK
11369 062364 012703 177770      MOV      (R1)+,R0    ;; LOWER WORD
11370 062370 012100      MOV      (R1)+,R1    ;; HIGH WORD
11371 062372 012101      CLR      R2          ;; TERMINATOR
11372 062374 005002      1$: MOVB   R2,-(R5)    ;; PUT CHARACTER IN DATA TABLE
11373 062376 110245      MOV      R0,R2      ;; GET THIS DIGIT
11374 062400 010002      DEC     R4          ;; COUNT THIS CHARACTER
11375 062402 005304      BGT     3$         ;; BR IF NOT THE LAST DIGIT
11376 062404 003007      BEQ    2$         ;; BR IF IT IS THE LAST DIGIT
11377 062406 001405      INC     R5         ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
11378 062410 005205      MOV     R5,2(SP)  ;; ASCII CHAR. & PUT IT ON THE STACK
11379 062412 010566 000002      RESREG          ;; RESTORE ALL REGISTERS
11380 062416 104416      RETURN         ;; RETURN TO USER
11381 062420 000207      2$: ASR     R3          ;; POSITION THE MASK FOR THE LAST DIGIT
11382 062422 006203      3$: ROR    R1          ;; POSITION THE BINARY NUMBER FOR
11383 062424 006001      ROR    R0          ;; THE NEXT OCTAL DIGIT
11384 062426 006000      ROR    R1
11385 062430 006001      ROR    R0
11386 062432 006000      ROR    R1
11387 062434 006001      ROR    R0
11388 062436 006000      ROR    R1
11389 062440 040302      BIC    R3,R2      ;; MASK OUT ALL JUNK
11390 062442 062702 000060      ADD    #'0,R2    ;; MAKE THIS CHAR. ASCII
11391 062446 000753      BR     1$        ;; GO PUT IT IN THE DATA TABLE
11392 062450 000016      $OCTVL: .REPT   14. ;; RESERVE DATA TABLE
11395          $OCT8=$OCTVL+4      ;; POINTER TO 11 DIGIT NUMBER

```

```

11397          .SBTTL  TABLES
11398
11399          .SBTTL  APT MAILBOX-ETABLE
11400 062466 $MAIL:
11401 062466 000000 $MSGTY: .WORD 0      ;;MESSAGE TYPE CODE
11402 062470 000000 $FATAL: .WORD 0      ;;FATAL ERROR NUMBER (ERROR PC)
11403 062472 000000 $TESTN: .WORD 0     ;;TEST PATTERN NUMBER
11404 062474 000000 $PASS:  .WORD 0     ;;PASS COUNT
11405 062476 000000 $DEVCT: .WORD 0     ;;DEVICE COUNT
11406 062500 000000 $UNIT:  .WORD 0     ;;I/O UNIT NUMBER
11407 062502 000000 $MSGAD: .WORD 0     ;;MESSAGE ADDRESS
11408 062504 000000 $MSGLG: .WORD 0     ;;MESSAGE LENGTH
11409 062506 $ETABLE: ;;APT ENVIRONMENT TABLE
11410 062506 000 $ENV:  .BYTE 0     ;;ENVIRONMENT BYTE ;SET TO A 1 FOR APT AUTO MODE
11411          ;NOTE: IF BIT #7 IS SET IN $ENVM THE TABLE BELOW (BEGINNING AT $MAMS1 AND
11412          ;      ENDING AT $MADR4) MUST BE FILLED IN TO INDICATE THE PROPER AMOUNT OF
11413          ;      EACH TYPE OF MEMORY.
11414 062507 000 $ENVM:  .BYTE 0     ;;ENVIRONMENT MODE
11415          ;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
11416 062510 000101 $SWREG: .WORD 101   ;;APT SWITCH REGISTER
11417 062512 000000 $USWR:  .WORD 0     ;;USED TO LIMIT THE NUMBER OF PASSES
11418 062514 000000 $CPUOP: .WORD 0     ;;CPU TYPE,OPTIONS
11419          ;*
11420          ;*      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
11421          ;*      11/70=06,PDQ=07,Q=10
11422          ;*
11423          ;* BIT 10=REAL TIME CLOCK
11424          ;* BIT 9=FLOATING POINT PROCESSOR
11425 062516 001 $MAMS1: .BYTE 1     ;;HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
11426 062517 004 $MTYP1: .BYTE 4     ;;MEM. TYPE,BLK#1
11427          ;*
11428          ;* MEM. TYPE BYTE -- (HIGH BYTE)
11429          ;*      900 NSEC CORE=001
11430          ;*      300 NSEC BIPOLAR=002
11431          ;*      PARITY MOS=003
11432 062520 177776 $MADR1: .WORD 177776 ;;HIGH ADDRESS,BLK#1
11433          ;*
11434 062522 000 $MAMS2: .BYTE 0     ;;HIGH ADDRESS,M.S. BYTE
11435 062523 000 $MTYP2: .BYTE 0     ;;MEM. TYPE,BLK#2
11436 062524 000000 $MADR2: .WORD 0     ;;MEM.LAST ADDRESS,BLK#2
11437 062526 000 $MAMS3: .BYTE 0     ;;HIGH ADDRESS,M.S.BYTE
11438 062527 000 $MTYP3: .BYTE 0     ;;MEM. TYPE,BLK#3
11439 062530 000000 $MADR3: .WORD 0     ;;MEM.LAST ADDRESS,BLK#3
11440 062532 000 $MAMS4: .BYTE 0     ;;HIGH ADDRESS,M.S.BYTE
11441 062533 000 $MTYP4: .BYTE 0     ;;MEM. TYPE,BLK#4
11442 062534 000000 $MADR4: .WORD 0     ;;MEM.LAST ADDRESS,BLK#4
11443 062536 000000 $VECT1: .WORD 0     ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
11444 062540 000000 $VECT2: .WORD 0     ;;INTERRUPT VECTOR#2BUS PRIORITY#2
11445 062542 000000 $BASE:  .WORD 0     ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
11446 062544 000000 $DEVN:  .WORD 0     ;;DEVICE MAP
11447
11448 062546 000000 $CDW1:  .WORD 0
11449 062550 000000 $CDW2:  .WORD 0

```

```

11451 ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
11452 ;ARE TO BE RUN FOR PARTICULAR MEMORIES
11453
11454 ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
11455 ;BITO SET WILL RUN THE FIRST ENTRY IN THE TABLE, BITO SET
11456 ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
11457
11458 ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
11459 ;
11460 062552 177777 ;FIELD SERVICE VALUE
11461 062554 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
11462 062556 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
11463 062560 177777 ;ECC PATTERNS 103777 TABLE = MKPAT:
11464 062562 177777 ;ECC PATTERNS 177777 TABLE = MKPAT:
11465 062564 177777 ;PARITY PATTERNS 003777 TABLE = MJPAT:
11469 062566 ;PARITY PATTERNS 177774 TABLE = MJPAT:
11470 ;SETEND:
11471 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
11472 ;INTERFACE SPEC.
11473 062566 $APTHD:
11474 062566 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
11475 062570 062466 $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
11476 062572 000043 $TSTM: .WORD 35. ;;RUN TIM OF LONGEST TEST
11477 062574 001274 $PASTM: .WORD 700. ;;RUN TIME IN SECS. OF 1ST PASS ON 128k (QUICK VERIFY)
11478 062576 000000 $UNITM: .WORD 0. ;;EXTRA RUN TIME OF A PASS FOR EACH ADDITIONAL 128k (QV)
11479 062600 000040 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```



.SBTTL ROUTINE TRAP DECODER

11481  
11482  
11483  
11484  
11485  
11486  
11487  
11488  
11489  
11490  
11491  
11492  
11493  
11494  
11495  
11496  
11497  
11498  
11499  
11500  
11501  
11502  
11503  
11504  
11505

062602 010046  
062604 016600 000002  
062610 005740  
062612 111000  
062614 006300  
062616 016000 062644  
062622 000200  
  
  
  
  
062624 011646  
062626 016666 000004 000002  
062634 000002  
  
062636  
062642 000000

```
*****
*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.

$TRAP:  MOV    RO,-(SP)      ;;SAVE RO
        MOV    2(SP),RO    ;;GET TRAP ADDRESS
        TST   -(RO)       ;;BACKUP BY 2
        MOVB  (RO),RO     ;;GET RIGHT BYTE OF TRAP
        ASL   RO          ;;POSITION FOR INDEXING
        MOV   $TRPAD(RO),RO ;;INDEX TO TABLE
        RTS   RO          ;;GO TO ROUTINE

;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO

$TRAP2: MOV   (SP),-(SP)   ;;MOVE THE PC DOWN
        MOV   4(SP),2(SP) ;;MOVE THE PSW DOWN
        RTI                    ;;RESTORE THE PSW

$NOTRAP:TYPE  MSG006      ;UNDEFINED TRAP INSTRUCTION
$HALT2: HALT
```

11508  
11509  
11510  
11511  
11512  
11513  
11514  
11515 062644 062624  
11516 062646 053320  
11517 062650 057746  
11518 062652 057722  
11519 062654 062636  
11520 062656 060150  
11521 062660 062636  
11522  
11523 062662 060550  
11524 062664 060374  
11525  
11526 062666 061114  
11527 062670 061234  
11528 062672 061620  
11529 062674 061770  
11530  
11531 062676 062156  
11532 062700 062214  
11533  
11534 062702 040156  
11535 062704 040166  
11536 062706 040176  
11537  
11538 062710 042374  
11539  
11540 062712 040206  
11541 062714 040232  
11542  
11543 062716 040250  
11544 062720 040344  
11545  
11546 062722 053654  
11547 062724 053702  
11548 062726 053730  
11549 062730 053760  
11550 062732 054042  
11551 062734 054064  
11552 062736 054114  
11553 062740 054134  
11554 062742 054156  
11555 062744 054176  
11556 062746 054220  
11557 062750 054242  
11558 062752 054262  
11559 062754 054300  
11560 062756 054316  
11561 062760 054336  
11562 062762 054354  
11563 062764 054372  
11564 062766 051130

