

MK,MJ,MKA11

11/70 MAIN MEMORY
CEMKABO

AH-C645B-MC

JAN 1980

COPYRIGHT 78-80

digital

FICHE 1 OF 6

MADE IN USA



MK,MJ,MKA11

11/70 MAIN MEMORY
CEMKAB0

AH-C645B-MC

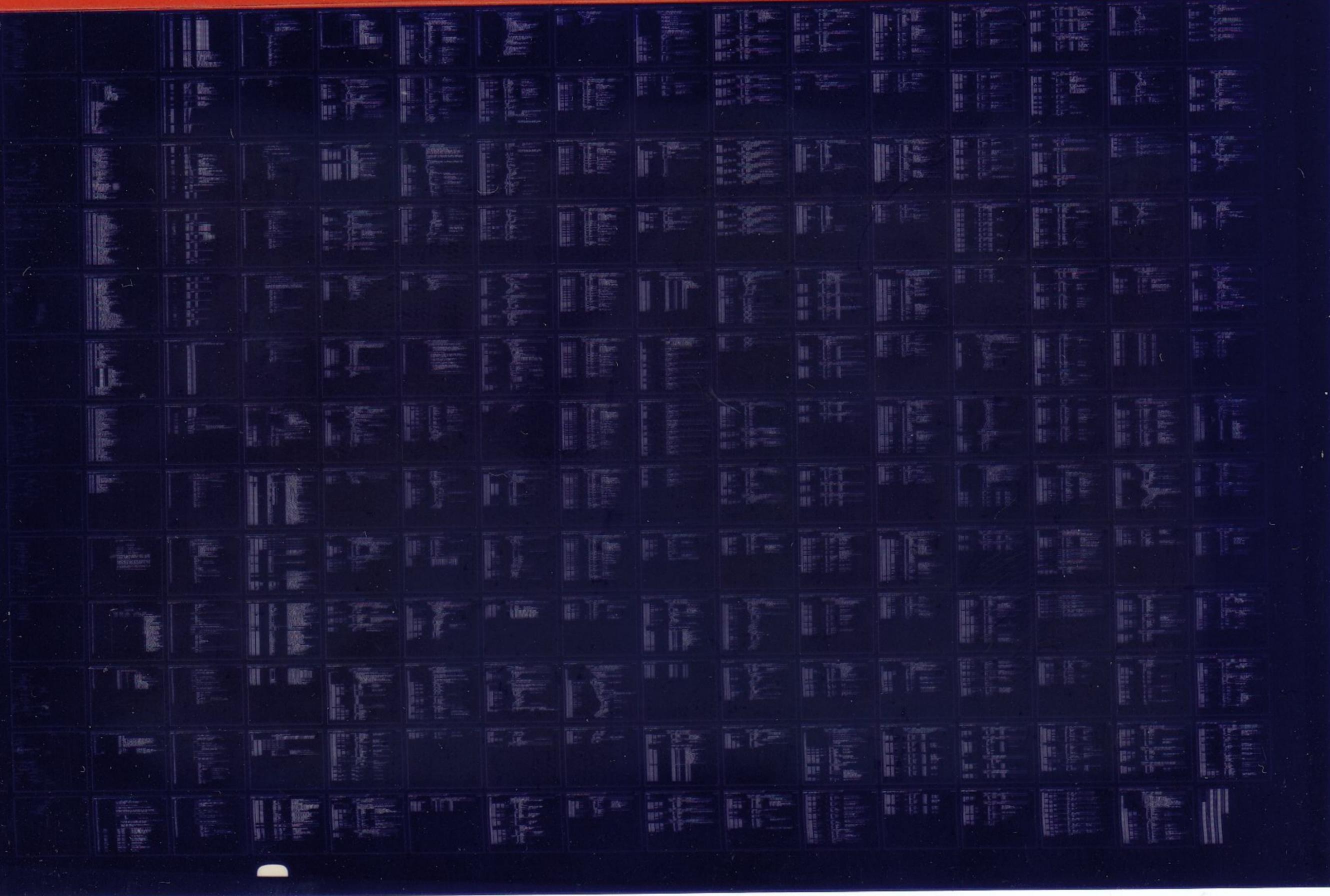
COPYRIGHT 78-80

FICHE 2 OF 6

JAN 1980

digital

MADE IN USA



MK,MJ,MKA11

11/70 MAIN MEMORY
CEMKAB0

AH-C645B-MC

COPYRIGHT 78-80
FICHE 3 OF 6

JAN 1980

digital

MADE IN USA

This image shows a microfiche card with a grid of 120 frames. Each frame contains a small, high-contrast image of a document page, likely containing technical or administrative information. The frames are arranged in a regular grid pattern across the entire surface of the card.

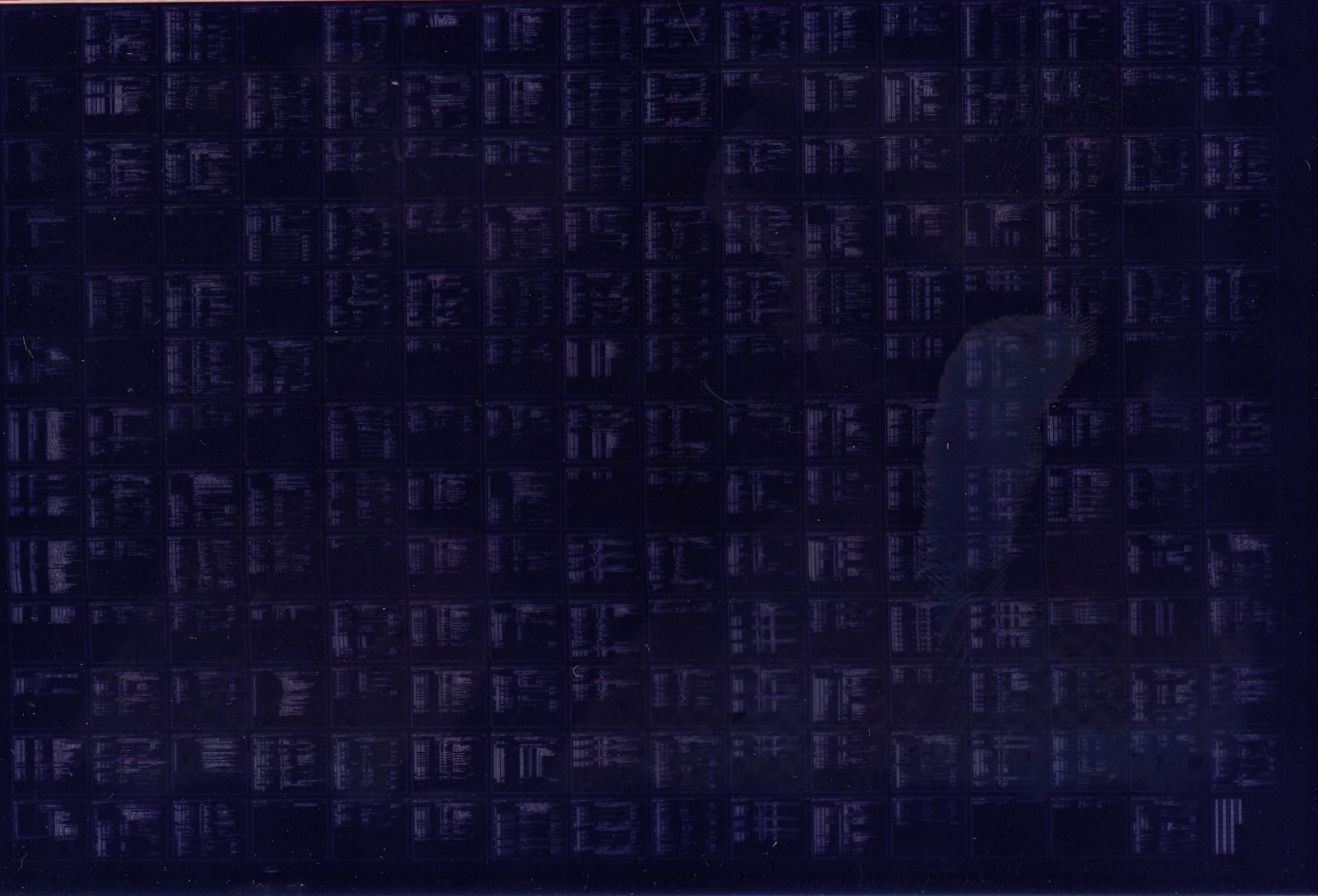
MK,MJ,MKA11

11/70 MAIN MEMORY
CEMKAB0

AH-C645B-MC

COPYRIGHT '78-80
FICHE 4 OF 6

JAN 1980
digital
MADE IN USA



MK,MJ,MKA11

11/70 MAIN MEMORY
CEMKAB0

AH-C645B-MC

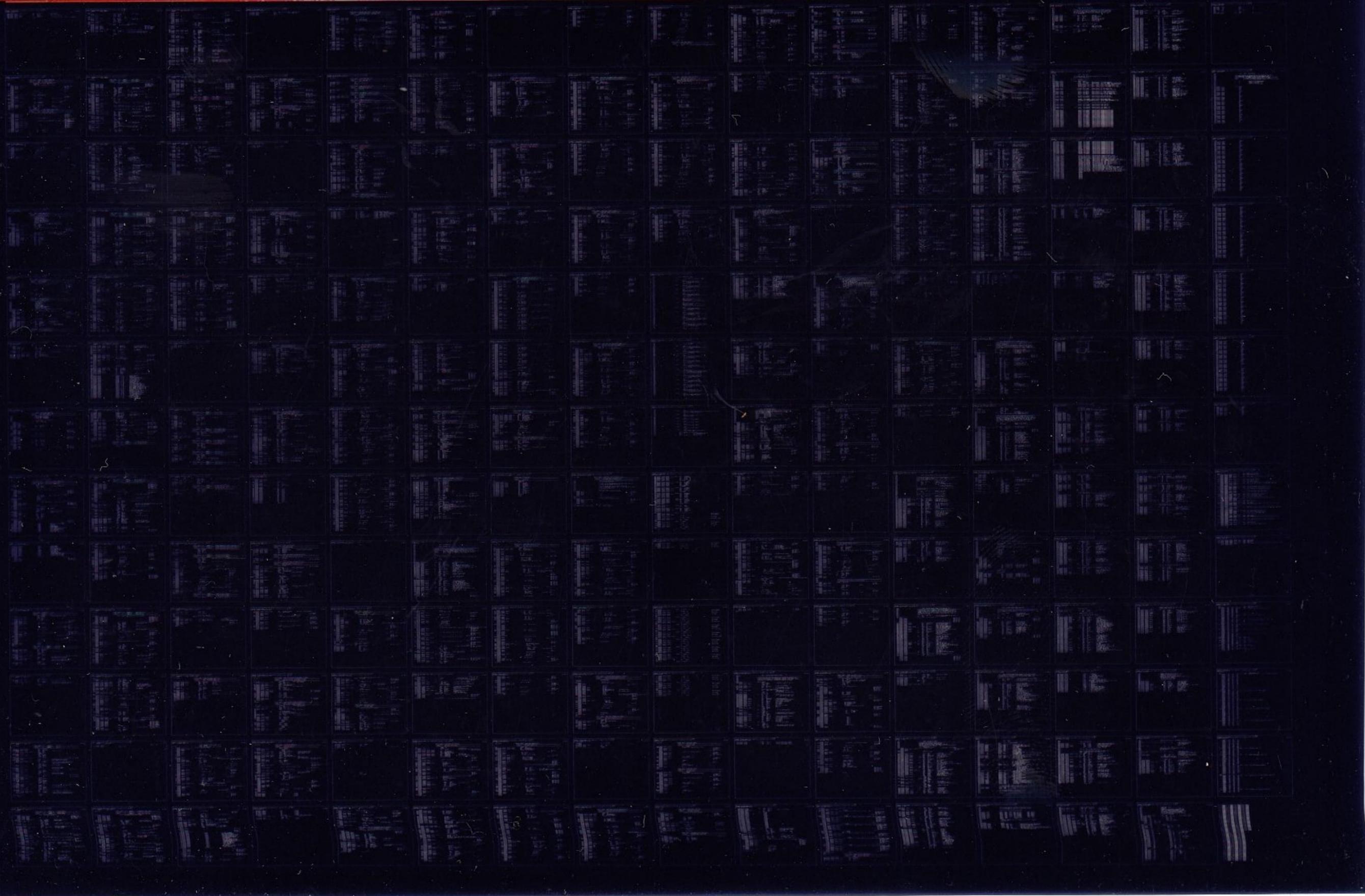
JAN 1980

COPYRIGHT 78-80

digital

FICHE 5 OF 6

MADE IN USA



MK,MJ,MKA11

11/70 MAIN MEMORY
CEMKAB0

AH-C645B-MC

COPYRIGHT '78-80
FICHE 6 OF 6

JAN 1980

digital

MADE IN USA

The microfiche card displays a grid of frames. The frames contain data organized into columns and rows. The data appears to be a list of items, possibly memory addresses or identifiers, with associated values or descriptions. The frames are arranged in a regular grid pattern, typical of microfiche storage.

IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-C644B-MC
PRODUCT NAME: CEMKABO 11/70 MAIN MEMORY DIAG
DATE CREATED: 31-JUL-79
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: KAY R. FISHER

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITH
OUT NOTICE
AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CO
R-
PORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR
ANY
ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUM
ENT IS FURNISHED UNDER A LICENSE
AND MAY ONLY BE USED OR COPIED IN ACCORDA
NCE 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 DIGI-
TAL.