.SBTTL TRAP TABLE

;\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
;\*BY THE "TRAP" INSTRUCTION.

```

:          ROUTINE
:          -----
$TRPAD:  .WORD  $TRAP2
          $TYPE  ;CALL=TYPEIT   TRAP+1(104401)  TTY TYPEOUT ROUTINE
          $TYPOC ;CALL=TYPOC   TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
          $TYPOS ;CALL=TYPOS   TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
          $NOTRAP;$STYPON ;CALL=TYPON   TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
          $TYPDS ;CALL=TYPDS   TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
          $NOTRAP;$STYPBN ;CALL=TYPBN   TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

          $GTSWR ;CALL=GTSWR   TRAP+7(104407)  GET SOFT-SWR SETTING
          $CKSWR ;CALL=CKSWR   TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR

          $RDCHR ;CALL=RDCHR   TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
          $RDLIN ;CALL=RDLIN   TRAP+12(104412) TTY TYPEIN STRING ROUTINE
          $RDOCT ;CALL=RDOCT   TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
          $RDDEC ;CALL=RDDEC   TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY

          $$SAVREG ;CALL=SAVREG TRAP+15(104415) SAVE R0-R5 ROUTINE
          $RESREG ;CALL=RESREG TRAP+16(104406) RESTORE R0-R5 ROUTINE

          $KERNEL ;CALL=KERNEL TRAP+17(104417) ENTER KERNEL MODE
          $ENERGIZE;CALL=ENERGIZETRAP+20(104420) TURN ON MEMORY MANAGEMENT & TRAPS
          $DEENERGI;CALL=DEENERGITRAP+21(104421) TURN OFF MEMORY MANAGEMENT & TRAPS

          $KMAP   ;CALL=KMAP   TRAP+22(104422) MAP KERNEL 1 TO 1

          $CACHN ;CALL=CACHON  TRAP+23(104423) TURN CACHE ON
          $CACHF ;CALL=CACHOFF TRAP+24(104424) TURN CACHE OFF

          $LOADC ;CALL=LOADCSR  TRAP+25(104425) LOAD CORRECT CSR
          $READC ;CALL=READCSR  TRAP+26(104426) READ CORRECT CSR

          $PER01 ;CALL=PERR01  TRAP+27(104427) PROGRAM DETECTED ERROR
          $PER02 ;CALL=PERR02  TRAP+30(104430) PROGRAM DETECTED ERROR
          $PER03 ;CALL=PERR03  TRAP+31(104431) PROGRAM DETECTED ERROR
          $PER04 ;CALL=PERR04  TRAP+32(104432) PROGRAM DETECTED ERROR
          $PER07 ;CALL=PERR07  TRAP+33(104433) PROGRAM DETECTED ERROR
          $PER10 ;CALL=PERR10  TRAP+34(104434) PROGRAM DETECTED ERROR
          $PER11 ;CALL=PERR11  TRAP+35(104435) PROGRAM DETECTED ERROR
          $PER12 ;CALL=PERR12  TRAP+36(104436) PROGRAM DETECTED ERROR
          $PER13 ;CALL=PERR13  TRAP+37(104437) PROGRAM DETECTED ERROR
          $PER14 ;CALL=PERR14  TRAP+40(104440) PROGRAM DETECTED ERROR
          $PER15 ;CALL=PERR15  TRAP+41(104441) PROGRAM DETECTED ERROR
          $PER16 ;CALL=PERR16  TRAP+42(104442) PROGRAM DETECTED ERROR
          $PER17 ;CALL=PERR17  TRAP+43(104443) PROGRAM DETECTED ERROR
          $PER20 ;CALL=PERR20  TRAP+44(104444) PROGRAM DETECTED ERROR
          $PER21 ;CALL=PERR21  TRAP+45(104445) PROGRAM DETECTED ERROR
          $PER22 ;CALL=PERR22  TRAP+46(104446) PROGRAM DETECTED ERROR
          $PER23 ;CALL=PERR23  TRAP+47(104447) PROGRAM DETECTED ERROR
          $PER24 ;CALL=PERR24  TRAP+50(104450) PROGRAM DETECTED ERROR
          $PER25 ;CALL=PERR25  TRAP+51(104451) PROGRAM DETECTED ERROR
    
```

Address	Hex	Hex	Label	Description
11565	062770	054562	\$PER26	;CALL=PERR26 TRAP+52(104452) PROGRAM DETECTED ERROR
11566	062772	054602	\$PER27	;CALL=PERR27 TRAP+53(104453) PROGRAM DETECTED ERROR
11567	062774	051356	\$PER30	;CALL=PERR30 TRAP+54(104454) PROGRAM DETECTED ERROR
11568	062776	054772	\$PER31	;CALL=PERR31 TRAP+55(104455) PROGRAM DETECTED ERROR
11569	063000	055070	\$PER32	;CALL=PERR32 TRAP+56(104456) PROGRAM DETECTED ERROR
11570	063002	055136	\$PER33	;CALL=PERR33 TRAP+57(104457) PROGRAM DETECTED ERROR
11571	063004	055216	\$PER34	;CALL=PERR34 TRAP+60(104460) PROGRAM DETECTED ERROR
11572	063006	055250	\$PER35	;CALL=PERR35 TRAP+61(104461) PROGRAM DETECTED ERROR
11573	063010	055304	\$PER36	;CALL=PERR36 TRAP+62(104462) PROGRAM DETECTED ERROR
11574	063012	062636	\$NOTRAP	;CALL=PERR37 TRAP+63(104463) PROGRAM DETECTED ERROR
11575	063014	062636	\$NOTRAP	;CALL=PERR40 TRAP+64(104464) PROGRAM DETECTED ERROR
11576	063016	062636	\$NOTRAP	;CALL=PERR41 TRAP+65(104465) PROGRAM DETECTED ERROR
11577	063020	062636	\$NOTRAP	;CALL=PERR42 TRAP+66(104466) PROGRAM DETECTED ERROR
11578	063022	062636	\$NOTRAP	;CALL=PERR43 TRAP+67(104467) PROGRAM DETECTED ERROR
11579				
11580	063024	040570	\$ECCDIS	;CALL=ECCDIS TRAP+70(104470) DISABLE ECC ON ALL CSR'S
11581	063026	040604	\$ECC1DIS	;CALL=ECC1DIS TRAP+71(104471) DISABLE ECC ON 1 SELECTED CSR
11582	063030	040616	\$ECCINIT	;CALL=ECCINIT TRAP+72(104472) INITIALIZE ALL MK11 CSR'S
11583	063032	040632	\$ECC1INIT	;CALL=ECC1INIT TRAP+73(104473) INITIALIZE 1 SELECTED MK11 CSR
11584	063034	040672	\$CBCSR	;CALL=CBCSR TRAP+74(104474) WRITE GENERATED CHECKBITS IN ALL CSR'S
11585	063036	040714	\$CB1CSR	;CALL=CB1CSR TRAP+75(104475) WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
11586	063040	040734	\$WASSBE	;CALL=WASSBE TRAP+76(104476) WAS THERE A SBE ON ANY CSR?
11587	063042	041050	\$WAS1SBE	;CALL=WAS1SBE TRAP+77(104477) WAS THERE A SBE ON 1 SELECTED CSR?
11588	063044	041100	\$WASDBE	;CALL=WASDBE TRAP+100(104500) WAS THERE A DBE ON ANY CSR?
11589	063046	041214	\$WAS1DBE	;CALL=WAS1DBE TRAP+101(104501) WAS THERE A DBE ON 1 SELECTED CSR?
11590	063050	041244	\$CLRCSR	;CALL=CLRCSR TRAP+102(104502) CLEAR ALL CSR'S
11591	063052	041256	\$CLR1CSR	;CALL=CLR1CSR TRAP+103(104503) CLEAR 1 SELECTED CSR
11592	063054	041266	\$CHKDIS	;CALL=CHKDIS TRAP+104(104504) DISABLE ECC & WRITE CKBITS FROM ALL CSR'S
11593	063056	041302	\$CHK1DIS	;CALL=CHK1DIS TRAP+105(104505) DISABLE ECC & WRITE CKBITS FROM 1 CSR
11594	063060	040644	\$ENASBE	;CALL=ENASBE TRAP+106(104506) ENABLE TRAPS ON SBE'S FROM ALL CSR'S
11595	063062	040660	\$ENA1SBE	;CALL=ENA1SBE TRAP+107(104507) ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
11596	063064	040364	\$TSTRD	;CALL=TSTREAD TRAP+110(104510) TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES
11597	063066	041362	\$INVALID	;CALL=INVALID TRAP+111(104511) INVALIDATE BACKGROUND PATTERN ON BANK
11598	063070	041412	\$ERRGEN	;CALL=ERRGEN TRAP+114(104512) TEST ERROR ADDRESS
11599	063072	062636	\$NOTRAP	
11600	063074	062636	\$NOTRAP	
11601	063076	062636	\$NOTRAP	
11602	063100	062636	\$NOTRAP	
11603	063102	062636	\$NOTRAP	
11604	063104	062636	\$NOTRAP	
11605	063106	062636	\$NOTRAP	

11608

177776

ST

=

177776

;STATUS REGISTER

.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

11611		
11612		
11613		
11614		
11615		
11616		
11617		
11618		
11619		
11620		
11621		
11622		
11623		
11624		
11625	063110	
11626	063110	065401
11627	063112	067526
11628	063114	063760
11629	063116	064316
11630		
11631	063120	064355
11632	063122	067035
11633	063124	063610
11634	063126	064174
11635		
11636	063130	064413
11637	063132	067115
11638	063134	063622
11639	063136	064311
11640		
11641	063140	064445
11642	063142	067115
11643	063144	063632
11644	063146	064311
11645		
11646	063150	064513
11647	063152	067151
11648	063154	063642
11649	063156	064174
11650		
11651	063160	064570
11652	063162	067151
11653	063164	063642
11654	063166	064174
11655		
11656	063170	064615
11657	063172	067151
11658	063174	063642
11659	063176	064174
11660		
11661	063200	066733
11662	063202	070171
11663	063204	064136
11664	063206	064174