COPYRIGHT (C) 1978, 1979 BY DIGITAL EQUIPMENT COR
PORATION

E 2

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 Starting Procedures
- 2.2 Special Environments
- 2.3 Program Options
- 2.4 Execution Times

3.0 ERROR INFORMATION

- 3.1 Error Reporting
- 3.2 Error Abbreviations
- 3.3 Error Halts

4.0 PROGRESS REPORTS

5.0 DEVICE INFORMATION TABLES (CSR'S)

- 5.1 Address(s) Of The CSR
- 5.2 Allocation Of CSR Bits
- 5.3 CSR Bit Summary

6.0 SUB-TEST SUMMARIES

- 6.1 Tests
- 6.2 Patterns
- 6.3 Multiport Tests

7.0 PROGRAM FEATURES

- 7.1 Fast Data Access Rates
- 7.2 Bank Zero Testing
- 7.3 Differences In Core Testing With 0-2 MEG (DEMJA)
- 7.4 Memory Configuration Map
- 7.5 Everything You've Always Wanted To Know About SUPERMAC ...

- 7.6 Ban
- ks Boards (MK11)
- 7.7 Memory Management Mapping
- 7.8 Listing Format

8.0
MULTIPOINT SYSTEM ORGANIZATION

- 8.1 Starting Slaves
- 8.2 Multipoint Testing

9.0 FLOW CHARTS

PAGE 3

1.0 GENERAL PROGRAM INFORMATION

1.1 Program Purpose (Abstract)

- a. Intended for use on all PDP-11/70's and PDP-11/74's including multi-processor multiported MKA11 configurations.
- 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 box and array.
- 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 (MJ11 MK11 & MKA11) on a variable number of CPU's (1-4).
- e. This program will functionally test MK11's, MKA11's, & MJ11's.
- f. It has special a maintenance mode (Field Service Mode) to provide specific functional capabilities.

1.2 System Requirements

1.2.1 Hardware Requirements -

1.2.1.1 Single Port Tests -

PDP-
11/70 or PDP-11/74 CPU and 64K (16 Bit Words) of Memory

SEQ 0005

1.2.1.2 Multiport Test

Each CPU must have a KW11L, a IIST, and a Multiprocessor Boot.

The use of the IIST and Multiprocessor Boot may be bypassed via start at 200.

Additionally each system will have the following minimum memory requirements.

<u># of CPU's</u>	<u>MKA11 Memory</u>
1	64K (16 Bit Words)
2	64K (16 Bit Words)
3	80K (16 Bit Words)
4	96K (16 Bit Words)

There must be no UNIBUS response to UNIBUS Address 752100 thru 752136.

If a special device or memory (other than MK11) resides there it must be removed in order to run this diagnostic. The lower 16K of physical memory must have no uncorrectable errors (DBE's).

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-11/70 Processor Handbook
2. PDP-11/70 Processor Handbook
3. MJ11 Maintenance Manual
4. MK11 Maintenance Manual
5. MKA11 Maintenance Manual
6. Programming Practices (175-003-009-02)
7. System Macro Manual (MAINDEC-11-DXQAC-C-D)
8. SUPER-MAC Reference Guide (130-380-007-00)
9. AP
111.MAN (APT/11-317-07-09)
10. ACT11.MAN (AUTOCAT-11-QZAUB-B-D)

1.4 Diagnostic Hierarchy Prerequisites

If the program hangs, dies, goes out to lunch, heads west, hiccups, vomits, eats itself, stutters, bombs, or in any way misbehaves, then:

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

1.5 Assumptions

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

2.0 OPERATING INSTRUCTIONS

2.1 Loading & Starting Procedures

2.1.1 Quick Starting -

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

2.1.2 Stopping -

1. Set SW8. or
2. Type control 'C' (Reference section 2.3.4.1).

2.1.3 Restarting (Preserve Configuration Table) -

1. Load address 202
2. Set s

witch register for options (Normally 0)

3. Start

2.1.4 Starting (With Multiprocessor and NO IIST or Boot) -

1. Load address 204
2. Set switch register for options (Normally 0)
3. Start

2.1.5 Switch Register Options -

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT RELOCATION
11	
10	QUICK VERIFY / INHIBIT MARGIN TESTING BELL ON ERROR
9	LOOP ON ERROR
8	HALT PROGRAM (UNRELOCATED & RESTORE LOADERS)
7	DETAILED ERROR REPORTS
6	PRINT CONFIGURATION
5	LIMIT MAX ERRORS PER BANK
4	FAT TERMINAL (1 32 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 Special Environments

2.2.1 XXDP -

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

2.2.2 ACT & APT Automatic Mode -

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

The first pass will be a quick verify pass

2.2.3 NO SBE Free Banks -

If the program cannot find any SBE (Single Bit Error) free locations (in non-protected MK11 memory) it will print out an error message and continue testing by-passing the CSR tests.

2.2.4 Mixed Core & MCS Configurations -

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

PAGE 9

2.3 Program Options

2.3.1 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 REL
LOCATION (10000)

This prevents the program from
moving and consequently
prevents the program from testing at l
east 16K of memo-
ry - in the case of MK11 memory up to 128K o
f memory
e may be bypassed because of PROTECTION POINTER coverag
(reference section 5.2.4).

SW11 = QUICK VERIFY/INHIB
IT MARGIN TESTING
(4000)

Each pass of the diagnostic normally runs all margins and checks all combinations of SBE's & DBE's. If this switch is selected no margins are run (shortening pattern tests by a factor of 5) and 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 = LO
OP ON ERROR
(1000)

This will cause looping from failure point back to the last correctly initialized area of the current test. This loop will frequently be too large for scoping purposes. Tighter loops are provided for in Field Service Mode reference section 2.3.4.5.10.

SW8 = HALT PROGRAM
(400)

- This initiates the following sequence:
1. clears margins in 'MAINT' REGISTER
 2. if program is relocated it moves back to bank

zero

D 2

SEQ 0016

3. flush out all possible DBE's.

4. turns off memory management

5. restore l

oaders

6. unmap the Unibus Map

7.

Halt if under APT or ACT branch sel.

PAGE 11

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, 'MEMERR', 'MAINT', 'CONTROL', 'LOADRS', 'HIADRS', 'CPUERR'

SW6 = PRINT CONFIGURATION MAP
(100)

This prints a map showing the memory configuration - reference section 7.4.

SW5 = LIMIT MAX ERRORS PER BANK
(40)

This will limit the number of error typeouts per bank.

The default is 10. DECIMAL, however this can be changed by changing location 'ERRMAX' manually or with reference Section 2.3.4.2).

SW4 = FAT TERMINAL
(20)

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

SW3-1 =
TEST MODE

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

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

NOTE: Margins follow patterns

Forward is (0,2,3,4,5)

Reverse is (5,4,3,2,0)

For more details reference section 2.3.1.1 and 2.3.1.2.

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 MK11 Single Bit errors to be reported by disabling error correction.

Error p

Printouts of SBE's are not distinguishable from
DBE's.

6 2

SEQ 0019

2.3.1.
1 Test Mode Example -

Example analysis of mode 5 'PAFBAW'. Assume Banks 0 & 1 are MJ11 (core) and Banks 2,3,4,& 5 are MK11 (MOS).

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

```
:TEST MK11 MEMORY TYPES FIRST
:TEST KNOWN BAD MEMORY (BANK 3)
```

```
PAT
TERN 17,      BANK 3,      MARGIN 0
PATTERN 17,   BANK 3,      MARGIN 2
PATTERN 17,   BANK 3,
MARGIN 3
PATTERN 7,    BANK 3,      MARGIN 0
PATTERN 7,    BANK 3,      MARGIN 2
PATTERN
7,          BANK 3,      MARGIN 3
PATTERN 1,    BANK 3,      MARGIN 0
PATTERN 1,    BANK 3,      MARGI
N 2
PATTERN 1,    BANK 3,      MARGIN 3
PATTERN 2,    BANK 3,      MARGIN 0
PATTERN 2,    B
ANK 3,      MARGIN 2
PATTERN 2,    BANK 3,      MARGIN 3
PATTERN 4,    BANK 3,      MARGIN 0
PATTERN 4,    BANK 3,      MARGIN 2
PATTERN 4,    BANK 3,      MARGIN 3
PATTERN 5,    BANK 3,
MARGIN 0
PATTERN 5,    BANK 3,      MARGIN 2
PATTERN 5,    BANK 3,      MARGIN 3
PATTER
RN 21,   BANK 3,      MARGIN 0
PATTERN 21,   BANK 3,      MARGIN 2
PATTERN 21,   BANK 3,      MA
RGIN 3
PATTERN 20,   BANK 3,      MARGIN 0
PATTERN 20,   BANK 3,      MARGIN 2
PATTERN 20,
BANK 3,      MARGIN 3
PATTERN 22,   BANK 3,      MARGIN 0
PATTERN 22,   BANK 3,      MARGIN
2
PATTERN 22,   BANK 3,      MARGIN 3
```

PATTERN 26,	BANK 3,	MARGIN 0
PATTERN 26,	BAN	
K 3,	MARGIN 2	
PATTERN 26,	BANK 3,	MARGIN 3

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

PATTERN 17,	BANK 2,	MARGIN 0
PATTERN 17,	BANK 2,	MARGIN 2
PAT		
TERN 17,	BANK 2,	MARGIN 3
PATTERN 17,	BANK 4,	MARGIN 0
PATTERN 17,	BANK 4,	
MARGIN 2		
PATTERN 17,	BANK 4,	MARGIN 3
PATTERN 17,	BANK 5,	MARGIN 0
PATTERN		
17,	BANK 5,	MARGIN 2

PATTERN 17,	BANK 5,	MARGIN 3
PATTERN 7,	BANK 2,	MARG
IN 0		
PATTERN 7,	BANK 2,	MARGIN 2
PATTERN 7,	BANK 2,	MARGIN 3
PATTERN 7,		
BANK 4,	MARGIN 0	
PATTERN 7,	BANK 4,	MARGIN 2
PATTERN 7,	BANK 4,	MARGIN 3
PATTERN 7,	BANK 5,	MARGIN 0
PATTERN 7,	BANK 5,	MARGIN 2
PATTERN 7,	BANK	
5,	MARGIN 3	
PATTERN 1,	BANK 2,	MARGIN 0
PATTERN 1,	BANK 2,	MARGIN 2
PATT		
ERN 1,	BANK 2,	MARGIN 3
PATTERN 1,	BANK 4,	MARGIN 0
PATTERN 1,	BANK 4,	M
ARGIN 2		
PATTERN 1,	BANK 4,	MARGIN 3
PATTERN 1,	BANK 5,	MARGIN 0
PATTERN		
1,	BANK 5,	MARGIN 2
PATTERN 1,	BANK 5,	MARGIN 3
PATTERN 2,	BANK 2,	MARGIN
0		
PATTERN 2,	BANK 2,	MARGIN 2
PATTERN 2,	BANK 2,	MARGIN 3
PATTERN 2,	BA	
NK 4,	MARGIN 0	
PATTERN 2,	BANK 4,	MARGIN 2
PATTERN 2,	BANK 4,	MARGIN 3
P		
ATERN 2,	BANK 5,	MARGIN 0
PATTERN 2,	BANK 5,	MARGIN 2
PATTERN 2,	BANK 5,	
	MARGIN 3	
PATTERN 4,	BANK 2,	MARGIN 0
PATTERN 4,	BANK 2,	MARGIN 2
PATTER		
N 4,	BANK 2,	MARGIN 3
PATTERN 4,	BANK 4,	MARGIN 0
PATTERN 4,	BANK 4,	MAR
GIN 2		
PATTERN 4,	BANK 4,	MARGIN 3
PATTERN 4,	BANK 5,	MARGIN 0
PATTERN 4,		
	BANK 5,	MARGIN 2
PATTERN 4,	BANK 5,	MARGIN 3
PATTERN 5,	BANK 2,	MARGIN 0
PATTERN 5,	BANK 2,	MARGIN 2

PATTERN 5.	BANK 2.	MARGIN 3
PATTERN 5.	BANK	
4.	MARGIN 0	
PATTERN 5.	BANK 4.	MARGIN 2
PATTERN 5.	BANK 4.	MARGIN 3
PAT		
TERN 5.	BANK 5.	MARGIN 0
PATTERN 5.	BANK 5.	MARGIN 2
PATTERN 5.	BANK 5.	
MARGIN 3		
PATTERN 21.	BANK 2.	MARGIN 0
PATTERN 21.	BANK 2.	MARGIN 2
PATTERN		
21.	BANK 2.	MARGIN 3
PATTERN 21.	BANK 4.	MARGIN 0
PATTERN 21.	BANK 4.	MARGIN
N 2		
PATTERN 21.	BANK 4.	MARGIN 3
PATTERN 21.	BANK 5.	MARGIN 0

PATTERN 21,			
BANK 5,	MARGIN 2		
PATTERN 21,	BANK 5,	MARGIN 3	
PATTERN 20,	BANK 2,	MARGIN 0	
PATTERN 20,	BANK 2,	MARGIN 2	
PATTERN 20,	BANK 2,	MARGIN 3	
PATTERN 20,	BANK		
4,	MARGIN 0		
PATTERN 20,	BANK 4,	MARGIN 2	
PATTERN 20,	BANK 4,	MARGIN 3	
PATT			
ERN 20,	BANK 5,	MARGIN 0	
PATTERN 20,	BANK 5,	MARGIN 2	
PATTERN 20,	BANK 5,	M	
ARGIN 3			
PATTERN 22,	BANK 2,	MARGIN 0	
PATTERN 22,	BANK 2,	MARGIN 2	
PATTERN 2,			
2,	BANK 2,	MARGIN 3	
PATTERN 22,	BANK 4,	MARGIN 0	
PATTERN 22,	BANK 4,	MARGIN	
2			
PATTERN 22,	BANK 4,	MARGIN 3	
PATTERN 22,	BANK 5,	MARGIN 0	
PATTERN 22,	BA		
NK 5,	MARGIN 2		
PATTERN 22,	BANK 5,	MARGIN 3	
PATTERN 26,	BANK 2,	MARGIN 0	
P			
ATTERN 26,	BANK 2,	MARGIN 2	
PATTERN 26,	BANK 2,	MARGIN 3	
PATTERN 26,	BANK 4,		
	MARGIN 0		
PATTERN 26,	BANK 4,	MARGIN 2	
PATTERN 26,	BANK 4,	MARGIN 3	
PATTER			
N 26,	BANK 5,	MARGIN 0	
PATTERN 26,	BANK 5,	MARGIN 2	
PATTERN 26,	BANK 5,	MAR	
GIN 3			