\$ERRTB:	;ERROR	1
	EM24	
	DH13	
	DT13	
	DF11	
	;ERROR	2
	EM2	
	DH1	
	DT1	
	DF2	
	;ERROR	3
	EM3	
	DH3	
	DT3	
	DF9	
	;ERROR	4
	EM4	
	DH3	
	DT4	
	DF9	
	;ERROR	5
	EM5	
	DH5	
	DT5	
	DF2	
	;ERROR	6
	EM6	
	DH5	
	DT5	
	DF2	
	;ERROR	7
	EM7	
	DH5	
	DT5	
	DF2	
	;ERROR	10
	EM53	
	DH25	
	DT25	
	DF2	

11667			:ERROR	11
11668	063210	064655	EM11	
11669	063212	067275	DH7	
11670	063214	063674	DT7	
11671	063216	064220	DF3	
11672			:ERROR	12
11673	063220	064655	EM11	
11674	063222	067275	DH7	
11675	063224	063674	DT7	
11676	063226	064233	DF4	
11677			:ERROR	13
11678	063230	064677	EM12	
11679	063232	067405	DH10	
11680	063234	063724	DT10	
11681	063236	064174	DF2	
11682			:ERROR	14
11683	063240	064655	EM11	
11684	063242	067275	DH7	
11685	063244	063674	DT7	
11686	063246	064246	DF5	
11687			:ERROR	15
11688	063250	064655	EM11	
11689	063252	067275	DH7	
11690	063254	063674	DT7	
11691	063256	064261	DF6	
11692			:ERROR	16
11693	063260	064723	EM13	
11694	063262	067526	DH13	
11695	063264	063760	DT13	
11696	063266	064316	DF11	
11697			:ERROR	17
11698	063270	064755	EM14	
11699	063272	067526	DH13	
11700	063274	063760	DT13	
11701	063276	064316	DF11	
11702			:ERROR	20
11703	063300	065021	EM15	
11704	063302	067526	DH13	
11705	063304	063760	DT13	
11706	063306	064316	DF11	
11707			:ERROR	21
11708	063310	066762	EM55	
11709	063312	070215	DH26	
11710	063314	064146	DT26	
11711	063316	064174	DF2	
11712			:ERROR	22
11713	063320	065067	EM17	
11714	063322	067275	DH7	
11715	063324	063674	DT7	
11716	063326	064246	DF5	
11717			:ERROR	23
11718	063330	066570	EM50	
11719	063332	070043	DH23	
11720	063334	064074	DT23	
11721	063336	064327	DF13	

11724			:ERROR 24	
11725	063340	065127	EM19	
11726	063342	067526	DH13	
11727	063344	063760	DT13	
11728	063346	064316	DF11	
11729			:ERROR 25	
11730	063350	065204	EM20	
11731	063352	067526	DH13	
11732	063354	063760	DT13	
11733	063356	064316	DF11	
11734			:ERROR 26	
11735	063360	000000	0	;NO MESSAGE
11736	063362	067521	DH12	
11737	063364	063754	DT12	
11738	063366	064174	DF2	
11739			:ERROR 27	
11740	063370	065266	EM21	
11741	063372	067503	DH11	
11742	063374	063746	DT11	
11743	063376	064174	DF2	
11744			:ERROR 30	
11745	063400	065325	EM22	
11746	063402	067526	DH13	
11747	063404	063760	DT13	
11748	063406	064316	DF11	
11749			:ERROR 31	
11750	063410	000000	0	;NO MESSAGE
11751	063412	067623	DH14	
11752	063414	064002	DT14	
11753	063416	064174	DF2	
11754			:ERROR 32	
11755	063420	065352	EM23	
11756	063422	067151	DH5	
11757	063424	063642	DT5	
11758	063426	064174	DF2	
11766			:ERROR 33	
11767	063430	065460	EM25	
11768	063432	067702	DH15	
11769	063434	064020	DT16	
11770	063436	064274	DF7	
11771			:ERROR 34	
11772	063440	065505	EM26	
11773	063442	070021	DH16	
11774	063444	064050	DT17	
11775	063446	064220	DF3	

11785			:ERROR	35
11786	063450	066706	EM52	
11787	063452	070171	DH25	
11788	063454	064136	DT25	
11789	063456	064174	DF2	
11790			:ERROR	36
11791	063460	065556	EM27	
11792	063462	070021	DH16	
11793	063464	064050	DT17	
11794	063466	064307	DF8	
11802			:ERROR	37
11803	063470	066362	EM35	
11804	063472	067275	DH7	
11805	063474	063674	DT7	
11806	063476	064220	DF3	
11807			:ERROR	40
11808	063500	065646	EM29	
11809	063502	067275	DH7	
11810	063504	063674	DT7	
11811	063506	064220	DF3	
11812			:ERROR	41
11813	063510	065730	EM30	
11814	063512	067275	DH7	
11815	063514	063674	DT7	
11816	063516	064246	DF5	
11817			:ERROR	42
11818	063520	066047	EM31	
11819	063522	067275	DH7	
11820	063524	063674	DT7	
11821	063526	064220	DF3	
11822			:ERROR	43
11823	063530	066147	EM32	
11824	063532	067275	DH7	
11825	063534	063674	DT7	
11826	063536	064220	DF3	
11827			:ERROR	44
11828	063540	066254	EM33	
11829	063542	067275	DH7	
11830	063544	063674	DT7	
11831	063546	064220	DF3	
11832			:ERROR	45
11833	063550	066624	EM51	
11834	063552	070122	DH24	
11835	063554	064116	DT24	
11836	063556	064337	DF14	
11837			:ERROR	46
11838	063560	066447	EM36	
11839	063562	067230	DH6	
11840	063564	063660	DT6	
11841	063566	064174	DF2	



11856			:ERROR 47
11857	063570	066516	EM40
11858	063572	067072	DH2
11859	063574	064056	DT20
11860	063576	064174	DF2
11898			:ERROR 50
11899	063600	067003	EM56
11900	063602	070233	DH27
11901	063604	064154	DT27
11902	063606	064346	DF15

11912						.SBITL	ERROR DATA TAGS (DT)
11913	063610	002016	002032	002042	DT1:	.WORD	ERRPC,ADDRESS,GOOD,BAD,0
	063616	002050	000000				
11917	063622	002016	002034	002070	DT3:	.WORD	ERRPC,PADDRESS,PARCNT,0
	063630	000000					
11918	063632	002016	002032	002066	DT4:	.WORD	ERRPC,ADDRESS,NEMCNT,0
	063640	000000					
11919	063642	002016	177572	177574	DT5:	.WORD	ERRPC,MMRO,MMR1,MMR2,MMR3,CPUERR,0
	063650	177576	172516	177766			
	063656	000000					
11920	063660	002016	002372	002350	DT6:	.WORD	ERRPC,APTPAR,LSIZE,APTECC,MSIZE,0
	063666	002374	002352	000000			
11921	063674	002016	002170	002032	DT7:	.WORD	ERRPC,DUMMY,ADDRESS,DUMMY,GOOD,BAD,BADXOR
	063702	002170	002042	002050			
	063710	002056					
11922	063712	002170	002170	002170		.WORD	DUMMY,DUMMY,DUMMY,DUMMY,0
	063720	002170	000000				
11923	063724	002172	002174	002176	DT10:	.WORD	DETRO,DETR1,DETR2,DETR3,DETR4,DETR5,DEI SP,DEI PSW,0
	063732	002200	002202	002204			
	063740	002206	002210	000000			
11924	063746	002016	002144	000000	DT11:	.WORD	ERRPC,CSR,0
11925	063754	002144	000000		DT12:	.WORD	CSR,0
11926	063760	002016	002170	002032	DT13:	.WORD	ERRPC,DUMMY,ADDRESS,DUMMY,TST DAT,TST DAT+2,CHECK,CSR,0
	063766	002170	002240	002242			
	063774	002274	002144	000000			
11927	064002	177746	177572	177574	DT14:	.WORD	CONTRL,MMRO,MMR1,MMR2,MMR3,CPUERR,0
	064010	177576	172516	177766			
	064016	000000					
11928	064020	002016	002170	002170	DT16:	.WORD	ERRPC,DUMMY,DUMMY,GOOD,GOOD2,GOOD3
	064026	002042	002044	002046			
11929	064034	002050	002052	002054		.WORD	BAD,BAD2,BAD3,DUMMY,DUMMY,0
	064042	002170	002170	000000			
11930	064050	002016	002170	000000	DT17:	.WORD	ERRPC,DUMMY,0
11935	064056	002016	002042	002050	DT20:	.WORD	ERRPC,GOOD,BAD,0
	064064	000000					
11939	064066	002016	002170	000000	DT22:	.WORD	ERRPC,DUMMY,0
11940	064074	002016	002170	002042	DT23:	.WORD	ERRPC,DUMMY,GOOD,BAD,DUMMY,DUMMY,DUMMY,DUMMY,0
	064102	002050	002170	002170			
	064110	002170	002170	000000			
11941	064116	002016	002170	002144	DT24:	.WORD	ERRPC,DUMMY,CSR,DUMMY,DUMMY,DUMMY,DUMMY,0
	064124	002170	002170	002170			
	064132	002170	000000				
11942	064136	002016	002042	002144	DT25:	.WORD	ERRPC,GOOD,CSR,0
	064144	000000					
11943	064146	002016	002050	000000	DT26:	.WORD	ERRPC,BAD,0
11944	064154	002016	002170	002032	DT27:	.WORD	ERRPC,DUMMY,ADDRESS,DUMMY,DUMMY,DUMMY,DUMMY,0
	064162	002170	002170	002170			
	064170	002170	000000				



Line No	Address	Code	Msg No	Msg Name	Description
11969				.SBTTL	ERROR MESSAGES (EM)
11978	064355	103	101	EM2:	.ASCIZ /CAN'T SET 22 BIT MODE IN MMR3/
11979	064413	120	101	EM3:	.ASCIZ /PARITY ERROR(S) IN BANK 0/
11980	064445	116	117	EM4:	.ASCIZ /NON-EXISTANT MEMORY (HOLES) IN BANK 0/
11981	064513	111	114	EM5:	.ASCIZ /ILLEGAL OR RESERVED INSTRUCTION (TRAP TO 10)/
11982	064570	125	116	EM6:	.ASCIZ /UNEXPECTED TRAP TO 4/
11983	064615	115	105	EM7:	.ASCIZ /MEMORY MANAGEMENT (TRAP TO 250)/
11987					
11988	064655	115	105	EM11:	.ASCIZ /MEMORY DATA ERROR/
11989	064677	104	105	EM12:	.ASCIZ /DETAILED ERROR DUMP/
11990	064723	115	111	EM13:	.ASCIZ /MISSING EXPECTED SBE FLAG/
11991	064755	127	122	EM14:	.ASCIZ /WRITE BYTE FAILED TO CLEAR SBE FLAG/
11992	065021	106	101	EM15:	.ASCIZ /FAILED TO GET INTERRUPT WITH DBE FLAG/
11993	065067	115	105	EM17:	.ASCIZ /MEMORY DATA ERROR IN CHECK BITS/
11994	065127	123	102	EM19:	.ASCIZ /SBE OR DBE CAUSED PARITY TRAP WHEN INHIBITED/
11995	065204	123	102	EM20:	.ASCIZ /SBE OR DBE DID NOT CAUSE PARITY TRAP WHEN ENABLED/
11996	065266	123	102	EM21:	.ASCIZ /SBE OR DBE ON MASTER TEST WORD/
11997	065325	115	111	EM22:	.ASCIZ /MISSING EXPECTED DBE/
11998	065352	125	116	EM23:	.ASCIZ /UNEXPECTED PARITY TRAP/
11999	065401	122	105	EM24:	.ASCIZ /RECEIVED DBE FLAG WHEN EXPECTING ONLY SBE FLAG/
12000	065460	103	110	EM25:	.ASCIZ /CHECK BIT DATA ERROR/
12001	065505	101	104	EM26:	.ASCIZ /ADDRESS PARITY ERROR DID NOT CAUSE ABORT/
12002	065556	105	103	EM27:	.ASCIZ /ECC INHIBIT MODE POINTER FAILURE - DID NOT PROTECT BANK/
12006	065646	103	117	EM29:	.ASCIZ /CORRECTION FAILURE WITH ECC ENABLED ON FORCED SBE/
12007	065730	127	122	FM30:	.ASCII /WRITE BYTE (MOVB) WITH ECC ENABLED FAILED TO CLEAR DATA AT/<<CRLF>
12008	066023	106	117		.ASCIZ /FORCED SBE LOCATION/
12009	066047	101	123	EM31:	.ASCIZ /ASRB (R3)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
12010	066147	115	117	EM32:	.ASCIZ /MOVB #360,(R2)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
12011	066254	115	117	EM33:	.ASCIZ /MOV #177400,(R1) WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
12012	066362	125	116	EM35:	.ASCIZ /UNEXPECTED CORRECTION WITH ECC DISABLE ON FORCED SBE/
12013	066447	101	120	EM36:	.ASCIZ /APT SIZE DISAGREES WITH PROGRAM SIZING/
12019	066516	102	122	EM40:	.ASCIZ /BRANCH GOBBLE FAILED (CONDITION CODES TEST/
12028	066570	102	101	EM50:	.ASCIZ /BAD ERROR ADDRESS GENERATED/
12029	066624	123	102	EM51:	.ASCIZ /SBE & DBE FLAGS NOT SET ON FORCED UNCORRECTED SBE/
12030	066706	102	111	EM52:	.ASCIZ /BIT SET ERROR IN CSR/
12031	066733	102	111	EM53:	.ASCIZ /BIT CLEAR ERROR IN CSR/
12032	066762	111	114	EM55:	.ASCIZ /ILLEGAL CSR TYPE/
12033	067003	102	101	EM56:	.ASCIZ /BAD PARITY TRAP GENERATED/

Address	Hex	Hex	Hex	Label	Format	PC	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10	Field 11	Field 12	
12039				.SBTTL	ERROR DATA HEADERS (DH)														
12040	067035	040	040	DH1:	.ASCIZ / PC		DEV ADD	GOOD	BAD/										
12041	067072	040	040	DH2:	.ASCIZ / PC		GD-CC	BD-CC/											
12042	067115	040	040	DH3:	.ASCIZ / PC		1ST ADD	# OF ERRORS/											
12043	067151	040	040	DH5:	.ASCIZ / PC		MMRO	MMR1	MMR2	MMR3	CPUERR/								
12044	067230	040	040	DH6:	.ASCIZ / PC		APTPAR	LSIZE	APTECC	MSIZE/									
12045	067275	040	040	DH7:	.ASCII / PC		BANK	VADD	PADD	GOOD/									
12046	067341	040	040		.ASCIZ /		BAD	XOR	CSR	MTYP	INT	PAT/							
12047	067405	040	040	DH10:	.ASCIZ / RO		R1	R2	R3	R4	R5	SP	PSW/						
12048	067503	040	040	DH11:	.ASCIZ / PC		CSR/												
12049	067521	040	103	DH12:	.ASCIZ /		CSR/												
12050	067526	040	040	DH13:	.ASCII / PC		BANK	VADD	PADD	WROTE1	WROTE2/								
12051	067603	040	103		.ASCIZ /		CHKBITS	CSR/											
12052	067623	103	117	DH14:	.ASCIZ /		CONTRL	MMRO	MMR1	MMR2	MMR3	CPUERR/							
12053	067702	040	040	DH15:	.ASCII / PC		BANK	PADD	GD-WD1	GD-WD2	GD-CHK/								
12054	067757	040	102		.ASCIZ /		BAD-WD1	BAD-WD2	BAD-CHK	INT	PAT/								
12055	070021	040	040	DH16:	.ASCIZ / PC		BANK/												
12060	070036	040	040	DH19:	.ASCIZ /		PC/												
12066	070043	040	040	DH23:	.ASCIZ / PC		BANK	GD-ERR	BAD-ERR	CSR	MTYP	INT	PAT/						
12067	070122	040	040	DH24:	.ASCIZ / PC		BANK	(CSR)	CSR	MTYP	INT	PAT/							
12068	070171	040	040	DH25:	.ASCIZ / PC		GD-DAT	(CSR)/											
12069	070215	040	040	DH26:	.ASCIZ / PC		BADCODE/												
12070	070233	040	040	DH27:	.ASCIZ / PC		BANK	VADD	PADD	CSR	MTYP	PAT/							

```

12073
12074 070307      200      040      103 MSG000: .SBTTL  MESSAGES
12075 070354      200      040      040 MSG001: .ASCIZ  <CRLF>' CZMSDB - MS11L/M MEMORY DIAGNOSTIC''
12076 070436      200      040      040 MSG002: .ASCIZ  <CRLF>/                                MEMORY CONFIGURATION MAP/
12077 070513      200      040      040 MSG003: .ASCIZ  <CRLF>/                                16K WORD BANKS/
12078 070555      040      040      040      .ASCIZ  /          4          5          6          7          /
12079 070620      200      040      040 MSG004: .ASCIZ  <CRLF>/                                012345670123456701234567/
12080 070661      060      061      062      .ASCIZ  /01234567012345670123456701234567012345670123/
12081 070726      200      105      122 MSG005: .ASCIZ  <CRLF>/ERRORS /
12082 070740      200      125      116 MSG006: .ASCIZ  <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>
12083 070775      200      111      116 MSG007: .ASCIZ  <CRLF>/INTRLV /                                ;INTERLEAVED CSR #
12084 071007      200      103      120 MSG008: .ASCIZ  <CRLF>/CPU MAP /                                ;CPU ACCESSED BANK
12085 071021      200      115      105 MSG009: .ASCIZ  <CRLF>/MEMTYPE /                                ;MEMORY TYPE
12086 071033      200      120      122 MSG010: .ASCIZ  <CRLF>/PROTECT /                                ;MEMORY PROTECTED
12087 071045      040      040      040 MSG011: .ASCIZ  /          0          1          2          3          4          5          6/
12088 071133      064      065      066 MSG012: .ASCIZ  /4567012345670123456701234567012345670123456701234567/
12089 071230      130      000
12090 071232      040      000
12091 071234      000      000
12092 071236      200      103      123 MSG013: .ASCIZ  /X/
12093 071250      040      040      040 MSG014: .ASCIZ  / /                                ;SPACE
12094 071261      040      040      000 MSG015: .BYTE   0,0                                ;FOR SINGLE ASCII CHARACTERS & TERMINATOR
12095 071264      040      040      040 MSG016: .ASCIZ  <CRLF>/CSR /
12096 071270      200      106      123 MSG017: .ASCIZ  / / /                                ;8 SPACES
12097 071311      200      103      117 MSG018: .ASCIZ  / / /                                ;2 SPACES
12098 071335      200      060      040 MSG019: .ASCIZ  / / /                                ;3 SPACES
12099 071346      200      061      040 MSG020: .ASCIZ  <CRLF>/FS COMMAND MODE/
12100 071363      200      062      040 MSG021: .ASCII  <CRLF>/COMMANDS AVAILABLE:/
12101 071400      200      063      040      .ASCII  <CRLF>/0 = EXIT/
12102 071423      200      064      040      .ASCII  <CRLF>/1 = READ CSR/
12103 071445      200      065      040      .ASCII  <CRLF>/2 = LOAD CSR/
12104 071477      200      066      040      .ASCII  <CRLF>/3 = EXAMINE MEMORY/
12105 071523      200      067      040      .ASCII  <CRLF>/4 = MODIFY MEMORY/
12106 071547      200      070      040      .ASCII  <CRLF>/5 = SELECT BANK & PATTERN/
12107 071571      200      071      075      .ASCII  <CRLF>/6 = TYPE CONFIG MAP/
12108 071611      200      061      060      .ASCII  <CRLF>/7 = SOB-A-LONG TEST/
12109 071634      200      061      061      .ASCII  <CRLF>/8 = ERROR SUMMARY/
12110 071664      200      061      062      .ASCII  <CRLF>/9= REFRESH TEST/
12111 071713      200      061      063      .ASCII  <CRLF>/10= SET FILL COUNT/
12112 071736      200      061      064      .ASCII  <CRLF>/11= ENTER KAMIKAZE MODE/
12117 071760      200      061      065      .ASCII  <CRLF>/12= EXIT KAMIKAZE MODE/
12118 072010      200      061      066      .ASCII  <CRLF>/13= TURN CACHE OFF/
12119 072033      200      061      067      .ASCII  <CRLF>/14= TURN CACHE ON/
12120 072054      200      061      070      .ASCII  <CRLF>/15= TEST SELECTED BANKS/
12121 072076      015      012      000      .ASCII  <CRLF>/16= TEST ALL BANKS/
12122 072101      200      127      110 MSG022: .BYTE   15,12,0
12123 072123      200      103      123 MSG023: .ASCIZ  <CRLF>/WHICH CSR(0-F)? /
12124 072137      200      103      123 MSG024: .ASCIZ  <CRLF>/CSR WORD? /
12125 072163      200      103      117 MSG025: .ASCIZ  <CRLF>/CSR DOES NOT EXIST/
12126 072175      200      117      114 MSG026: .ASCIZ  <CRLF>/COMMAND:/
12127 072212      200      103      123 MSG027: .ASCIZ  <CRLF>/OLD CSR WAS/
12128 072226      200      105      130 MSG028: .ASCIZ  <CRLF>/CSR IS NOW/
12129 072246      200      102      101 MSG029: .ASCIZ  <CRLF>/EXAMINE MEMORY/
12130 072265      200      120      110 MSG030: .ASCIZ  <CRLF>/BANK(0-177)? /
12131 072325      200      120      101 MSG031: .ASCIZ  <CRLF>/PHYSICAL ADDRESS(0-17757776)? /
12132 072344      200      124      111 MSG032: .ASCIZ  <CRLF>/PARITY ABORT/<32>
12133 072363      200      102      131 MSG033: .ASCIZ  <CRLF>/TIMEOUT TRAP/<32>
MSG034: .ASCIZ  <CRLF>/BYPASSING ECC LOGIC TESTS ON BANK /