;RELOCATE TEST PROGRAM SPACE (BANK 0 & 1)

PATTERN 1,	BANK		
0,	MARGIN 0		
PATTERN 1,	BANK 0,	MARGIN 2	
PATTERN 1,	BANK 0,	MARGIN 3	
PATT			
ERN 1,	BANK 0,	MARGIN 4	
PATTERN 1,	BANK 0,	MARGIN 5	
PATTERN 2,	BANK 0,	M	

ARGIN 0			
PATTERN 2.	BANK 0.	MARGIN 2	
PATTERN 2.	BANK 0.	MARGIN 3	
PATTERN 2.	BANK 0.	MARGIN 4	
PATTERN 2.	BANK 0.	MARGIN 5	
PATTERN 3.	BANK 0.	MARGIN	
0			
PATTERN 3.	BANK 0.	MARGIN 2	
PATTERN 3.	BANK 0.	MARGIN 3	
PATTERN 3.	BA		
NK 0.	MARGIN 4		
PATTERN 3.	BANK 0.	MARGIN 5	
PATTERN 4.	BANK 0.	MARGIN 0	
P			
ATTERN 4.	BANK 0.	MARGIN 2	
PATTERN 4.	BANK 0.	MARGIN 3	
PATTERN 4.	BANK 0.		
	MARGIN 4		

GE 16

PATTERN 4,	BANK 0,	MARGIN 5
PATTERN 5,	BANK 0,	MARGIN 0
PATTE		
RN 5,	BANK 0,	MARGIN 2
PATTERN 5,	BANK 0,	MARGIN 3
PATTERN 5,	BANK 0,	MA
RGIN 4		
PATTERN 5,	BANK 0,	MARGIN 5
PATTERN 26,	BANK 0,	MARGIN 0
PATTERN 26		
	BANK 0,	MARGIN 2
PATTERN 26,	BANK 0,	MARGIN 3
PATTERN 26,	BANK 0,	MARGIN
4		
PATTERN 26,	BANK 0,	MARGIN 5
PATTERN 1,	BANK 1,	MARGIN 0
PATTERN 1,	BAN	
K 1,	MARGIN 2	
PATTERN 1,	BANK 1,	MARGIN 3
PATTERN 1,	BANK 1,	MARGIN 4
PA		
TTERN 1,	BANK 1,	MARGIN 5
PATTERN 2,	BANK 1,	MARGIN 0
PATTERN 2,	BANK 1,	
	MARGIN 2	
PATTERN 2,	BANK 1,	MARGIN 3
PATTERN 2,	BANK 1,	MARGIN 4
PATTERN 2,		
	BANK 1,	MARGIN 5
PATTERN 3,	BANK 1,	MARGIN 0
PATTERN 3,	BANK 1,	MAR
GIN 2		
PATTERN 3,	BANK 1,	MARGIN 3
PATTERN 3,	BANK 1,	MARGIN 4
PATTERN 3,		
	BANK 1,	MARGIN 5
PATTERN 4,	BANK 1,	MARGIN 0
PATTERN 4,	BANK 1,	MARGIN 2
PATTERN 4,		
	BANK 1,	MARGIN 3
PATTERN 4,	BANK 1,	MARGIN 4
PATTERN 4,	BANK	
1,	MARGIN 5	
PATTERN 5,	BANK 1,	MARGIN 0
PATTERN 5,	BANK 1,	MARGIN 2
PAT		
TERN 5,	BANK 1,	MARGIN 3
PATTERN 5,	BANK 1,	MARGIN 4
PATTERN 5,	BANK 1,	
MARGIN 5		
PATTERN 26,	BANK 1,	MARGIN 0
PATTERN 26,	BANK 1,	MARGIN 2
PATTERN 26,		
	BANK 1,	MARGIN 3

PATTERN 26,
PATTERN 26,
N 5

BANK 1,
BANK 1,

MARGIN 4
MARGI

NOTE

ple & not an actual se-
quence. This is an exam

The pattern sequence was forward (the simple patterns first, complex patterns last) sequence of patterns (MK11 = 17, 7, 1, 2, 4, 5, 21, 20, 22, 26) (MJ11 = 1, 2, 3, 4, 5, 26).

Since the patterns were forward so were the margins (0,2,3,4,5).

If the patterns were run in reverse (MK11=26,22,20,21,5,4,2,1,7,17) (MJ11=26,5,4,3,2,1,) then the margins would also have been in reverse (5,4,3,2,0)

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

1. MK11 banks that are not protected or program space (from 0 to 167).
2. MJ11 banks that are not program space (from 0 to 167).

3. The program now relocates & tests:

4. MK11 banks that were protected or program space (from 0 to 167).
5. MJ11 banks that were program space (from 0 to 167).

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

1. MK11 banks that are known bad and are not protected or program space (from 0 to 167).
2. MJ11 banks that are known bad and are not program space (from 0 to 167).
3. MK11 banks that are presumed good and are not protected or program space (from 0 to 167)

4. MJ11 banks that are presumed good and are not program space
(from 0 to 167).

5. The program now relocates & tests:

6. MK11 banks that are known bad and were protected or program
space
(from 0 to 167).

7. MJ11 banks that are known bad and were program spac
e (from 0
to 167).

8. MK11 banks that are presumed good and
were protected or pro-
gram space (from 0 to 167).

9. MJ11 b
anks that are presumed good and were program space
(from 0 to 167
).

2.3.1.2 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 167 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 167 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 167 except those requiring relocation*, then presumed good banks are tested from 0 to 167 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 167 except those requiring relocation*, then

presumed good banks are tested from 0 to 167 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 on to the more complex.

While testing each pattern the banks
are run from 0 to 167 except those requiring relocation*.

E 19

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 167 except those requiring relocation* is run, then presumed good banks are run from 0 to 167 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 167 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 167 except those that require relocation* is run, then presumed good banks are run from 0 to 167 except those requiring relocation*.

NOTE

* Relocation is required to test the bank(s) in program space and also to test any MK11 banks protected by diagnostic checkmode with the inhibit mode

pointer off (zero):

H 3

SEQ 0033

2.3.1.3 Test Mode Applications -

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

Advantages: Easy to understand.

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

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

Advantages: Seeks Bad Banks. Easy to understand.

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

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

Advantages: Covers all banks fast. Easy to understand.

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

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

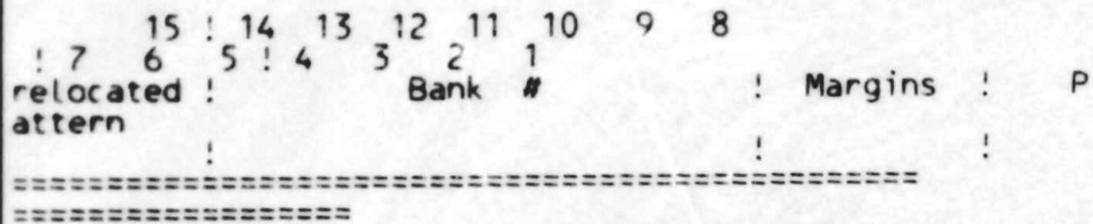
Advantages: Covers all Banks fast.

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

2.3.2 Display Register -

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

Display fields are as follows:



OR

In the case of multiprocessor tests the entire display register is used to represent the two's complement of the multiprocessor test #.

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 Manual by looking up the symbolic tags 'MTONN' and 'MTPNN' - where 'NN' is the Pattern number. MTONN refers to the routine that sets up for the test Pattern whereas MTPNN is the actual pattern itself.

NOTE

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

MARGINS = The 3 bit value loaded into the Maintenance register (17777750) Bits (3:1). Margins are decoded as follows:

Margin	MJ11(core)	MK11(MOS)
0	Normal Operation	Normal Operation
2	Early Strobe	Early MDR L

oad	3	Late Strobe	Late Refresh
	4	Low Current	Reserved
(Normal)	5	High Current	Reserved (Normal)

Margin 1 = Force
 es "wrong address parity" error

BANK = The number of the Bank (16K) of
 memory under test (0-167).
 these bits directly map to physical address bit

s (21:15). If the Bank # is above 167 octal this entire display is
 i
 n Multiprocessor mode - see above.

RELOCATED = This BIT indicates that the p
 rogram is relocated and no
 longer in Bank 0. It will be relocated to the
 first
 known good non-protected memory bank indicated on
 the configur
 ation map (reference section 7.4).

2.3.3 Special Memory Locations -

²2.3.3.1 CACHE Constant -

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

If you desire to run with group 0 disabled change this CACHE constant to 4 or 5, if you desire to run with group 1 disabled change the CACHE constant to 10 or 11, if you desire to disable all of CACHE change the CACHE constant to 14 or 15.

NOTE

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

2.3.3.2 Configuration Table

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

CONFIG: First 16K Configuration words (3 each)
 and 16K² Configuration words (3 each)

 167th 16K
 configuration words (3 each)

Configuration Words:
 LOW: BIT 0-2 INTERLEAVE FACTOR
 BIT 3 EXTERNAL INTERLEAVE
 BIT 4 MASTER CAN ACCESS (MEMORY EXISTS)
 BIT 5 SLAVE #1 CAN ACCESS (SHARED)
 BIT 6 SLAVE #2 CAN ACCESS (SHARED)
 BIT 7 SLAVE #3 CAN ACCESS (SHARED)
 BIT 8-10 BOX NUMBER (0-7)
 BIT 11 ERRORS PRESENT
 BIT 12 PROTECTED REGION OF A MK11 BOX
 BIT 13-14 MEMORY TYPE; 0=MJ11; 1=MK11
 BIT 15 PROTECTED (PROGRAM SPACE)
 MED: BIT 0-5 NOT USED
 BIT 6-8 MK11 CSR ADDRESS
 BIT 9-11 NOT USED
 BIT 12 INTERNAL INTERLEAVE
 BIT 13 'BACKGROUND PATTERN VALID' FLAG
 BIT 14 BANK SELECTED FOR TEST BY FIELD SERVICE MODE
 BIT 15 LOADERS HOME BANK
 HIGH: BIT 0 INDICATES BANK IS ADDRESSED BY CSR 172100
 BIT 1 INDICATES BANK IS ADDRESSED BY CSR 172104
 BIT 2 INDICATES BANK IS ADDRESSED BY CSR 172110
 BIT 3 INDICATES BANK IS ADDRESSED BY CSR 172114
 BIT 4 INDICATES BANK IS ADDRESSED BY CSR 172120
 BIT 5 INDICATES BANK IS ADDRESSED BY CSR 172124
 BIT 6 INDICATES BANK IS ADDRESSED BY CSR 172130
 BIT 7 INDICATES BANK IS ADDRESSED BY CSR 172134

CATES BANK IS ADDRESSED BY CSR 172134
BIT 8-15 NUMBER OF ERRORS

SEQ 0039

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

2.3.4 Terminal Commands -

2.3.4.1 Control 'C' -

This command will
:

1. Clear any margin conditions.
2. 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 5, attempt to Boot RK05 Drive 1.
7. Failing 6, go to 5.

This command will only be recognized at the completion of the current test or pattern.

2.3.4.2 Control 'D' (Debug) -

This command will enter a modified version of ODT.

This is a subset of the ODT as documented in the RT11 System Reference Manual with the addition of Control 'C', Control 'F', & control 'E' which function as outlined here.

NOTE

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

PAGE 25

2.3.4.3 Control 'E' (procEEd) -

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

2.3.4.4 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 MAR= 3 PAT= 26
```

2.3.4.5 Control 'S' (Stop) -

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

2.3.4.6 Control 'Q' (Quit) -

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.

2.3.4.7

Control 'F' (field service mode) -

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

When the program is looking for a sub command and you 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

default Typing just carriage return is a command 0.

2.3.4.7.1 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 0.

2.3.4.7.2 Field Service Command 1 (Read CSR) -

This command will typeout the contents of the MK11 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(0-7)' to which you must respond with an Octal number from 0 to 7. 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 refer
ences are done in accordance
with section 5.1.

2.3.4.7.

3 Field Service Command 2 (Load CSR) -

This command will enable you to load the MK11 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(0-7)' to which you must respond with an Octal number from 0 to 7. 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 'FIRST CSR WORD' to which you must respond with an Octal number. Note typing just carriage return is a default 0.

The program will then ask you for the '2ND CSR WORD' to which you must respond with an Octal number.

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

2.3.4.7.4

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 (0-16777776)' to which you must respond with an Octal number.

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

The contents of
your physical address will be typed.

2.3.4.7.5 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 (0-16777776)' to which you must respond with an Octal number.

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

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

2.3.4.7.6 Field Service Command 5 (Select Bank, Margin & Pattern) -

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

The program will ask you 'BANK(0-167)' to which you must respond with an Octal number. If the bank is not accessible. The program will type 'BANK NOT ACCESSIBLE' and ask question over.

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

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

A
PT 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.

The program will then ask 'MARGIN (0,2,3,4,5)?' to which you must respond with an Octal number of the set (0,2,3,4,5). If SW5 is set in the Switch register the program will type 'MARGINS INHIBITED BY SW5' and ask you for the margin again. You must then either turn off SW5 or select a margin of 0.

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 de-

pressed to escape this loop.

J 4

SEQ 0048

2.3.4.7.7 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 7.4.

2.3.4.7.8 Field Service Command 7 (Battery Backup Test) -

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

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

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

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

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

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

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

2.3.4.7.10 Field Service Command 9 (Super Tight scope loop) -

This command allows super tight scope loops on any physical address with any margin setting.

The program will ask for 'PHYSICAL ADDRESS' to which you must respond with an octal number.

The program will then ask 'MARGIN (0,2,3,4,5)?' to which you must respond with an Octal number of the set (0,2,3,4,5). If SW5 is set in the Switch register the program will type 'MARGIN INHIBITED BY SW5' and ask you for the margin again. You must then either turn off SW5 or select a margin of 0.

The program will then type 'SW R ---> MEM, MEM ---> DISPLAY' indicating that it will move the switch register into the physical address, move the physical address into the Display Register, and loop.

The the program arms the console keyboard for interrupts and types 'TO ESCAPE TYPE ANY KEY!'

The program now enters a loop at symbolic location 'CMD9LOOP' which consists of the following code:

```

CMD9LOOP:      MOV      @SWR,(
RO)           ;WRITE SWITCH REGISTER INTO MEMORY
              MOV      (RO),@DISPLAY ;READ MEMORY INTO DI
DISPLAY REGISTER
              BR      CMD9LOOP      ;LOOP TILL KEYBOARD INTERRUPT

```

Typing any key will exit this command.

PAGE 32

2.3.4.7.11 Field Service Command 10 (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.

2.3.4.7.12 Field Service Command 11 (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.

2.3.4.7.13 Field Service Command 12 (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, VT61's, etc.

2.3.4.7.14 Field Service Command 13 (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.

2.3.4.7.15 Field Service Command 14 (Exit Kamikaze Mode) -

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

2.3.4.7.16 Field Service Command 15 (Turn Cache Off) -

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

2.3.4.7.17 Field Service Command 16 (Turn Cache On) -

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

2.3.4.7.18 Field Service Command 17 (Run Only Multiport Tests) -

Single Port Testing will be disabled. This is used to extensively test only logic that is unique to the Multiprocessors.

2.3.4.7.19 Field Service Command 18 (Resume Running Both Single and Multiport Tests) -

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

2.3.4.7.20 Field Service Command 19 (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.

2.3.4.7.21 Field Service Command 2

0 (Resume Testing All Banks) -

Return to the default mode of testing (undo C
ommand 19).

AGE 34

2.4 Execution Times

2.4.1 Typical (System) -

Execution time depends on many variables however here are some measured times.

	CEMKAB(70 MEM)	DEMJA (OLD 0-2 MEG)
128K of MJ11 (core) Memory		
Normal Pass	2 Min 45 Sec	11 Min 41 Sec
Quick Verify	33 Sec	2 Min
1		
28K of MK11 (MOS) Memory		
Normal Pass	2 Min 52 Sec	11 Min 41 Sec
Quick Verify	45 Sec	2 Min
Kamikaze Mode	29 Min	
Kamikaze QV	9 Min 35 Sec	

2.4.2 Calculations (System) -

Normal Pass (all margins)

- Add 14 Sec per MK11 Box (CSR)
- Add 20 Sec per BANK of MOS
- Add 21 Sec per BANK of Core

Quick Verify Pass

- Add 5 Sec per BANK of MOS
- Add 4 Sec per BANK of Core

Kamikaze Mode

Multiply above times by 10.

2.4.3 Typical (Patterns) -

<u>Pattern</u>	<u>Time</u>	<u>Description</u>
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 ONES TEST
MT0005	: 1 SEC	ROTATING ZEROS 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 S
MT0013	: 1 SEC	CREATE DOUBLE BIT ERROR TEST
MT0014	: 1 SEC	WRITE INHIBIT
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
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	SKewed UP ADDRESS TEST
MT0033	:<1 SEC	WRITE RECOVERY TEST
MT0034	:35 SEC	BRANCH GOBBLE TEST
MT0035	:<1 SEC	SOFT ERROR TEST

3.0 ERROR INFORMATION

3.1 Error Reporting

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

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

Example:

PC	BANK	VADD	PADD	GOOD	B		
AD	XOR	MAR	BOX	MTYPE	INT	PAT	ARRAY
022132	37	060006	03700006	000000	0		
00100	000100	0	0	MK11	0	06	15
022132	37	060006	03700006	000000	0		
00100	000100	0	0	MK11	0	06	15
022132	37	060006	03700006	000000	0		
00100	000100	0	0	MK11	0	06	15
022132	37	060006	03700006	000000	0		
00100	000100	0	0	MK11	0	06	15

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). The margins were at 0 (no margin) and the memory resides in Box 0, it is an MK11 (MOS) Memory and its interleave factor is 0 (not interleaved). The CSR identified an error in slot A1 (reference Section 7.6).

NOTE

Subsequent errors of the same test do
not type a ne

w heading.

The Box Interleave and slot would be a "?" if the memory type wa
s MJ11
(Core).

If an MJ11 (Core) gives Parity Aborts then the Bad Data & XOR
will be
"?????".

If no MK11 has no CSR with a SBE or DBE then slot will b
e a "?".

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 MK11 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.
BDSWITCH	Bad Switch setting in binary.
BOX	The number
of the Box in the system. MJ11 boxes have no numbers.	
CHKBITS	The 7 bit value
of the Check Code Bits.	
CONTRL	The CACHE Control register.
CPUERR	CPU Error
register.	
CSR	Control and Status Register.
CSR+2	The 2nd half the the MK11
double word CSR.	
CSRNUM	CSR NUMBER (0-7).
DBE	Double Bit Error (uncorrectable error).
DEV ADD	Device Address.
ECC	Error Correction.
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.
GDSWITCH	Good Switch setting in binary.
GOOD	Good data.
HIADRS	High Address error register.
IIST	Inter-processor Interrupt and Sanity Timer.
INT	Interleave total including internal and external.
INTRLV	Interleave total including internal and external.

LOADRS Low Address error register.
 MAINT Maintenance register.
 MAR M
 Margin. Bits 3-1 of the PDP-11/70 Maintenance Register.
 MASTER The Master CPU.

MEMERR Memory Error register.
 MJSIZE MJ11 Size.
 MKSIZE MK11 Size.

MMRO Memory Management Register #0.
 MMR1 Memory Management Register #1.
 MMR2 Memo
 ry Management Register #2.
 MMR3 Memory Management Register #3.
 MTYPE Memory
 Type (MJ11 or MK11).
 PADD Physical Address (asserted by the program after mapp
 ing).
 PAT Pattern number.
 PC Program Counter at the time the error occurred.

REAL# The # of CPU's actually on the system.
 SBE Single Bit Error (correcta
 ble error).
 SLAVE# The Slave CPU Number.
 SLAVE1 The 1st Slave CPU.
 SLAVE2
 The 2nd Slave CPU.
 SLAVE3 The 3rd Slave CPU.
 VADD Virtual Address (asserted
 by the program before mapping).

PAGE 38

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'.

Slave CPU's that did not get permission from the master CPU to proceed will halt at symbolic location '\$HALT'.

If any CPU detects that multiple CPU's are assuming master status then that CPU will halt itself at symbolic location 'MHALT1', 'MHALT2', or 'MHALT3'.

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	#	1
END PASS	#	2
END PASS	#QV	3

NOTE

SEQ 0063

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

5.0 DEVICE INFORMATION TABLES
(CSR's)

1st Word

```

15 ! 14 13 12 11 10 9 8 ! 7 6 5
! 4 ! 3 ! 2 ! 1 ! 0
DBE ! Check/Syndrome/Capacity ! Bad array
! SBE !Protect!DIAG ! ECC ! ENA
! ! ! ! !
! !Pointer!CHECK! DIS ! PAR
! ! ! ! !
! ! ! ! ! TRAP

```

2nd Word