```

12134	072427	040	104	125	MSGB34:	.ASCIZ	/ DUE TO LACK OF SBE FREE LOCATIONS/
12135	072472	121	126	000	MSG035:	.ASCIZ	/QV/
12136	072475	200	115	117	MSG036:	.ASCIZ	<CRLF>/MODIFY MEMORY/
12137	072514	200	117	114	MSG037:	.ASCIZ	<CRLF>/OLD DATA WAS /
12138	072533	200	104	101	MSG038:	.ASCIZ	<CRLF>/DATA IS NOW /
12139	072551	200	111	116	MSG039:	.ASCIZ	<CRLF>/INPUT NEW DATA? /
12140	072573	200	123	105	MSG040:	.ASCIZ	<CRLF>/SELECT BANK & PATTERN TEST/
12141	072627	200	102	101	MSG041:	.ASCIZ	<CRLF>/BANK NOT ACCESSABLE/
12142	072654	200	120	101	MSG042:	.ASCIZ	<CRLF>/PATTERN(0-35)? /
12143	072675	200	120	101	MSG043:	.ASCIZ	<CRLF>/PATTERN 0 DATA IS? /
12144	072722	200	124	117	MSG046:	.ASCIZ	<CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
12145	072755	200	124	105	MSG047:	.ASCIZ	<CRLF>/TEST COMPLETE/
12146	072774	040	116	117	MSG048:	.ASCIZ	/ NOT AVAILABLE NOW - TRY LATER! /
12147	073034	200	102	101	MSG049:	.ASCIZ	<CRLF>/BANK REQUIRES RELOCATION/
12148	073066	200	102	101	MSG050:	.ASCII	<CRLF>/BATTERY BACKUP TEST/
12149	073112	200	127	122	.ASCIZ	<CRLF>/WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND/	
12150	073174	200	120	117	MSG051:	.ASCIZ	<CRLF>/POWER RECOVERY/
12151	073214	200	122	105	MSG052:	.ASCIZ	<CRLF>/REMOVE SYSTEM POWER FOR/
12152	073245	040	123	105	MSG053:	.ASCIZ	/ SECONDS MAX! /
12153	073263	200	116	117	MSG054:	.ASCIZ	<CRLF>/NOW STARTING READ TEST OF MEMORY BANKS/
12154	073333	200	123	117	MSG055:	.ASCIZ	<CRLF>/SOB-A-LONG TEST/
12155	073354	200	102	105	MSG056:	.ASCIZ	<CRLF>/BELL = EACH PASS COMPLETE/
12156	073407	200	040	040	MSG058:	.ASCIZ	<CRLF>/ CSR CSR .../
12157	073431	077	077	077	MSG061:	.ASCIZ	/?????/?/
12158	073440	111	116	120	MSG062:	.ASCIZ	/INPUT MUST BE A/
12159	073460	116	040	117	MSG063:	.ASCIZ	/N OCTAL /
12160	073471	116	125	115	MSG064:	.ASCIZ	/NUMBER/<CRLF>
12161	073501	040	104	105	MSG065:	.ASCIZ	/ DECIMAL /
12162	073513	200	105	122	MSG066:	.ASCIZ	<CRLF>/ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN/
12163	073576	106	101	124	MSG067:	.ASCIZ	/FATAL /
12164	073605	113	040	127	MSG070:	.ASCIZ	/K WORDS OF MEMORY TOTAL/<CRLF>
12165	073636	113	040	117	MSG071:	.ASCIZ	/K OF BIPOLAR/<CRLF>
12166	073654	113	040	117	MSG072:	.ASCIZ	/K OF MF11S-K/<CRLF>
12167	073672	200	122	105	MSG073:	.ASCIZ	<CRLF>/REFRESH TEST/
12168	073710	200	122	105	MSG075:	.ASCIZ	<CRLF>/RELOCATION NOT POSSIBLE/<32>
12169	073742	200	040	040	MSG076:	.ASCIZ	<CRLF>/ BANK ERRORS/<CRLF>
12170	073763	200	105	116	MSG077:	.ASCIZ	<CRLF>/END PASS #/
12171	073777	040	105	122	MSG079:	.ASCIZ	/ ERROR(S) DETECTED/<CRLF>
12178	074023	200	106	111	MSG085:	.ASCIZ	<CRLF>/FILL COUNT(OCTAL)? /
12186	074050	200	113	105	MSG088:	.ASCIZ	<CRLF>/KERNEL STACK/
12187	074066	200	123	125	MSG089:	.ASCIZ	<CRLF>/SUPERVISOR STACK/
12188	074110	200	125	123	MSG090:	.ASCIZ	<CRLF>/USER STACK/
12189	074124	040	111	123	MSG091:	.ASCIZ	/ IS EMPTY/
12190	074136	122	105	114	MSG092:	.ASCIZ	/RELOCATED /
12191	074152	102	101	116	MSG093:	.ASCIZ	/BANK=/
12192	074160	040	040	120	MSG095:	.ASCIZ	/ PAT=/
12200	074167	200	105	116	MSG101:	.ASCIZ	<CRLF>/ENTERING KAMIKAZE MODE/
12201	074217	200	114	105	MSG102:	.ASCIZ	<CRLF>/LEAVING KAMIKAZE MODE/
12202	074246	200	114	105	MSG103:	.ASCIZ	<CRLF>/LEAVING FS MODE/<CRLF>
12203	074270	032	000		MSG104:	.BYTE	32,0 ;CONTROL Z
12204	074272	200	105	116	MSG105:	.ASCIZ	<CRLF>/ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINATE (177)/
12205	074376	200	103	101	MSG106:	.ASCIZ	<CRLF>/CACHE IS OFF/
12206	074414	200	103	101	MSG107:	.ASCIZ	<CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
12211	074471	200	117	116	MSG110:	.ASCIZ	<CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
12212	074535	200	101	114	MSG111:	.ASCIZ	<CRLF>/ALL BANKS WILL BE TESTED/
12213	074567	113	040	117	MSG112:	.ASCIZ	/K OF MS11-L/<CRLF>
12214	074604	113	040	117	MSG113:	.ASCIZ	/K OF MS11-M/<CRLF>

12215	074621	113	040	117	MSG114:	.ASCIZ	/K OF UNIBUS PARITY/<CRLF>
12216	074645	200	040	040	MSG116:	.ASCIZ	<CRLF>" 11/34"
12217	074657	200	040	040	MSG117:	.ASCIZ	<CRLF>" 11/44"
12218	074671	200	040	040	MSG118:	.ASCIZ	<CRLF>" 11/60"
12219	074703	200	040	040	MSG119:	.ASCIZ	<CRLF>/ NO/
12220	074712	040	103	101	MSG120:	.ASCIZ	/ CACHE AVAILABLE/
12221	074733	040	103	101	MSG121:	.ASCIZ	/ CACHE BYPASSED/
12222	074753	200	103	123	MSG122:	.ASCII	<CRLF>/CSR NUMBER /
12223	074767	000			MSG122:	.BYTE	0
12224	074770	040	103	117		.ASCIZ	/ CONTROLS TOO MANY BANKS/
12225	075021	200	120	122	MSG123:	.ASCIZ	<CRLF>/PROGRAM RELOCATED - ECC TESTS INHIBITED/
12226	075072	200	116	125	MSG124:	.ASCIZ	<CRLF>/NUMBER OF CSR'S IS WRONG IN BANK /
12227	075135	040	120	101	MSG125:	.ASCIZ	/ PASSES COMPLETED/
12228	075157	200	120	122	MSG126:	.ASCIZ	<CRLF>/PROGRAM CSR COULD NOT BE DETERMINED/
12229	075224	200	124	122	MSG127:	.ASCIZ	<CRLF>/TRACE ENABLED/
12230	075243	200	124	122	MSG128:	.ASCIZ	<CRLF>/TRACE DISABLED/
12231						.EVEN	
12237	075264					\$\$END	
12238	075264				END:		
12239	075264	004124				.PRINT	60000-SUPLIMIT ;SUPERVISOR ADDPESSES LEFT
12251	075264	002514				.PRINT	100000-END ;ADDRESSES LEFT IN 16K
12255		000200				.END	START3