```

15 ! 1
4 13 12 11 ! 10 ! 9 ! 8 7 6 5 4 3 2 1 0
DBE ! Interl
eave !Interleave or ! Bad ! Starting Address
! !S
tarting Add !control!
! ! Load ! !

```

5.1
Address(s) Of The CSR

The CSR is accessed as a 32-bit word on the 11/70 main memory bus.
(To the processor, the CSR looks like two consecutive 16 bit words.)

A switch on the M8158 address buffer module selects one of two options for CSR address access, either Direct or Indirect.

5.1.1 Direct Address

(For this option the 11/70 processor requires modification.) Switches on the M8158 address buffer module select one of the following eight CSR (double word) locations (octal addresses):

- 177772100
- 177772104
- 177772110
- 177772114
- 177772120
- 177772124
- 177772130
- 177772134

5.1.2 Indirect Address

(No 11/70 modifications required for this option but its use is quite complex.) Switches on the M8158 address buffer module select one of the following eight CSR (double word) locations (octal addresses):

- 17772100
- 17772104

- 17772110
- 17772114
- 17772120
- 17772124
- 17772130
- 17772134

NOTE

K
These addresses are in the upper 128
word section of the existing 11/70 2M
address space. Normally this is
word reserved address space never
implemented in memory and not directly accessible by
the processor.

PAGE 42

However, these CSR locations may be accessed for diagnostic purposes through the unibus map with cache turned off. It is not recommended to use this method in an operating system environment since the unibus map is required if other NPR activity is present. One cookbook method of implementing a CSR access (within this diagnostic program) is:

1. Halt all NPR activity on the system.
2. Turn cache off.
3. Par 6 <---
177400 (other Par's Normal)
4. Enable memory management in 22 bit mode
5. Map 36 <--- 17760000 (other Maps = 0).
6. Turn on Unibus Map.
7. Access virtual 152100 through 152136 as CSR's.

5.1.2.1 Historical Note -

This Indirect path to the CSR's was discovered in early 1977 by an expedition consisting of my three brothers and myself.

Field Service Engineer, Memory Engineer, System Engineer, and myself Software Engineer,

departed the CPU on our search for the CSR's. Memory Engineer promised that at CSR's would conditionally reside in the shadow of a giant memory depending on whether or not they felt switched.

We began our search for the first CSR believing that if we found one, that the rest of the tribe would surely be nearby. Our task was clear, find CSR 17772100 behind the giant memory 0 through 16777776 (2 Meg words).

Field Service Engineer suggested that since we could arrive in the east by heading west, then surely we could find 17772100 by heading for 152100. This didn't make too much sense to my other brothers and myself, however, since Field Service was willing to pay for the expedition, we all left CPU city in search of 152100.

The first village we came upon was managed by an old experienced locator named Kay Tee. We told Mr. Tee of our expedition and our high expectations of finding 17772100 by looking for 152100. He expressed doubt that the feat could be accomplished, but he said since our three most significant bits were six that perhaps his most trusted friend Six (one of the Par brothers) could help us. So we proceeded to see Six Par.

We found Six Par at a busy highway control center directing traffic with his brothers. He greeted us warmly and assured us that we had come to the right place. Six said that his older brother Seven was

not to be trusted since he preferred devices to people. Also, Six^{E 6}
told us
how his younger brothers were much too inexperienced to guide
us and that the
y usually only do 1 to 1 relocations. So, at Six Par's
highway we were given 1
77400 which he said to multiply by 64 and add
to our 12100 (152100 less 3 M
SB's). Thus, we were leaving Kay Tee's
village with a total address of 1775210
0.

At the outskirts of the village there were two toll gates one to Cash
an
d one to Uni. We decided to take the Cash route at System
Engineer's
suggestion that "Cash makes no enemies". However, the gate
guard would not
let us pass because he thought we were a device. We
tried to explain but he in
sisted that unless our most significant bits
were 16, we could not pass. Re
luctantly, we decided to try the Uni
gate. Here the guard was more negotiable
and he let us pass charging
only four bits. Just outside the gate we board
ed a Unibus with ad-
dress 752100.

PAGE 44

The bus trip was supposed to be quite long with many stops, however, it turned out the first stop was the last stop for anybody without a device address. Here we departed the Unibus to find a long row of 36 checkout girls at special registers called Maps. Each Map register had a number 0 through 36 and a checkout girl with a matching bust size. We all agreed to visit Map register 36. The checkout girl was very cordial and went on to say that she preferred customers whose most significant five bits were 36 also. She charged us our five most significant bits and added her favorite number 17760000 to our address of 12100 and we departed with 17772100, which Memory Engineer promised would lead to the CSR.

Next we came to the cash highway, but since the cash register was turned off, we were declared amiss and allowed to pass all the way down the highway toll free where we arrived at the Memory bus. We gave the driver our address of 17772100 and proceeded on our way. The driver warned us that no one ever goes there and he waited for us at our stop as it was not a regular part of his route.

Here we were waiting at the bus stop with open arms, we found the CSR and the entire tribe. We briefly exchanged data and climbed back on the Memory bus as we had to return to CPU city before the last Unibus timed out.

The journey home was without incident. It is interesting to note that once home none of the engineers including myself, could remember what the checkout girl's face looked like.

5.1.3 Indirect Manual Mode: -

1. Turn off Cache 17777
746 <--- 14
2. Set up MAP0 17770200 <--- 170000
 17770202 <--- 77
3. Turn on Unibus Map 17772516
<--- 40
4. Turn display switch to CONS PHY
5. Examine or
deposit 17002100 through 17002136 as CSR's

5.2 Allocation of CSR Bits

5.2.1 Enable Parity Traps (Bit0 - of 1st Word) -

If this bit is set, it allows bad parity to be asserted (for a READ operation) if a double error occurs or a single error occurs with ECC disabled. Section 5.2.5 summarizes the operation of this bit along with other control bits. This bit is set on power up and can be written or read by the main memory bus.

5.2.2 EC Disable (Bit1 - 1st Word) -

If this bit is set, the error correction circuits are disabled and no single bit error correction will occur. However, error detection circuitry is enabled and any single or double bit error may still be detected on a READ operation and correct check bits are still written into memory on a write operation.

This is a diagnostic aid to allow writing and reading data from memory without interference from the error correction logic. This is a READ/WRITE bit cleared on power up.

5.2.3 Diagnostic Check (Bit2 - 1st Word) -

This mode allows a means of writing data stored in CSR Bits (14:8) into the MOS chips used for check bit storage. A READ cycle to memory will read the check bits from MOS storage into CSR bits (14:8).

This allows an integrity check on the check bit storage chips and a

check on the error correction logic.

This bit is cleared on power up and can be written or read by the main memory bus.

5.2.4 Protection Pointer (Bit3 - 1st Word) -

This bit selects the 16K section of memory which will be protected (not be affected by ECC disable and diagnostic check). This 16K section will contain the diagnostic program. When this bit is unasserted, the first 16K associated with each control card is protected. When this bit is asserted the second 16K associated with each control card will be protected. This is a READ/WRITE bit cleared on power up.

5.2.5 Summary of Bits(3:0) of 1st Word -

CSR	BIT3	BIT2	BIT1	BIT0	FUNCTION
Parity trap, disable from power up	0	0	0	0	Parity trap, disable from power up
Traps on SBE's	0	0	0	1	Enable traps on DBE's (Initialized Value)
Used by program	0	0	1	0	Disable error correction traps
Bits (14:8) of 1st word	0	0	1	1	Enable traps on SBE's
Program relocated	0	1	0	0	R/W checkbits
	0	1	0	1	Not used by program
	0	1	1	0	Not used by program
	0	1	1	1	Not used by program
	1	0	0	?	Read Box Capacity in Bit
	1	0	1	0	Disable error correction traps (Program relocated)
	1	0	1	1	Enable traps on SBE's (Program relocated)
	1	1	0	0	R/W Checkbits (Program relocated)
	1	1	0	1	Not used by program
	1	1	1	0	Not used by program
	1	1	1	1	Not used by program

? = DON'T CARE

5.2.6 Single Bit Error Indication (Bit4 - 1st Word) -

On a READ cycle to memory, if a single bit error occurs, this bit is set. Simultaneously, the syndrome bits are latched into CSR Bits (14:8) and the array identification and controller selected are latched into CSR (Bits [7:5] 1st word) and Bit9 2nd word.

An examination of the CSR can thus be performed to see if a single bit error had occurred and the array that failed if such error occurred. This is a READ/WRITE bit cleared on power up.

5.2.7 Array Select (Bits (7:5) 1st Word) -

During a READ cycle to memory, latched address bits on the selected control board will be stored in these locations upon an error. Any DBE will rewrite these bits & SBE's from protected memory will not be latched.

5.2.8 Check Bit/Syndrome Bit Storage/Box Capacity (Bits (14:8) 1st Word) -

5.2.8.1 Check Bit Storage (DIAG CK = 1) -

During a diagnostic check, bits are read from or written into memory as described in Section 5.2.3.

5.2.8.2 Syndrome Bit Storage -

If a single bit error occurs on a read cycle, then the syndrome bits indicating what bit failed are stored into those CSR bits.

These CSR bits can be written or read by the main memory bus. They are neither set nor reset on power up.

5.2.8.3 Hamming Code Matrix -

Each check bit (C1-C32) is the result of odd parity generated on the data bits designated by X.

$$\begin{aligned}
 C1 &= f(1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31) \\
 C2 &= f(2,3,6,7,10,11,14,15,18,19,22,23,26,27,30,31) \\
 C4 &= f(4,7,12,15,20,23,28,31) \\
 C8 &= f(8,15,24,31) \\
 C16 &= f(16,31) \\
 C32 &= f(32)
 \end{aligned}$$

Check bit CT (check total) is the result of even parity generated on the data bits.

$$CT = f(0,3,5,6,8,11,13,14,16,19,21,22,25,26,28,31)$$

Syndrome bits (S1-ST) are formed by a comparison of the check bits read from memory and new check bits generated on the data read from memory.

A single error will cause an odd number of syndrome bits to go high. If Bit 4 should fail S16, S8, & S4 would be set.

An examination of the syndrome bits can disclose the failing bit.

If there are more than one syndrome bits set and still an odd number of bits then the following simplification exists. S4-S1 indicate in octal the failing Bit (0-7). S32-S8 indicate the Bad Byte. The cleared bit in S32-S8 indicates which Byte from left to right. S32-S8 = 3 = Byte 0; 5 = Byte 1. 6 = byte 2; and all set = 7 = Byte 3. ST (Syndrome total) should be set or cleared to produce an odd number of syndrome.

If only one syndrome bit is set it directly indicates which check bit is bad.

If an even number of syndrome bits are set
something is wrong with the
hardware.

SYNDROME BITS

		BIT		ST	S32	S16	S8	S4	S2	S1
/	0	X	X				X	X		X
			/							
		X	X							
		BYTE 0	/				X	X	X	X
	L	4		X	X					
			L	X		X	X	X		
X										
			L	X		X	X	X	X	X
			L			X	X	X	X	X
		/	8			X				
			/							
		X	X			X				X
10		BYTE 1	/	X	X		X		X	X
			L			X				
			L	X	X		X	X		
X										
			L	X	X		X	X	X	
			L			X	X	X	X	
X										
			/	X	X	X				
			/			X	X			X
18		BYTE 2	/	X	X	X			X	X
			L			X			X	
			L	X	X	X		X		X
L	22	X	X	X	X	X		X	X	X
			L			X	X			
/	24		X	X	X	X	X			X
			/							
X	X	BYTE 3	/			X	X	X	X	X
			L			X	X	X	X	
			L	X	X	X	X	X		X
L	30	X	X	X	X	X	X	X	X	X
			L	X	X	X	X	X	X	X
/	C1			X						

C 7

SEQ 0080

CHECK BITS /
C2
C4
C8

L
L
L
CT

X X X
X X X

5.2.8.4 Box Capacity

Multiplexed with the locations for reading and/or writing check bits or reading syndrome bits is the ability to read the total number of 16K X 32 sections (double blocks) in memory. In order to read this box capacity, the special code as indicated in Section 5.2.5 (bits (3:1) = 1,0,0) will have to be loaded into the CSR 1st word prior to performing the READ cycle.

(Note that if ECC is manually disabled, writing a Zero into CSR Bit2 1st word is overridden and a One remains. In order to read the box capacity for this case, an internal flip-flop is used whose state is what was written into CSR Bit2 1st word. Hence writing Bits(3:1) = 1,0,0 1st word will still allow box capacity to be read even though CSR BIT2 1st word is still a One.)

5.2.9 Double Error Indication (Bit 15 of 1st or 2nd Word) -

On a READ cycle, these bits are set if a double error occurs. Simultaneously the array select bits latched on the control board selected are stored into CSR Bits (7:5) 1st word and the latched control is stored in Bit 9 2nd word. It should be noted that Bits 15 of 1st & 2nd word are separate and independent. Clearing one bit via a write to that byte will not affect the other. They are however asserted by the addressed controller simultaneously. These are READ/WRITE bits reset on power up.

5.2.10 Starting Address (A24-A16) (Bits (8:0) 2nd Word) -

These bits are used to load a starting box address through t

he CSR on
a WRITE cycle or can be used to read the starting address of the mem
ory box if the address is set by switches on the data interface card or
remotely from the processor. The starting address can be reconfigured
on a 32K word segment. These bits are READ/WRITE. On power up the
switch settings on the data interface card are loaded into these CSR
locations. In order to load the starting address Bit 10 2nd word must
accompany the load.

5.2.11

Control Select (Bit 9 2nd Word) -

If a single or double bit error occurs during a READ cycle, then the address of the controller selected is stored in this CSR bit location.

This bit is read only and cleared on power up.

5.2.12 Starting Address Selector (Bit 10 2nd Word) -

This bit controls selection of the starting address & External Interleave loaded into the CSR or the starting address and external interleave dictated by the switches on the data buffer board or control panel. A Zero in this bit indicates selection of the switches, a One indicates selection of the starting address loaded into CSR bits (8:0) 2nd word and External Interleave bits (14-12) 2nd word!

(Note that a READ operation to the CSR will always give the actual box starting address in Bits (8:0) 2nd word regardless of its origin.)

This is a READ/WRITE bit cleared on power up. (However, a switch on the data buffer board can hold this bit cleared, thereby defeating any attempt at loading the starting address and External Interleave through the CSR.)

5.2.13 Internal Interleaving (Bit 11 2nd Word) -

This bit indicates if there is internal interleaving within a memory box. If this bit is cleared there is no internal interleaving and if it is set, there is internal interleaving.

NOTE

Only if each controller has identical memory capacity is this bit capable of being set. However, writing a Zero to this bit disables all possible internal interleaving. This is a READ/WRITE bit

set on power up if each of the two control card sets has identical memory capacity.

5.2.14 External Interleaving (Between Boxes) (Bits (14:11) 2nd Word) -

These bits indicate how the system is configured for external interleaving as follows:

BIT 14	BIT 12	BIT 13	NUMBER OF WAYS EXTERNALLY INTERLEAVED
0	0	0	No External Interleaving
1	0	0	Not Used
0	1	0	2 Way: Memory Responds to Even Double Words
1	0	0	: Memory Responds to Odd Double Words
0	0	1	4 Way: Memory Selected If: A02=0; A03=0
1	0	1	: Memory Selected If: A02=1; A03=0
0	1	1	: Memory Selected If: A02=0; A03=1
1	1	1	: Memory Selected If: A02=1; A03=1

The bits will be treated in a similar manner to the Start Address as described in Section 5.2; i.e., CSR Bit10 2nd word will determine the origin. Also, on power up the corresponding switch settings will be loaded into these CSR bit locations.

5.2.15 Interleave Summary -

BIT14	BIT13	BIT12	BIT11	DESCRIPTION
0	0	0	0	No Interleave
0	0	0	1	Internal 2-1
0	0	1	0	External 2-1 Even Double Words
0	0	1	1	Ext 2- Even Double Words
0	1	0	0	Ext 4-1 A3=0 A2=0
0	1	0	1	Ext 4 A3=0 A2=0
0	1	1	0	Ext 4- A3=1 A2=0
0	1	1	1	Ext 4-1 Int A3=1 A2=0
1	0	0	0	Impossible (E RROR)
1	0	0	1	Impossible (ERROR)
1	0	1	0	Ext 2-1 Odd Double Wor ds
1	0	1	1	Ext 2-1 Int Odd Doub le Words
1	1	0	0	Ext 4-1 A3=0, A2 =1

1 1 0 1 Ext 4-1 Int
 A3=0,A2=

1

1 1 1 0 Ext 4-1
 A3=1,A2=1

1 1 1 1 Ext 4-1 Int
 A3=1,A2=1

5.3 CSR Bit Summary

CSR BIT ALLOCATION - AS
DESIGNED COMMENTS

1st word Bit
0 Enable Parity Traps : Set on Power Up (Read/Write)
Bit1 ECC Disable :
Clear on Power Up (R/W)
Bit2 Diagnostic Check : Clear on Power Up (R/W)

Bit3 Protection Pointer : Clear on Power Up (R/W)
Bit4 Single Bit Error Indication : Clear on Power Up (R/W)
Bit5 Array Sel A01 : Clear on Power Up
(Read Only)
Bit6 Array Sel A02 : Clear on Power Up (Read Only)
Bit7 Arr : Clear on Power Up (Read Only)
ay Sel A03 : Clear on Power Up (Read Only)
Bit8 CK Bit1/Syndrome 1/Capacity 1 : (Read/Write) Bits, Neither
y 1 : (Read/Write) Bits, Neither
Bit9 CK Bit2/Syndrome 2/Capacity 2 : Set n
or Reset on Power
Bit10 CK Bit4/Syndrome 4/Capacity 4 : Up; Based on CSR Bit
2 bit3 Check
Bit11 CK Bit8/Syndrome 8/Capacity 8 : Bits Alone are Read from
or
Bit12 CK Bit16/Syndrome 16/Capacity16 : Written into these locations
Bit13 CK Bit32/Syndrome 32/Capacity32 : Otherwise Syndromes are Written on SBE'
s
Bit14 CK Bit T/Syndrome T/Capacity64
Bit15 Double Bit Error Indication
: Clear on Power Up (R/W)

2nd word Bit0 Start Address (A16) : (Read/Write) B
its Capable of
Bit1 Start Address (A17) : Being Loaded via the Main Data
Bit2 Start Address (A18) : Bus or Switches on the Data
Bit3 Start Address (A19) : Card. On Power Up, the switch
are loaded into
Bit4 Start Address (A20) : Settings
Bit5 Start Address (A21) : These Locations.
Bit6 Start Address (A22)
Bit7 Start Address (A23)
Bit8 Start Address (A24)
Bit9 Co
ntroller Select : Cleared on Power Up (Read Only)
Bit10 Start Address Selec
tor : Loaded on Power Up (Read/Write)

Bit11 Internal Interleaving : Load
 d on Power Up (Read/Write)
 Bit12 Two Way : Loaded Same as Bits (8:0) 2nd
 word
 Bit13 Four Way : Loaded Same as Bits (8:0) 2nd word
 Bit14 Ext A02 : Loaded Same as Bits (8:0) 2nd word
 Bit15 Double Bit Error Indications :
 Clear on Power Up (R/W)

PAGE 55

6.0 SUB-TEST SUMMARIES

6.1 Tests

TEST 1

ANALYZE CONFIGURATION
(CSR Access may cause wrong Type of Traps)

TEST 2

TEST BANK 0 ACCESSES
Failures are fatal.

TEST 3

TEST
BANKS 1-167 (OCTAL) FOR ZEROS ONES
Errors are not typed here - only logged i
n
the configuration table

TEST 4

MK11 PROTECTION POINTER TEST

TEST 5

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

TEST 6

UNIQUE BANK TEST
Pattern 27 is run

TEST 7

FORCE ADDRESS PARITY ERRORS
Insures that no bank responds when there
is
an address parity error.

6.2 Patterns

6.2.1 General Pattern Information -

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

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

Patterns reside in 4 scripts that are scanned for execution. Symbolic location 'MKCSRT' is a table of patterns that can run once for each MK11 controller (twice per MK11 Box). Symbolic location 'MKPAT' is a table of patterns that can run on each Bank of MK11 memory. Symbolic location 'MJPAT' is a table of patterns that can run on each Bank of MJ11 memory. Symbolic location 'FSPAT' is a table of patterns that can be run in Field Service Mode (command 5).

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

Example E-table Segment:

```

:THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
:ARE TO BE RUN FOR PARTICULAR MEMORIES
:
:REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
:
: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

```

```

:
$DDW0: .WORD 177777 ;MK11 CSR TESTS      TABLE = MKCSRT:
$DDW1: .WORD 177777 ;MK11 CSR TESTS      TABLE = MKCSRT:
7      ;MK11 CSR TESTS      TABLE = MKCSRT:
$DDW2: .WORD 177777 ;MK11 PATTERNS      TABLE =
MKPAT:
$DDW3: .WORD 177777 ;MK11 PATTERNS      TABLE = MKPAT:
$DDW4: .WORD 177777 ;MK11 PATTERNS      TABLE = MKPAT:
;MJ11 PATTERNS      TABLE = MJPAT:
$DDW5: .WORD 177777 ;MJ11 PATTERNS      TABLE = MJPA
T:
```

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 executes out of the USER Instruction PAR's.

N

OTE

It is frequently modified dynamically such that (1) it returns after writing only (the 1st NOP is replaced with a RE-TURN) or (2) it only counts Errors (the code PERRO2 and NOP are replaced with INC @#PATERR)

6.2.2.2 Pattern 1 Address Test -

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

It executes 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 executes 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 4 and 10 change. This is because the PDP-11/70 fetches double words therefore a classical 3 XOR 9 in a small PDP-11 must look like a 4 XOR 10 in a PDP-11/70.

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 executes 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 Rota
ting Zeros Test -

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

It executes 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.

PAGE 59

6.2.2.6 Pattern 5 Rotat
ing 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 compleme
nt of the coverage of Pattern 4 in data.

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

NOTE

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

PAGE 60

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 B

it 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

ess 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.

GE 61

6.2.2.10 Pattern 11 Single Bit Error Test -

1. Create a Single Bit Error.
2. Read data Uncorrected (with ECC Disable).
3. Read First Word of data corrected (with ECC Enabled)
4. Check that the CSR Single Bit Error Flag was set.
5. Clear SBE Flag.
6. Read Second word of data corrected (with ECC Enabled).
7. Check that the CSR Single Bit Error Flag was set.
8. Do (1-7) for a Single Bit Error in each of 32 positions of a double word.
i.e. (32 Times)
9. 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)
10. 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))
11. Clear any errors out of test locations.

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

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 Byte that had SBE then the SBE Flag should be OFF.
If Byte that did not have SBE then SBE Flag should be set.
5. Byte should have been equal to Ones.
6. Do (1-5) for each of the 4 Bytes of the Double Word
7. Do (1-6) for a Single Bit Error in each of 32 positions of a Double Word
i.e. (32 Times)
8. If not in Quick Verify Mode then do (1-7) for data consisting of 1 Bit set in each of 32 positions of a double word.
i.e. (32 X 32 = 1024 Times)
9. Clear any errors out of test locations.

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

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 Flags are both set.
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. 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 un
der ACT or APT.

PAGE 64

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. 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.

6.2.2.14 Pattern 15 wr
ite Inhibit Of Byte With DBE -

1. Create a Double Bit Error.
2. Do a MOVB immediate to test byte.
3. Check that Double Word is sti
(1 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 the
n 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. Clear any errors out of test locations.

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

NOTE

n during the This test is only ru
first
(QV) pass when under ACT or APT.

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. 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.

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

SEQ 0107

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.

PAGE 67

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

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

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

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

1. Write a Background of alternating Bytes of Zeros & Ones
2. For the 16K Bank addressing Down
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word
3. For the 16K Bank addressing Up
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word
4. For the 16K Bank addressing Up
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word
5. For the 16K Bank addressing Down
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word

This checks the integrity of the 32 Bit Double Words.
 It executes 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 posi-

ould

tion relative to a byte then you s
ignore the second error report.

6.2.2.19 Pattern 22
Refresh Test -

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

This pattern is on
addresses not bit positions.

Example:

Address	MSB's
1 0 0 0 1 0	----- LSB's : 0 0 0 1 0 0 0 1 : 0 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 : 1 0 0 0 1 0 0 0
0	

NOTE

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

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

NOTE

This test is not normally executed except under APT or ACT.
It may be 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
6.2.22 Pattern 25 Interrupt Enable Test -

1. Set CSR to Allow DBE Traps.
2. Access Test Double Words.
3. Check for no Double Bit Error Trap.
4. Enable CSR for SBE Traps.
5. Access Test Double Words.
6. Check for no SBE Trap.
7. Write a SBE in 1 Byte.
8. Disable CSR for Parity Traps.
9. Access Test Double Words.
10. Check for no Traps.
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 for Parity Traps.
17. Access the Test Double Word.
18. Check for no Traps.
19. Enable CSR for DBE Traps.
20. Access the Test Double Word.
21. Check to Insure Trap Occu

rred.

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.

PAGE 72

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 executes out of the User Data PAR's, the Kernel Data PAR's, and the Supervisor Data PAR's.

6.2.2.24 Pattern 27 Unique Bank Test -

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

It does not test Banks that require relocation to test.

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

6.2.2.25 Pattern 30 Flush Out DBE's Test -

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

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

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,
I believe it is 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
M
OV 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',
'SD
DW2-5'. Field Service Mode command 8 is the normal method of run-
ning this p
attern.

NOTE

This test i
s not normally executed ex-
cept under APT or ACT. It may
be in-
voked VIA Field Service Command 13 (Ka-
mikaze Mode).

PAGE 74

6.2.2.27 Pattern 32 Skewed Up Address Test -

Referen
ce the Multiport section

6.2.2.28 Pattern 33 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.

W
RITE 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 A

S 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 "J"
MP (RO)" INSTRUCTION IS MET
AND EXECUTED. RO CONTAINED THE RETURN ADDRESS B

ACK

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).

GE 75

6.2.2.29 Pattern 34 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.

```

BGTES
T:      0                ;TEST WORD
BRGOBB: SEC              ;INC LOW BYTE
        ADCB    BGTEST   ;END LOOP
        BMI     1$
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$:
        CLV
R V-BIT
        INCB    BGTEST   ;INC HIGH BYTE ONE LAST TIME
        BCS     3$       ;BRANCH IF C-BIT S
ET (SHOULD NOT BE)
        BVC     3$       ;BRANCH IF V-BIT CLEAR (SHOULD NOT BE)
        BMI     4$       ;B
RANCH IF N-BIT SET (SHOULD BE)
3$:     ERROR          ;ERROR TRAP
4$:     RETURN
    
```

This code originally came from the PDP-11 Family Instruction Exerciser DZQKA-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 APT or ACT. It may be invoked

VIA Field Service Command 13 (Ka-
mikaze Mode).

G 10

SEQ 0123

6.2.2.30
Pattern 35 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.

6.2.2.31 Pattern 999 Null Test -

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

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

77

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.

It is my belief that one of the prime reasons that system programs like RST, S, 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 with his silo backed up (perhaps only 3-8 words) 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.

I have attempted to optimize this diagnostic's error detecting abilities around the primary design criteria of speed. To this end I have taken the following steps.

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 Su

pervi-
sor mode for mapping purposes. All core patterns and some MOS Pat-
terns are subroutines that are moved to this Bipolar region referred
to in th
e program as Fast City.

PAGE 78

7.1.2 SOB's -

Utilization of the PDP-11/70 Instruction Set to speed pattern algorithms (principally the SOB). No attempt is made to support anything less than a PDP-11/70.

7.1.3 CACHE -

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

7.2 Bank Zero Testing

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

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

Systems with memory management can overcome this because all traps are in Kernel Virtual space even power 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.

PDP-11/70 Systems 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).

7
7.3 Differences In Core Testing With 0-2 MEG (DEMJA)

7.3.1 Delta Noise
Test Only Performed With 3 XOR 9 Test -

Reason: DEMJA performed the delta noise test with constant data however, it will only have an adverse effect if performed with a worst-case pattern.

7.3.2 8 XOR 13 Test Deleted -

Reason: Designed for a PDP11-20 Memory not presently in use.

7.3.3 3 XOR 9 Rewritten -

Reason: was actually a 4 XOR 9 and should have been a 4 XOR 10 for PDP-11/70's to account for reading Double Words.

a Hexadecimal number indicating which
CPU's (SLAVE3,SLAVE2,SLAVE1,MASTER) could
access each bank.

The interleaving on Banks 0-37 is 2 way. This may be external with no internal or internal with no external - however since there is only one box of MOS (BOX 0) it must be internal interleave.

Banks 0-37 are Memory Type K (MK11) and Banks 40-167 do not exist. Memory Type J would indicate MJ11 memory.

Banks 0-37 are located in box 0. Boxes are assigned to CS R locations in order and MJ11 Boxes will never have a Box number only a question mark.

Banks 0 and 2 are protected because they are program space. Bank 0 and 2 are also protected because they are in the bottom 16K of an MK11 controller. The protection is hierarchical and program space overshadows MK11 protection. Banks 0 and 2 will not be tested until the program relocates. If any bank is protected by MK11 and not because it is in program space it will have a 'K' typed in this row.

7.5 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 differently

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.

```

7.5.
1 Sample Source File -
  .ENABL ABS
  .ENABL AMA
  .MCALL .SUPER
  .SUPER
  ;L
STSS=0 .
  BITS=40
A:      0
B:      0
C:      0
D:      0
E:      0
F:      0
G:      0
H:      0
I:      0
J:      0
;L
ET 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 AN
D 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 RES
ULT 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 EXAM
PLES
    $RETURN
    $RETURN ERROR
    $RETURN NOERROR
:CASE EXAMPLES
    MOV A,RO
    CA
SE RO
    A
    B

```

C
D
E
F
END ;OF CASE RQ
.END

```

7.5.2 S
ample Listing File (with no expanded macros) - -
S 1 000000 .ENABL AB
2
3 .ENABL AMA
4 000000 .MCALL .SUPER
R .SUPE
5
6 ;LST$$=0
7 000000 000040 BIT5=40
A: 8 000002 000000 B: 0
9 000004 000000 C: 0
10 0000 D: 0
06 11 000010 000000 E: 0
12 000012 000000 F: 0
13 000014 000000 G: 0
14 000016 000000 H: 0
15 000020 00 I: 0
0000 16 000022 000000 J: 0
18 ;LET EXAMPLES
19 LET RO := A
000024 20 000030 LET B := C + D
21 000044 LE
TE := F + #1 22 000056 LET G := H + #2
23 000072 LET J :=
J + #01 24 000100 LET A :B= B
25 ;IF EXAMPLES
106 26 000 IF A IS TRUE
27 000114 012737 000023 000006 MOV #23,D
28 29 000122 END ;OF IF A
30 000130 01 IF B IS FALSE
2737 31 000136 000034 000010 / MOV #34,E
32 00 END ;OF IF B
0136 33 000154 IF A EQ B THEN LET C := D
34 000164 IF A LT B
013737 35 000172 000004 000006 MOV C,D
737 36 000174 013 ELSE
37 000010 000006 MOV E,D
38 000202 END ;OF IF A
38 000202

```

```

39 000222 013737 000012 000014 IF A EQ B AND C NE D
                                MOV F,G
40 000230                                END ;OF IF A
41 000230                                IF A EQ B OR C NE D
42 0
00250 013737 000012 000014 MOV F,G
43 000256                                END ;OF IF A
44 000256                                IFB A EQ B AND C EQ #1
45 000276 013737 000016 000022
MOV H,J                                ELSE
46 000304                                MOV
47 000306 013737 000010 000022
E,J                                    END ;OF IFB A
48 000314                                IFB A EQ B ANDB C EQ
49 000314                                #1
50 000334 013737 000016 000022 MOV H,J
51 000342                                ELSE
52 000344 013737 000010 000022 MOV E,J
53 000352                                END ;OF I
FB A                                    IF RESULT IS EQ
54 000352
55 000354 013737 000000 000002
MOV A,B                                END ;OF IF RESULT
56 000362                                IF #BIT5
57 000362                                SET.IN A
58 000372 013737 000002 000004 MOV B,C
59 000400
END ;OF IF #BIT5

```

```

60 000400          IF #BIT5 OFF.IN A
61 000410 013737          MOV C,D
   000004 000006
62 000416          END ;OF IF #BIT5
63
;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
64          ;ON.ERROR EXAMPLES
65 000416          ON.ERROR
66 000420 013737 000000 000002          MOV A,B
67 000426          ELSE
68 000430 013737 000004 000002          MOV C,B
69          END ;OF ON.ERROR
70 000436          ON.NOERROR
71 000440 0
13737 000004 000002          MOV C,B
72 000446          ELSE
73 000450 01373          MOV A,B
7 000000 000002          END ;OF ON.NOERROR
74 000456
75 00
0456          ON.ERROR THEN LET A :B= B
76          ;FOR EXAMPLES
77 000466
   000000 000000          FOR I := #-5 TO #23
78 000474 005237 000000          INC A
79 00050
0          END ;OF FOR I
80 000514          FOR R0 := #0 TO #140 BY #4
81 0
00516 005360 000000          DEC A(R0)
82 000522          END ;OF FOR R0
83
000534          FOR I := #133 DOWNTD #3 BY #2
84 000542 063737 000000 0000          ADD A,B
02          END ;OF FOR I
85 000550          ;BEGIN EXAMPLES
86
87 000566          BEGIN ALPHA
88 000566          FOR R0 := #0 TO #167
89          000570 116037 000000 000002          MOV B A(R0),B
90 000576          IF
B LT #0 THEN LEAVE ALPHA
91 000604          END ;OF FOR R0
92 000614          FOR R0 := #400 TO #567
93 000620          IF B GE #0 THEN LEAVE ALPHA
94 000626          END ;OF FOR R0
95 000636          END ALPHA
96
;$RETURN EXAMPLES

```

97	000636		\$RETURN
98	000640		\$RETURN ERROR
99	000644		\$RETURN NOERROR
100			;CASE EXAMPLES
101	000650		
	013700	000000	MOV A,R0
102	000654		CASE R0
103	000664	000000	
		A	
104	000666	000002	B
105	000670	000004	C
106	00		
0672	000006		D
107	000674	000010	E
108	000676	000012	
F			
109	000700		END ;OF CASE R0
110			
111		000001	.END

7.5.3

Sample Listing File (with expanded macros) -