ABORTF	002140	B10	014560	CBCSR =	104474	DEENER=	104421	DT20	064056
ACFLAG	002114	B11	016176	CB1CSR=	104475	DETAIL	057132	DT22	064066
ACTFLA	002320	B12	016444	CHECK	002274	DETFLA	002212	DT23	064074
ADDRES	002032	B13	016452	CHKDIS=	104504	DETPSW	002210	DT24	064116
ANA2	007134	B14	016470	CHKGEN	041662	DETRO	002172	DT25	064136
APTDOW	045210	B15	016532	CHKTAB	041770	DETR1	002174	DT26	064146
APTECC	002374	B16	016604	CHK1DI=	104505	DETR2	002176	DT27	064154
APTFLA	002322	B17	022000	CKEND	060706	DETR3	002200	DT3	063622
APTHAN	014514	B2	012450	CKSWR =	104410	DETR4	002202	DT4	063632
APTHLT	045256	B20	022240	CLRCR=	104502	DETR5	002204	DT5	063642
APTPAR	002372	B21	024260	CLREX	007104	DETSP	002206	DT6	063660
APTSIZ	002414	B22	024264	CLRMEM	007004	DET1	057622	DT7	063674
BACKGN	036470	B23	024464	CLR1CS=	104503	DF11	064316	DUMMY	002170
BAD	002050	B24	024472	CMD16A	050762	DF13	064327	DUMPCS=	000061
BADPC	002020	B25	024770	CMD16L=	000052	DF14	064337	ECCDIS=	104470
BADPSW	002030	B26	034410	CMD5B	047200	DF15	064346	ECCINI=	104472
BADSP	002024	B27	034446	CMD5C	047450	DF2	064174	ECC1DI=	104471
BADSTA	040126	B3	012512	CMD7B	047700	DF3	064220	ECC1IN=	104473
BADXOR	002056	B30	034462	CMD7C	047754	DF4	064233	EMTVEC=	000030
BAD2	002052	B31	034602	CMD9B	050376	DF5	064246	EM11	064655
BAD3	002054	B32	034762	CMD9C	050452	DF6	064261	EM12	064677
BAFPAF	014650	B33	035004	CONFGE	002420	DF7	064274	EM13	064723
BAFPAR	014756	B34	036200	CONFIE	003630	DF8	064307	EM14	064755
BAKPAT	002572	B35	040746	CONFIG	002624	DF9	064311	EM15	065021
BANK	002100	B36	040752	CONTFI	002214	DH1	067035	EM17	065067
BANKIN	002102	B37	041112	CONTRL=	177746	DH10	067405	EM19	065127
BANKMO	043710	B4	013316	CONTS	061056	DH11	067503	EM2	064355
BANKOK	044710	B40	041116	CONTS1	060434	DH12	067521	EM20	065204
BAWPAF	015064	B41	041322	CONTS2	061060	DH13	067526	EM21	065266
BAWPAR	015214	B42	041326	CONTS3	053536	DH14	067623	EM22	065325
BGTEST	036046	B43	042530	CONTT	061002	DH15	067702	EM23	065352
BIT0 =	000001	B44	042536	COUNT	002340	DH16	070021	EM24	065401
BIT1 =	000002	B45	042666	CPUBIT	002104	DH19	070036	EM25	065460
BIT10 =	002000	B46	042702	CPUERR=	177766	DH2	067072	EM26	065505
BIT11 =	004000	B47	047704	CR =	000015	DH23	070043	EM27	065556
BIT12 =	010000	B5	013460	CRLF =	000200	DH24	070122	EM29	065646
BIT13 =	020000	B50	050126	CSR	002144	DH25	070171	EM3	064413
BIT14 =	040000	B51	050402	CSRADD=	172100	DH26	070215	EM30	065730
BIT15 =	100000	B52	050676	CSRCAS	017356	DH27	070233	EM31	066047
BIT2 =	000004	B53	050676	CSRFBA	002224	DH3	067115	EM32	066147
BIT3 =	000010	B54	050774	CSRFIR	002220	DH5	067151	EM33	066254
BIT4 =	000020	B55	051734	CSRHOL	002476	DH6	067230	EM35	066362
BIT5 =	000040	B56	051740	CSRINC	002300	DH7	067275	EM36	066447
BIT6 =	000100	B57	052236	CSRINF	002432	DIAGFL	002002	EM4	064445
BIT7 =	000200	B6	013522	CSRINT	002230	DISPLA	002600	EM40	066516
BIT8 =	000400	B60	052244	CSRLAS	002222	DISPRE=	000174	EM5	064513
BIT9 =	001000	B61	057366	CSRLBA	002226	DISPTB	014176	EM50	066570
BLOCK1	045260	B62	057372	CSRL00	002302	DOBACK	014550	EM51	066624
BLOCK2	045300	B63	057470	CSRNO	002146	DSWR =	177570	EM52	066706
BLOCK3	045314	B64	057562	CSROUT	041314	DT1	063610	EM53	066733
BMFLAG	002126	B65	057646	CSRSTU=	000014	DT10	063724	EM55	066762
BOOT	044766	B7	014006	CTEST	006106	DT11	063746	EM56	067003
BOOT1	045032	CACHKF	002520	CTLKVE	002142	DT12	063754	EM6	064570
BRGOBB	036050	CACHKN	002514	DATARG=	177754	DT13	063760	EM7	064615
BSIZE	002344	CACHOF=	104424	DATBUF	002234	DT14	064002	ENASBE=	104506
B0	004522	CACHON=	104423	DBEMSK	002250	DT16	064020	ENA1SB=	104507
B1	011670	CACHVE=	000114	DDISP =	177570	DT17	064050	END	075264

ENERGI= 104420	E52 050736	HIPAT 044742	LOADHO 002536	L153 021452
ENEXBK 044700	E53 050736	HOLDLO= 000011	LOOP 014150	L154 021516
ERRADD 002430	E54 051012	HT = 000011	LOWMAP 043644	L155 021526
ERRGEN= 104512	E55 051774	I 002422	LSIZE 002350	L156 021526
ERRMAX 002524	E56 051774	IIII = 177777	LWDBE = 000037	L157 021536
ERROR = 104000	E57 052300	ILLCSR 013216	LWSBE = 000035	L16 010662
ERRPC 002016	E6 013550	IMPTE\$ 012160	LO 004532	L160 021574
ERRPSW 002026	E60 052300	INCBNK 044752	L1 004574	L161 021604
ERRSP 002022	E61 057434	INCPAT 044726	L10 005072	L162 021604
ERRVEC= 000004	E62 057434	INCRPT 044726	L100 014150	L163 021642
EUFLAG 002130	E63 057520	INHBAN 002510	L101 014264	L164 021652
EVEN 002334	E64 057614	INHECC 002506	L102 014364	L165 021652
EXBANK 044240	E65 057700	INTFLA 002134	L103 014374	L166 021732
EXCMD3 046330	E7 014126	INT64K 002136	L104 014544	L167 021742
EXCMD4 046650	FASTCI= 177640	INVALI= 104511	L105 014544	L17 010664
EXIT 045074	FATAL\$ 002062	IOTVEC= 000020	L106 014632	L170 021742
EXIT2 045100	FCMD10 050536	JMPRL1 043240	L107 014744	L171 022104
EO 004550	FCMD11 050564	KAMIKA 002004	L11 005100	L172 022032
E1 012106	FCMD12 050606	KAMITE 026370	L110 015052	L173 022032
E10 014646	FCMD13 050626	KDIAG = 000010	L111 015202	L174 022064
E11 016216	FCMD14 050650	KDPAR0= 172360	L112 015332	L175 022064
E12 016770	FCMD15 050666	KDPAR6= 172374	L113 015462	L176 022142
E13 016754	FCMD16 050752	KDPAR7= 172376	L114 015634	L177 022176
E14 016710	FCMD17 051014	KERNEL= 104417	L115 015764	L2 004702
E15 016710	FCMD18 051030	KERSTK= 002000	L116 016136	L20 011476
E16 016660	FIELDS 045344	KFLAG 002500	L117 016216	L200 022200
E17 022076	FINDBA= 000045	KIPAR0= 172340	L12 005130	L201 022226
E2 012652	FINT 006404	KIPAR4= 172350	L120 016254	L202 022230
E20 022340	FIRST = 060000	KIPAR5= 172352	L121 016264	L203 022314
E21 024442	FLIPLO 002556	KIPAR6= 172354	L122 016710	L204 022426
E22 024426	FLIPWA 036340	KIPDR0= 172300	L123 016672	L205 022430
E23 024650	FLUSH 014270	KMAP = 104422	L124 016736	L206 023270
E24 024634	FSCMD0 045542	KPFLAG 002112	L125 017200	L207 023322
E25 025046	FSCMD1 045644	KSIZE 002346	L126 017310	L21 011512
E26 034754	FSCMD2 045754	KSTACK 002534	L127 017446	L210 023366
E27 034740	FSCMD3 046122	LAST = 157776	L13 005362	L211 023636
E3 012636	FSCMD4 046376	LASTBA 002526	L130 017474	L212 023646
E30 034554	FSCMD5 046716	LASTBL 002530	L131 017500	L213 023646
E31 034724	FSCMD6 047540	LASTER 002014	L132 017502	L214 024412
E32 035000	FSCMD7 047546	LBLS0 = 000455	L133 017546	L215 024360
E33 035024	FSCMD8 050040	LBLS1 = 000065	L134 017602	L216 024412
E34 036322	FSCMD9 050244	LBLS2 = 000447	L135 017606	L217 024456
E35 041020	FSINFL 002412	LBLS3 = 000442	L136 017610	L22 011636
E36 041020	FSPAT 047354	LBLS4 = 000315	L137 017756	L220 024620
E37 041164	FSSTAC 002266	LBLS5 = 000317	L14 005362	L221 024566
E4 013452	FS1 045430	LBLS6 = 000016	L140 020432	L222 024620
E40 041164	FS7FLA 002416	LCSR0U= 000041	L141 021300	L223 025032
E41 041356	FULLRE 002512	LCSRRE= 000057	L142 021310	L224 025032
E42 041356	GBLENG= 000076	LCSRSA= 000055	L143 021310	L225 025140
E43 042666	GETDAT 051424	LEGALC= 000003	L144 021346	L226 025112
E44 042646	GETDA1 051522	LF = 000012	L145 021356	L227 025170
E45 043106	GETDIS 055570	LINK1 002472	L146 021356	L23 011614
E46 043014	GOOD 002042	LINK2 002474	L147 021406	L230 025360
E47 047746	GOOD2 002044	LKS = 177546	L15 010656	L231 025712
E5 013520	GOOD3 002046	LOADBA 002402	L150 021412	L232 026202
E50 050230	GTSWR = 104407	LOADCS= 104425	L151 021442	L233 026240
E51 050444	HEADER 002552	LOADER= 000043	L152 021452	L234 026340

L235	026412	L317	043004	L400	054426	L53	013442	MSG007	070775
L236	026420	L32	011754	L401	054440	L54	013416	MSG008	071007
L237	026424	L320	043052	L402	054456	L55	013372	MSG009	071021
L24	011626	L321	043052	L403	054460	L56	013364	MSG010	071033
L240	030242	L322	043102	L404	054500	L57	013370	MSG011	071045
L241	030240	L323	043222	L405	054512	L6	005100	MSG012	071133
L242	030242	L324	044402	L406	054530	L60	013414	MSG013	071230
L243	031670	L325	044626	L407	054532	L61	013410	MSG014	071232
L244	034432	L326	045010	L41	012354	L62	013414	MSG015	071234
L245	034442	L327	045114	L410	054546	L63	013442	MSG016	071236
L246	034554	L33	012072	L411	054750	L64	013442	MSG017	071250
L247	034504	L330	045120	L412	054756	L65	013442	MSG018	071261
L25	011642	L331	045126	L413	055004	L66	013536	MSG019	071264
L250	034516	L332	045142	L414	055022	L67	013606	MSG020	071270
L251	034534	L333	045154	L415	055034	L7	005100	MSG021	071311
L252	034542	L334	045366	L416	055046	L70	013760	MSG022	072101
L253	034566	L335	045376	L417	055060	L71	014150	MSG023	072123
L254	034724	L336	045532	L42	012354	L72	014116	MSG025	072137
L255	034624	L337	045572	L420	055102	L73	014034	MSG026	072163
L256	034636	L34	012024	L421	055150	L74	014036	MSG027	072175
L257	034670	L340	045576	L422	055230	L75	014076	MSG028	072212
L26	012072	L341	045626	L423	055244	L76	014112	MSG029	072226
L260	034654	L342	045640	L424	055246	L77	014146	MSG030	072246
L261	034666	L343	047026	L425	055352	MAINT =	177750	MSG031	072265
L262	034712	L344	047066	L426	055414	MA:HO =	170202	MSG032	072325
L263	034700	L345	047124	L427	055424	MAPLO =	170200	MSG033	072344
L264	034712	L346	047272	L43	012354	MAPL1 =	170204	MSG035	072472
L265	035024	L347	047306	L430	055726	MAPPER	042030	MSG036	072475
L266	036702	L35	012072	L431	055726	MBERR	013074	MSG037	072514
L267	037330	L350	047320	L432	056070	MEMDON	014216	MSG038	072533
L27	012072	L351	047616	L433	056030	MFPT =	000007	MSG039	072551
L270	037516	L352	047620	L434	056050	MJPAT	020002	MSG040	072573
L271	037620	L353	047732	L435	056070	MJTEST	017676	MSG041	072627
L272	040532	L354	050230	L436	056266	MKCONT	016244	MSG042	072654
L273	040540	L355	050214	L437	056136	MKCSRT	017366	MSG043	072675
L274	040776	L356	050164	L44	012360	MKFLAG	002116	MSG046	072722
L275	040776	L357	050314	L440	056264	MKLOOP	016426	MSG047	072755
L276	041040	L36	012144	L441	056214	MKPAT	017616	MSG048	072774
L277	041046	L360	050316	L442	056226	MKTEST	017456	MSG049	073034
L3	004724	L361	050430	L443	056264	MMRO =	177572	MSG050	073066
L30	012072	L362	050720	L444	056274	MMR1 =	177574	MSG051	073174
L300	041142	L363	050722	L445	056574	MMR2 =	177576	MSG052	073214
L301	041142	L364	051166	L446	057412	MMR3 =	172516	MSG053	073245
L302	041204	L365	051202	L447	057616	MMTRAP	040102	MSG054	073263
L303	041212	L366	051206	L45	012636	MMVEC =	000250	MSG055	073333
L304	041334	L367	051224	L450	057622	MSEEDH	002546	MSG056	073354
L305	042502	L37	012150	L451	057702	MSEEDL	002550	MSG058	073407
L306	042516	L370	051352	L452	057706	MSGA12	074767	MSG061	073431
L307	042530	L371	051354	L453	061022	MSGA34	072363	MSG062	073440
L31	011750	L372	051420	L454	061112	MSGB34	072427	MSG063	073460
L310	042530	L373	051644	L455	061114	MSG000	070307	MSG064	073471
L311	042632	L374	051752	L46	012636	MSG001	070354	MSG065	073501
L312	042662	L375	052256	L47	012612	MSG002	070436	MSG066	073513
L313	043004	L376	053574	L5	005020	MSG003	070513	MSG067	073576
L314	043004	L377	053576	L50	012564	MSG004	070620	MSG070	073605
L315	043004	L4	004742	L51	012574	MSG005	070726	MSG071	073636
L316	042764	L40	012152	L52	012614	MSG006	070740	MSG072	073654

MSG073	073672	MTPB21	034262	MT0017	021674	PCONFS	037070	QVFLAG	002316
MSG075	073710	MTPB24	035116	MT0020	021716	PCONF1	037000	RANODD	035646
MSG076	073742	MTPB25	035510	MT0021	023006	PCONF2	037036	RDCHR =	104411
MSG077	073763	MTPB26	035632	MT0022	023260	PDP110	040114	RDDEC =	104414
MSG079	073777	MTPC03	027204	MT0023	023312	PD1	051644	RDLIN =	104412
MSG085	074023	MTPC20	034126	MT0024	023356	PERA05	054012	RDOCT =	104413
MSG088	074050	MTPC21	034316	MT0025	023622	PERBNK	054644	READCS =	104426
MSG089	074066	MTPC24	035132	MT0026	023670	PERECC	054724	READON	002360
MSG090	074110	MTPC25	035550	MT0027	024172	PERRAB	054462	REALPA	002260
MSG091	074124	MTPC26	035666	MT0030	024656	PERRAW	054410	REFRES	034756
MSG092	074136	MTPD03	027222	MT0031	025160	PERRA3	051212	REFSUB	035026
MSG093	074152	MTPD20	034156	MT0032	025350	PERRA7	054534	REGCOP	036330
MSG095	074160	MTPD21	034352	MT0033	025702	PERR01 =	104427	RELENT	043112
MSG101	074167	MTPD25	035414	MT0034	026070	PERR02 =	104430	RELOCA	042466
MSG102	074217	MTPD26	035706	MT0035	026242	PERR03 =	104431	RELOC1	043126
MSG103	074246	MTPE20	034206	MT020Y	022554	PERR04 =	104432	RESREG =	104416
MSG104	074270	MTPE25	035436	MT020Z	022370	PERR05	054006	RESTAR	002566
MSG105	074272	MTP000	026774	MT0999	026354	PERR06	054034	RESVEC =	000010
MSG106	074376	MTP001	027020	MT1	016224	PERR07 =	104433	RES0	045640
MSG107	074414	MTP002	027052	MT2	016230	PERR10 =	104434	RES1	045720
MSG110	074471	MTP005	027316	MUT	002106	PERR11 =	104435	RES2	046066
MSG111	074535	MTP006	027352	NC	053576	PERR12 =	104436	RLFLAG	002124
MSG112	074567	MTP007	027552	NEMCNT	002066	PERR13 =	104437	RRFLAG	002122
MSG113	074604	MTP010	027652	NEWBAN	002270	PERR14 =	104440	RTNVAL =	X000000
MSG114	074621	MTP011	027760	NEWKER	044140	PERR15 =	104441	SAVREG =	104415
MSG116	074645	MTP012	030556	NEWLOA	044206	PERR16 =	104442	SBEMSK	002244
MSG117	074657	MTP013	031144	NOCH	060416	PERR17 =	104443	SBENT	017330
MSG118	074671	MTP014	031660	NOERRO	002400	PERR20 =	104444	SBETES	017052
MSG119	074703	MTP015	032442	NOFSMO	002376	PERR21 =	104445	SCOPE =	000004
MSG120	074712	MTP016	033206	NONEM	002076	PERR22 =	104446	SDPAR0 =	172260
MSG121	074733	MTP017	033770	NONEXI	040024	PERR23 =	104447	SDPAR5 =	172272
MSG122	074753	MTP020	034046	NOOJ	036702	PERR24 =	104450	SDPAR6 =	172274
MSG123	075021	MTP022	034402	NOPAR	062074	PERR25 =	104451	SDPAR7 =	172276
MSG124	075072	MTP025	035150	NOSCOP	002410	PERR26 =	104452	SEEDHI	002542
MSG125	075135	MTP030	035724	NOSUPE	002426	PERR27 =	104453	SEEDLO	002544
MSG126	075157	MTP031	035734	NOTAB	002342	PERR30 =	104454	SELONL	002000
MSG127	075224	MTP032	036012	NOTRCE	055366	PERR31 =	104455	SETPAT	044742
MSG128	075243	MTF033	036044	NO22BI	002424	PERR32 =	104456	SHADL1	011546
MSIZE	002352	MTP034	036142	NULLFL	002314	PERR33 =	104457	SHUTUP	045126
MTA030	024670	MTP035	036166	OLDCAC	002262	PERR34 =	104460	SIPAR0 =	172240
MTB020 =	000017	MTST3	011516	OLDCSR	002150	PERR35 =	104461	SIPAR3 =	172246
MTEST	016150	MTV020	022366	ONES	002554	PERR36 =	104462	SIPAR5 =	172252
MTLA11	030006	MT0000	020062	PADDRE	002034	PERR37 =	104463	SIPAR6 =	172254
MTLB11	030020	MT0001	020142	PAFBAF	015344	PERR40 =	104464	SIPDR0 =	172200
MTLC11	030032	MT0002	020262	PAFBAW	015474	PERR41 =	104465	SIZE =	040000
MTLD11	030126	MT0003	020422	PARBAF	015646	PERR42 =	104466	SKIPKA	002006
MTLO20	021772	MT0004	020654	PARBAW	015776	PERR43 =	104467	SKIPMK	002312
MTPA03	027104	MT0005	020776	PARCNT	002070	PERXOR	054620	SKPERR	002064
MTPA04	027242	MT0006	021132	PARITY	037720	PFLAG	002120	SKUB	043102
MTPA20	034046	MT0007	021166	PARTHE	002264	PGMCSR	002502	SKUJ	013100
MTPA21	034232	MT0010	021230	PARVEC =	000114	PHEBE	013076	SOBK	002532
MTPA24	035056	MT0011	021264	PASFLG	002256	PHYADD	002036	SOBLEN =	000056
MTPA25	035466	MT0012	021332	PATERR	002072	PROTYP	003710	SOFTPA	002560
MTPA26	035616	MT0013	021426	PATPLU	004514	PSIZE	002354	SOURCE	002272
MTPB03	027144	MT0014	021502	PATTER	002110	PSW =	177776	SPLTCS	002232
MTPB04	027276	MT0015	021560	PCBUMP	002276	PWRVEC =	000024	SSP =	X000006
MTPB20	034076	MT0016	021626	PCONFI	036570	QUICK	002406	ST =	177776

STACK =	002000	TAG75\$	056744	UNRELO	043356	\$DDW3	062560	\$MADR2	062524
START	003630	TAG76\$	056756	UPPFLG	002257	\$DDW4	062562	\$MADR3	062530
START1	000300	TAG77\$	057022	USERMA	044056	\$DDW5	062564	\$MADR4	062534
START2	000310	TAG78\$	057030	USESTK=	000700	\$DEENE	040176	\$MAIL	062466
START3	000200	TAG79\$	057110	USP =	%000006	\$DEVCT	062476	\$MAMS1	062516
STAR27	024252	TAG9\$	010764	WARN1	011052	\$DEVN	062544	\$MAMS2	062522
STOPOK	002370	TBG4\$	026676	WARN2	027116	\$DIDDO=	000000	\$MAMS3	062526
STRIPE	002336	TCFIG1	037214	WARN3	027132	\$DOAGA	014544	\$MAMS4	062532
SUBAAA	004552	TCFIG2	037346	WARN4	027156	\$DOAGN	014440	\$MBADR	062570
SUBAAB	004702	TCFIG3	037534	WARN5	027172	\$DOWN	052074	\$MNEW	061606
SUBAAI	011542	TCONF1	037072	WARN6	036552	\$DTBL	060354	\$MSGAD	062502
SUBAAP	013260	TEMP	002404	WARN6A	036512	\$ECCDI	040570	\$MSGLG	062504
SUBAAR	012422	TEST	006010	WARN6B	036544	\$ECCIN	040616	\$MSGTY	062466
SUBAAS	010474	TESTAD	002362	WARN7	024230	\$ECC1D	040604	\$MSWR	061575
SUCCE\$	002304	TESTMO	002522	WASDBE=	104500	\$ECC1I	040632	\$MTYP1	062517
SUPDOA	002254	TIME	002310	WASSBE=	104476	\$ENASB	040644	\$MTYP2	062523
SUPDO1	026424	TIMEOU	040070	WAS1DB=	104501	\$ENATS	040660	\$MTYP3	062527
SUPDO2	026440	TKVEC =	000060	WAS1SB=	104477	\$ENDAD	014430	\$MTYP4	062533
SUPDO3	026602	TMFLAG	002132	WHICHC	051042	\$ENERG	040166	\$NOTRA	062636
SUPDO4	026616	TOOMAN	002356	WOOPEN	053052	\$ENV	062506	\$NULL	002326
SUPDRO	002152	TOTCSR	002216	WOOPS	052504	\$ENVM	062507	\$NWTST=	000001
SUPDR1	002154	TRACE	006104	WOOPSA	053102	\$EOP	014274	\$OCNT	060144
SUPDR2	002156	TRAPVE=	000034	WOOPUP	052670	\$ERFLG	002012	\$OCTVL	062450
SUPDR3	002160	TSTBAN	011404	WORST	002540	\$ERRGE	041412	\$OCT8 =	062454
SUPDR4	002162	TSTDAT	002240	XOCHAR	053444	\$ERROR	055642	\$OMODE	060146
SUPDR5	002164	TSTRD1	040542	XXDPCH	002324	\$ERRTB	063110	\$OVER	055556
SUPDR6	002166	TSTREA=	104510	ZEROS	002306	\$ERTY	056306	\$OS =	000000
SUPLIM	053654	TST1	005402	\$APTHD	062566	\$ERTTL	002570	\$PASS	062474
SUPSTK=	000740	TST2	010500	\$AUTO	002060	\$ESCAP	002332	\$PASTM	062574
SWAPAT	002574	TST3	010664	\$BANK	002011	\$ETABL	062506	\$PATMA	002010
SWR	002576	TST4	011652	\$BASE	062542	\$ETEND	062566	\$PER01	053654
SWREG =	000176	TST5	014150	\$BELL	002613	\$EXHAL	045120	\$PER02	053702
SW0 =	000001	TST6	014222	\$CACHF	040232	\$ES =	000001	\$PER03	053730
SW1 =	000002	TYPDS =	104405	\$CACHN	040206	\$FATAL	062470	\$PER04	053760
SW10 =	002000	TYPEIT=	104401	\$CBCSR	040672	\$FILLC	002612	\$PER07	054042
SW11 =	004000	TYPOC =	104402	\$CB1CS	040714	\$FILLS	002327	\$PER10	054064
SW12 =	010000	TYPOS =	104403	\$CDW1	062546	\$FS =	000000	\$PER11	054114
SW13 =	020000	TYP\$0 =	000000	\$CDW2	062550	\$GTSWR	060550	\$PER12	054134
SW14 =	040000	TYP\$1 =	000002	\$CHARC	053632	\$HALT	056110	\$PER13	054156
SW15 =	100000	TYP\$2 =	000000	\$CHKDI	041266	\$HALT2	062642	\$PER14	054176
SW2 =	000004	TYP\$3 =	000000	\$CHK1D	041302	\$HIBTS	062566	\$PER15	054220
SW3 =	000010	TYP\$4 =	000000	\$CKSWR	060374	\$HIOCT	061766	\$PER16	054242
SW4 =	000020	TYP\$5 =	000000	\$CLRCS	041244	\$ILLUP	052476	\$PER17	054262
SW5 =	000040	TYP\$6 =	000002	\$CLR1C	041256	\$INVAL	041362	\$PER20	054300
SW6 =	000100	T12A	033206	\$CMTAG	002000	\$ITEMB	002013	\$PER21	054316
SW7 =	000200	T12B	033230	\$CMTGE	002514	\$IS =	000001	\$PER22	054336
SW8 =	000400	UDPAR0=	177660	\$CNTLC	061556	\$KERNE	040156	\$PER23	054354
SW9 =	001000	UDPAR7=	177676	\$CNTLG	061570	\$KMAP	042374	\$PER24	054372
SYSSIZ	003712	UIPAR0=	177640	\$CNTLK	060774	\$KS =	000061	\$PER25	051130
TAG2\$	011136	UIPAR1=	177642	\$CNTLU	061563	\$L =	000066	\$PER26	051562
TAG3\$	011172	UIPAR2=	177644	\$CPUOP	062514	\$LF	002621	\$PER27	054602
TAG4\$	026520	UIPAR3	177646	\$CRLF	002620	\$LL =	000064	\$PER30	051356
TAG70\$	056602	UIPAR4=	177650	\$DBLK	060364	\$LOADC	040250	\$PER31	054772
TAG71\$	056612	UIPAR5=	177652	\$DB20	062344	\$LPADR	062562	\$PER32	055070
TAG72\$	056622	UIPAR6=	177654	\$DDW0	062552	\$LPERR	002564	\$PER33	055136
TAG73\$	056672	UIPDRO=	177600	\$DDW1	062554	\$_S =	000000	\$PER34	055216
TAG74\$	056732	UNITOP	002366	\$DDW2	062556	\$MADR1	062520	\$PER35	055250

\$PER36	055304	\$SAVRE	062156	\$TN	= 000007	\$TYPEC	053446	\$WASDB	041100
\$PWRDN	051524	\$SAVR6	052502	\$TPB	002610	\$TYPEX	053634	\$WASSB	040734
\$PWRUP	052100	\$SCOPE	055334	\$TPFLG	002330	\$TYPOC	057746	\$WAS1D	041214
\$QUES	002617	\$STN	= 000001	\$TPS	002606	\$TYPON	057762	\$WAS1S	041050
\$R	= 177777	\$SVLAD	055542	\$TRAP	062602	\$TYPOS	057722	\$XTSTR	055434
\$RAND	062252	\$SV\$	= 000000	\$TRAP2	062624	\$T1	= 000000	\$Y\$	= 000000
\$RDCHR	061114	\$SWR	= 163000	\$TRPAD	062644	\$T2	= 000455	\$ZAP42	014410
\$RDDEC	061770	\$SWREG	062510	\$TSM	062572	\$UNIT	062500	\$Z\$	= 000000
\$RDLIN	061234	\$T	= 000456	\$TSTRD	040364	\$UNITM	062576	\$S\$	= 000000
\$RDOCT	061620	\$TESTN	062472	\$TTYIN	061532	\$USWR	062512	\$ST	= 000441
\$READC	040344	\$TKB	002604	\$TYPDS	060150	\$VECT1	062536	\$STT	= 000447
\$RESRE	062214	\$TKS	002602	\$TYPE	053320	\$VECT2	062540	\$UFILL	060145

. ABS. 075264 000  
000000 001  
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 26074 WORDS ( 102 PAGES)  
DYNAMIC MEMORY: 20346 WORDS ( 78 PAGES)  
ELAPSED TIME: 01:11:02  
MC8E,MC8E/-SP=CZMSDB.SML,MC8E.P11