```

S 1 000000 .ENABL AB
2 .ENABL AMA
3 .MCALL .SUPER
R 4 000000 .SUPE
5 000000 LST$$=0
6 000040 BIT5=40
7 000000 000000
00 8 000002 000000 A: 0 B: 0
9 000004 000000 C: 0
1 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
020 15 000 000000 I: 0
16 000022 000000 J: 0
18 ;LET EXAMPLES
19 000024 013700 000000 LET R0 := A MOV A,R0
20 000030 013737 000004 000002 LET B := C + D MOV C,B
0036 00 063737 000006 000002 ADD D,B
21 000044 LET E := F + # MOV F,E
1 000044 013737 000012 000010
000052 005237 000010 INC E
22 000056 013737 000016 000014 LET G := H + #2 MOV H,G
23 000064 062737 000002 000014 ADD #2,G
72 0000 062737 000001 000022 LET J := J + #01 ADD #01,J
24 000100 113737 000002 000000 LET A :B= B MOV B,A
25 ;IF EXAMPLES
26 000106 005737 000000 IF A IS TRUE TST A
000112 001403 BEQ L0

```

```

3 27 000114 012737 00002
    000006      MOV #23,D
28 000122      END ;OF IF A
    000122
: : : : :
29 000122      IF B IS FALSE
    000122 005737 000002
                                TST B
                                BNE L1
30 000126 001003
    000130 012737 000034 000010      MO
V #34,E
31 000136      END ;OF IF B
    000136
                                L1: : : : :
32 000136      IF A EQ B THEN LET C := D
P A,B
    000136 023737 000000 000002
                                CM
                                BNE L2
    000144 001003
    000146 013737 000006 000004
    MOV D,C
    000154
33 000154      IF A LT B
    000154 023
737 000000 000002      CMP A,B
    000162 002004      BGE L3
34 000164 013737 000004 000006      MOV C,D
000164 000172      ELSE
35 000172 0004
                                BR L4
03 000174
36 000174 013737 000010      MOV E,D
    000006
                                L3: : : : :

```

```

37 000202          END ;OF IF A                      L4:****
   000202
::
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
   000222 013737 000012
39 000222          MOV F,G
   000014
40 000230          END ;OF IF A                      L5:****
   000230
::
41 000230          IF A EQ B OR C NE D
   000230 023737 000000 000002          CMP A,B
   000236 001404          BEQ L6
   000240 023737 000004 000006          CMP C,D
   000246 001403          BEQ L7
   000250          L6:*****
42 000250 013737 000012 000014          MOV F,G
43 000256          END ;OF IF
A
   000256          L7:*****
44 000256          IFB A EQ B AND C EQ #1
   000          CMPB A,B
256 123737 000000 000002          BNE L10
   000264 001010
   000266 023727 000004 000001          CMP C,#1
   000274 001004          BNE
L10
45 000276 013737 000016 000022          MOV H,J
46 000304          ELSE
   000304 000403          BR L11
   000306          L10:*****
47 000306          MOV E,J
   013737 000010 000022          END ;OF IFB A
48 000314          L11:*****
   000314          IFB A EQ B ANDB C EQ #1
49 000314          CMPB A,B
   000314 123737          BNE L12
   000000 000002
   000322 001010          CMPB C,#1
   000324 1          BNE L12
23727 000004 000001
   000332 001004

```

50 000334 013737 000016 000022
 51 000342
 000342
 000403
 000344
 52 000344 013737
 000010 000022
 53 000352
 000352
 L13:::~::~
 54 000352
 000352 001003
 4
 55 000354 013737 000000 000002
 56 000362
 IF RESULT
 000362
 57 000362
 000362 032737 000040 000000
 000370 001403
 BEQ L15
 58 000372 013737 000002 000004
 59 000400
 ND ;OF IF #BIT5
 000400
 60 000400
 .IN A
 000400 032737 000040 000000
 000406 001003
 61 000410 013737 000004 000006
 62 000416

MOV H,J
 ELSE
 MOV E,J
 END ;OF IFB A
 IF RESULT IS EQ
 MOV A,B
 END ;OF
 IF #BIT5 SET.IN A
 MOV B,C
 E
 IF #BIT5 OFF
 BNE L16
 MOV C,D
 END ;OF IF #BIT5

BR L13
 L12:::~::~

BNE L1

L14:::~::~

BIT #BIT5,A

L15:::~::~

BIT #BIT5,A

```

000416
63 000416 103004 ;ON.ERROR IS
LIKE AN IF STATEMENT ON THE C-BIT
64 000420 013737 000000 ;ON.ERROR EXAMPLES
65 000426 000002 MOV A,B
6 000426 000403 ON.ERROR
66 000430 013737 000004 000002 MOV C,B
67 000436 000403 ELSE
68 000436 013737 000004 000002 MOV C,B
69 000436 000403 END ;OF ON.ERROR
70 000436 103404 ON.NOERROR
436 000440 013737 00 MOV C,B
0004 000446 000403 ELSE
L22 000450 013737 000000 000002 MOV
73 000456 000403 END ;OF ON.NOERROR
74 000456 103003 ON.ERROR THEN LET A :B= B
75 000456 000002 000000 MOV B,A
460 000466 012737 000020
76 000474 000000 INC A
77 000500 005237 000 END ;OF FOR I
005237 000504 023727 000020 000023 INC I
020 000512 000514 FOR R
003770 000514
80 000514 005000
0 := #0 TO #140 BY #4
000514 005000

```

L16:.....

BCC L17

BR L20

L17:.....

L20:.....

BCS L21

BR

L21:.....

L22:.....

BCC L23

MOV B,A
L23:.....

MOV #-5,I
B0:.....

CMP I,#23

BLE B0
E0:.....

CLR R0

```

000516
:.....
81 000516 005360 000000
82 000522

```

```

DEC A(R0)
END ;0

```

```

F FOR R0
000522 062700 000004
000526 020027 000140

```

ADD #4,R0

CMP R0,#140

BLE B1

```

000532 003771
000534

```

E1:.....

```

::
83 000534 012737 000133
000534 012737 000133
000020
000542

```

FOR I := #133 DOWNT0 #3 BY #2

MOV #133,I

B2:.....

```

84 000542 063737
000000 000002

```

ADD A,B

END ;OF FOR I

```

85 000550 162737 0
000550 162737 0

```

SUB #2,I

```

00002 000020 000556 023727 000020 000003

```

CMP I,#

```

3
000564 002366
000566

```

BGE B2

E2:.....

```

86
BEGIN EXAMPLES

```

BEGIN ALPHA

B3:.....

```

87 000566
000566

```

```

88 000566 005000
000566 005000
000

```

FOR R0 := #0 TO #167

CLR R0

570

B4:.....

PAGE 89

```

      89 000570 116037 000000 000002      MOVB      A(R0)
      90 000576      IF B LT #0 THEN LEAVE ALPHA
      000576 005737 000002
                                     TST B
      000602 002415
      91 000604      END ;OF F
                                     BLT E3
OR RO
      000604 005200
      000606 020027 000167
                                     INC R0
                                     CMP R0
      #167
      000612 003766
      000614
      92 0
      00614      FOR R0 := #400 TO #567
                                     MOV #400,
RO
      000614 012700 000400
                                     B5:
      000620
      93 000620      IF B GE #0 THEN LEAVE AL
      000620 005737 000002
      000624 002004
                                     TST B
                                     BGE E3
      94 000626
      000626 005200
      000630 02
      0027 000567
      000634 003771
      000636
      95 000636
      000636
      96
      97 000636
      000636 000207
      98 000640
      000640 000261
      00
      0642 000207
      99 000644
      000644 00024
      1
      000646 000207
      100
      101 000650 013700 000000      ;CASE EXAMPLES
      102 000654
      000654 0
      10046
      000656 006316
      000660 004737
      000700
      103 000664 000000
      A

```

```

002 104 000666 000
      105 000670 000004
      106 000672 000006
      1
07   000674 000010
      108 000676 000012
      109 000700
:OF CASE R0
      000700
      000700 062616
SP   000702 013646
      000704 004736
.a(SP)+
110
111      000001

```

B

C
D

E
F
END

```

L24:.....
      ADD (SP)+,@
      MOV @ (SP)+,-(SP)
      JSR PC

```

.END

7.6 Banks & Boards (MK11)

Boards with 4K MOS chips are 32K words each.
Boards with 16K MOS chips are 128K words each.

Imagine that the front view of the memory box is as follows:

BL, 15
13, 11, 9, 7, 5, 3, 1, CB, CA, DA, BL, AD, BL, CA, CB, 0, 2, 4, 6, 8, 10, 12, 14, BL

Where 0 thru 15 are the MOS array cards. CB is Control 'B'. CA is Control 'A'. DA is the Data Interface. AD is the Address Interface. BL is a Blank (unwired) slot in the backplane.

4k Parts with No Interleave

Array 15,13,11, 9, 7, 5, 3,
 1, 0, 2, 4, 6, 8,10,12,14
 Banks 36 32 26 22 16 12 6 2 0 4 10 14 20 24 30 34
 37 33 27 23 17 13 7 3 1 5 11 15 21 25 31 35

4k Parts with Internal Interleave

Array 15,13,11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8,10,12,14
 Banks 34 30
 24 20 14 10 4 0 0 4 10 14 20 24 30 34
 35 31 25 21 15 11 5 1 1 5 11 15
 21 25 31 35
 36 32 26 22 16 12 6 2 2 6 12 16 22 26 32 36
 37 33 27 23 17
 13 7 3 3 7 13 17 23 27 33 37

4k Parts with 2-1 External No Internal Interleave

Array 15,13,11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8,10,12,14
 Banks 74
 64 54 44 34 24 14 4 0 10 20 30 40 50 60 70
 75 65 55 45 35 25 15 5 1 11 21
 31 41 51 61 71
 76 66 56 46 36 26 16 6 2 12 22 32 42 52 62 72
 77 67 57 47 3
 7 27 17 7 3 13 23 33 43 53 63 73

4k Parts with 2-1 External Internal Interleave

Array 15,13,11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8,10,12,14
 Banks 70 6
 0 50 40 30 20 10 0 0 10 20 30 40 50 60 70
 71 61 51 41 31 21 11 1 1 11 21 3
 1 41 51 61 71
 72 62 52 42 32 22 12 2 2 12 22 32 42 52 62 72
 73 63 53 43 3
 3 23 13 3 3 13 23 33 43 53 63 73
 74 64 54 44 34 24 14 4 4 14 24 34 44 54 6
 4 74
 75 65 55 45 35 25 15 5 5 15 25 35 45 55 65 75
 76 66 56 46 36 26 16 6
 6 16 26 36 46 56 66 76
 77 67 57 47 37 27 17 7 7 17 27 37 47 57 67 77

4k Parts with 4-1 External No Internal Interleave

4K Parts with
erleave

4-1 External Internal Int

Array 15, 13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10, 12, 14

Banks 160 140 120 100 60 40 20 0 0 20 40 60 100 120 140 160

41 121 101 61 41 21 1 1 21 41 61 101 121 141 161

2 42 22 2 2 22 42 62 102 122 142 162

3 23 43 63 103 123 143 163

64 104 124 144 164

145 165

147 127 107 67 47 27 7 7 27 47 67 107 127 147 167

70 50 30 10 10 30 50 70 110 130 150

11 31 51 71 111 131 151

12 132 152

154 134 114 74 54 34 14 14 34 54 74 114 134 154

55 35 15 15 35 55 75 115 135 155

6 56 76 116 136 156

7 157

16K Parts with
No Interleave

Array 15, 13, 11, 9, 7, 5, 3, 1

Banks 40 60 100 150 130 110 70 50 30 10 0 20

1 141 161

153 133 113 73 53 33 13 3 23 43 63 103 123 143 163

74 54 34 14 4 24 44 64 104 124 144 164

5 25 45 65 105 125 145 165

66 106 126 146 166

PAGE 93

16K Parts with External Interleave		2-1 External and No Interna												
Array 15, 13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10, 12,														
14 Banks														
1	41 101 141	160	120	60	20	0	40	100	140					
		161	121	61	21									
		162	122	62	22	2	42	102	142					
03	143	163	123	63	23	3	43	1						
		164	124	64	24	4	44	104	144					
		165	125	65	25	5	45	105	145					
		166	126	66	26	6	46	106	146					
		167	127	67	27	7	47	107	147					
		1												
30	70 30 10 50 110 150	131	71	31	11	51	111	151						
		132	72	3										
2	12 52 112 152	133	73	33	13	53	113	153						
		134	74	34	14	5								
4	114 154	135	75	35	15	55	115	155						
		136	76	36	16	56	116	1						
56		137	77	37	17	57	117	157						

16K Parts with External Interleave		2-1 External and Int												
Array 15, 13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10,														
12, 14 Banks														
141		140	100	40	0	0	40	100	140					
		141	101	41	1	1	41	101						
		142	102	42	2	2	42	102	142					
		143	103	43	3	3	43	103	143					
144	104 44 4 4 44	104	144											
		145	105	45	5	5	45	105	145					
		146	106											
46	6 6 46 106 146	147	107	47	7	7	47	107	147					
		150	110	50	10									
10	50 110 150	151	111	51	11	11	51	111	151					
		152	112	52	12	12	52							
112	152	153	113	53	13	13	53	113	153					
		154	114	54	14	14	54	114	154					
		155	115	55	15	15	55	115	155					
		156	116	56	16	16	56	116	156					
		157												
117	57 17 17 57 117	157												

						160 120 60	20 20 60	120 160
21	21	61	121	161		161 121 61		
						162 122 62	22 22 62	122 162
63	123	163				163 123 63	23 23 63	
						164 124 64	24 24 64	124 164
165						165 125 65	25 25 65	125 165
						166 126 66	26 26 66	126 166
						167 127 67	27 27 67	127 167
	130	70	30	30	70	130		
						131 71 31	31 31 71	131
32	72	132				132 72 32		
						133 73 33	33 33 73	133
						134 74 34	34 34 74	134
	135	75	35	35	75	135		
						136 76 36	36 36 76	136
37	77	137				137 77 37		

16K Parts with 4-1 External and No Internal Interleave

Ar ray Banks 15, 13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10, 12, 14

40 0 100 141 41 1 101 142 42 2 102 143 43 3 103

144 44 4 104 145 45 5 105 146 46 6 106 147 47 7 107

113 150 50 10 110 151 51 11 111 152 52 12 112 153 53 13 154 54 14 114 155 55 15 115 156 56 16 116 157 57

17 117 160 60 20 120 161 61 21 121 162 62 22 122 163

63 23 123 164 64 24 124 165 65 25 125 166 66 26 126

167 67 27 127 70 30 130 71 31 131 72 32 132 73 33 133 74 34 134 75 35 135 76 36 1

36 77 37 137

16K Parts with 4-1 External and Internal Interleaving

Array Banks 15, 13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10, 12, 14

		100	0	0	100
		101	1	1	101
		102	2	2	102
		103	3	3	1
03		104	4	4	104
		105	5	5	105
		106	6	6	106
		107	7		
7	107	110	10	10	110
		111	11	11	111
		112	12	12	112
		113	1		
3	13 113	114	14	14	114
		115	15	15	115
		116	16	16	116
		1			
17	17 17 117	120	20	20	120
		121	21	21	112
		122	22	22	122
		123	23	123	
	123 23	124	24	24	124
		125	25	25	125
		126	26	26	12
6		127	27	27	127
		130	30	30	130
		131	31	31	131
		132	32	32	
	132	133	33	33	133
		134	34	34	134
		135	35	35	135
		136	36		
	36 136	137	37	37	137
		140	40	40	140
		141	41	41	141
		14			
2	42 42 142	143	43	43	143
		144	44	44	144
		145	45	45	145

146 46 46 146

147	47	47	147
150	50	50	150
151	51	51	151
152	52	52	152
153	53	53	153
154	54	54	154
155	55	55	
156	56	56	156
157	57	57	157

155

PAGE 96

61	161	160	60	60	160
		161	61		
		162	62	62	162
		163	63	63	163
		164	64	64	164
5	65	65	165	16	
		166	66	66	166
		167	67	67	167
			70	70	
71	71				
		72	72		
		73	73		
		74	74		
		75	75		
76	76				
		77	77		

7.7 Memory Management Mapping

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

7.8 Listing Format

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

You see -
you can have your cake and eat it too!

8.0 MULTIPOINT SYSTEM ORGANIZATION

The multipoint section of the MKA11 Diagnostic runs after the completion of testing as a single port system but before the end-of-pass code.

The tests are performed on a Master-slave basis with the Master being the CPU that the diagnostic is started at.

8.1 Starting Slaves

The Master will use the IIST and Multiprocessor boot to start and identify all slave CPU's with software locks to insure that only one CPU runs at a time (not ASRB locks). If the IIST or Multiprocessor Boot is not available they may be bypassed by starting at address 204 and following the dialogue for manual intervention.

8.1.1 Starting Code -

As each slave starts it executes shared code with the master using its own unique stack. The master has set the "CPUBIT" to indicate slave status and loops until the slave started places his unique IIST ID number (or System ID number if IIST's are bypassed) in the port directory. This is done for each slave, after which the master sets the "CPUBIT" to indicate master status.

Each slave when started sizes memory indicating accessed banks via a Bit Set of the "CPUBIT" into the configuration table. If IIST's are in use it enables interrupts on its IIST, lowers its priority to 4 and executes a WAIT until the master wakes it up. If IIST's are by-

passed it moves a little program into the User PAR's and Jumps to it.
This program waits for the master to write a countdown from 10 to 0 in the slaves pseudo TTY buffer location. Either way this is known as the Sleep state.

After all slaves have been put to sleep the master checks the port directory of ID numbers and insures that each CPU has a unique number.

Testing is now started as in a normal single port environment. Each slave remains dormant during this time.

8.2 Multiport Testing

The master wakes each slave via a
n interrupt with the IIST (or if
IIST's are bypassed via a countdown).

The master then orders each
slave to run the MK11 CSR tests. The master now
orders each slave to
relocate to a unique memory bank that has no errors. Us
ually this is
puts slave 1 in Bank 1, slave 2 in Bank 3 and slave 3 in bank 4).

The master then directs each slave and also himself in the execution
of t
he following tests.

Access Paths Test

Address Test

Asynchronous Access Address Un

iqueness Test

Skewed Up Address Test

Port Arbitration Symmetry Test

I/O Priority T

est

Execution Contention Test

ASRB Test

CACHE Flush Test

Next the master orders
each slave to stop and each slave will map back
into bank 0 (shared code area)
, lower it's priority to 4 and execute a
wait instruction thereby sleeping unti
l the next pass.

The master then returns to single port testing just prio
r to the
end-of-pass code. r.

PAGE 101

8.2.1 Access Paths Test -

The master causes each CPU (making himself a slave when necessary) to write into memory and all other CPU's to simultaneously read.

This is done as follows:

```
FOR EACH CPU
  FOR EACH BANK
    IF BANK IS ACCESSABLE & NOT PROGRAM SPACE
      FOR EACH OF 5 PATTERNS
        SELECTED CPU WRITES
        ALL OTHER CPU'S READ
        NEXT PATTERN
      NEXT BANK
    NEXT CPU
```

8.2.2 Address Test -

The master causes each CPU (making himself a slave when necessary) to write an address pattern into memory and all other CPU's to simultaneously read.

This is done as follows:

```

      FOR EACH CPU
        FOR EACH BANK
          IF BANK IS ACCESSABLE
            & NOT PROGRAM SPACE
              FOR EACH ADDRESS PATTERN
                --AND COMPLEMENT ADDRESS PATTERN
                  SELECTED CPU WRITES
                    ALL OTHER CPU'S READ
                      NEXT PATTERN
                      NEXT BANK
                NEXT CPU

```

8.2.3 Asynchronous Access Address Uniqueness Test -

The master causes each CPU (making himself a slave when necessary) to test 5 patterns on 1/4 bank simultaneously. This insures that no address errors will allow corruption of another CPU's data.

This is done as follows:

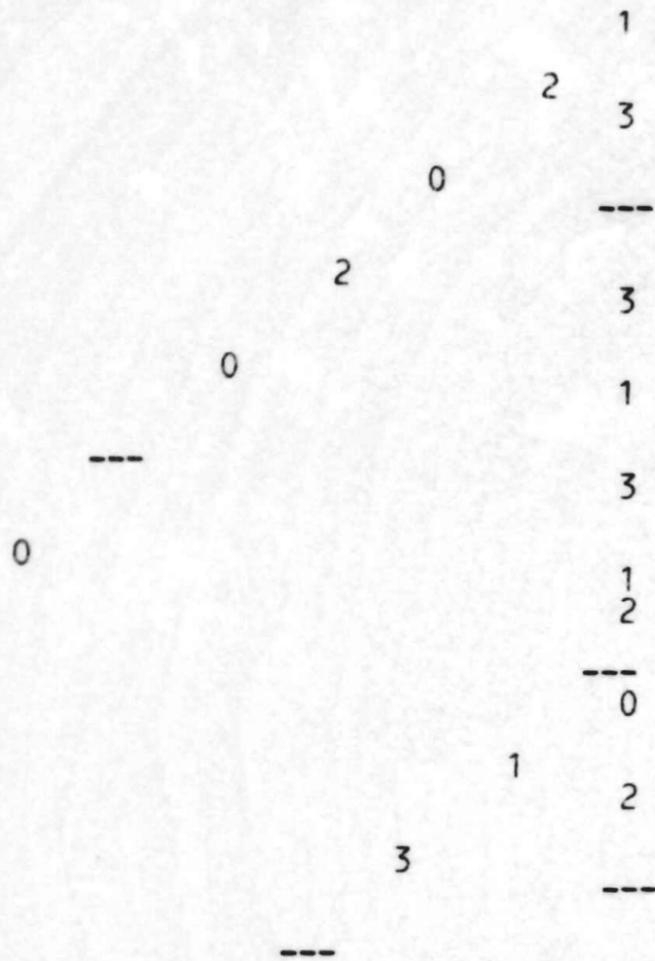
```

      FOR EACH BANK
        IF BANK IS ACCESSABLE & NOT PROGRAM SPACE
          FOR EACH OF 4 STARTING ADDRESSES
            FOR EACH OF 5 PATTERNS
              EACH CPU TESTS 5 PATTERNS ASYNCHRONOUSLY
                NEXT PATTERN
              NEXT STARTING ADDRESS
            NEXT BANK

```

8.2.4 Skewed Up Address Test -

The master causes each CPU (making himself a slave when necessary) to write and read there IIST ID number in every 4th address - skewed by 1 address and offset by there IIST ID number. Assuming a 4 CPU system then example data would be.



This is done as follows:

```

      FOR
EACH BANK      IF BANK IS ACCESSABLE & NOT PROGRAM SPACE
                FOR EACH OF 4 STARTING ADDRESSES
EACH CPU RUNS THE SKEWED UP ADDRESS TEST
                --OFFSET BY

```

IT'S LIST ID

NEXT STARTING ADDRESS

SEQ 0168

NEXT BANK

GE 104

8.2.5 Port Arbitration Symmetry Test -

The master causes each CPU (making himself a slave when necessary) to access (with memory fetches of MOV R5,R5 instructions) the same location 64K times. Each slave executes out of the memory bank under test thereby increasing the probability of contention. While this access loop is executing each slave is counting tics of the line clock (KW11L) under interrupt control.

After each slave completes his access test he passes his tic count back to the master for analysis.

The master checks every bodies tic count against his own and if any differ by more than 2 tics he reports an error.

This is done as follows:

```
NOT PROGRAM SPACE      FOR EACH BANK
                        IF BANK IS ACCESSABLE &
                        ALL CPU'S RUN CONTENTION TEST
                        CHECK TIC COUNTER FOR BALANCE
                        NEXT BANK
```

8.2.6 I/O Priority Test -

This tests the ability of the memory port to give I/O higher priority than CPU accesses.

Since the difference is difficult to measure on small configurations this test is only run when there are 4 CPU's present.

The master causes each CPU (except one) to raise his priority to that of I/O. This is done by setting a margin of 6 in the maintenance register of the PDP11-70.

The master causes all CPU's to run the same assessment as the 'Port arbitration symmetry test'. In the symmetry test the master checks for a balance. In this test the master checks for an lack of balance.

This is done as follows:

```

        FOR EACH BANK
            IF BANK IS ACCESSABLE & NOT PROGRAM SPACE
                FOR EACH CPU BEING LOW PRIORITY
                    ALL CPU'S BUT ONE S
                ET FOR I/O PRIORITY
                    ALL CPU'S RUN CONTENTION TEST
                CHECK TIC COUNTER FOR LACK OF BALANCE
        NEXT CPU
    NEXT BANK
    
```

8.2.7 Execution Contention Test -

The master causes each CPU (Making himself a slave when necessary) to execute random instructions (INC (R0) or TST (R0)) simultaneously out of shared program space.

Success is indicated by survival & correct count in location pointed to by R0.

This is done as follows:

```

FOR EACH BANK
    IF BANK IS ACCESSABLE & NOT PROGRAM SPACE
        SETUP RANDOM INSTRUCTIONS
        ALL
CPU's EXECUTE IN BANK
    CHECK COUNTERS FOR ALL EQUAL
    NEXT BANK
  
```

8.2.8 ASRB Test -

In a multi-processor system, certain portions of main memory must be capable of locking out all other references during the DATIP-DATO cycle. This is accomplished through a hardware and software LOCK process

8.2.8.1 Software Lock -

The Arithmetic Shift Right Byte (ASRB) instruction is used to attempt ownership of a particular block of shared data and/or code which may include I/O routines, data dependent routines, and etc. The ASRB instruction changes the state, (the UNLOCKED state or free state is defined as a binary 1 and the LOCKED state is defined as a binary 0), of

the control byte's destination location associated with the data structure/code being accessed; if that location is in the UNLOCKed state. The C bit of the PSW will store the information (0 if LOCK, 1 if UNLOCK) resulting from data manipulation during the LOCK/UNLOCK routine. The ASRB instruction is followed by a BCS (Branch if Carry Bit is Set) instruction which will branch and execute the desired code. When finished the LOCK routine will move a 1 back into the low order bit of the control byte's destination location; indicating to the other processors that this memory block is UNLOCKed. The ASRB instruction maintains its identity as a normal shift instruction.

8.2.8.2 Hardware Lock -

The PDP11/70 multiprocessor modifications treat the ASRB as a DATIP-DATO cycle to main memory bypassing CACHE.

PAGE 107

8.2.8.3 ASRB Test Structure -

The master causes each CPU (making himself a slave when necessary) to attempt to lock location zero of a bank under test simultaneously. After each attempt the master insures that one and only one CPU was allowed to lock the test location. This test is repeated 1000 times to insure that all CPUs are able to lock the test location is relatively even distribution (+ or - 256. Locks of the master). To insure that the test does not synchronize with the memory arbitration scheme every time test is repeated the previous losers (non lockers) are given a head start in a chain of NOP instructions and the previous winner is defaulted to the entire chain.

This is done as follows:

```

FOR EACH BANK
  IF BANK IS ACCESSABLE & NOT P
    FOR ALL 8 POSSIBLE INTERLEAVE ADDRESSES
      FOR 1000 TRIES
        UNLOCK TEST
        ALL CPUS ATTEMPT LOCK
      RECORD WINNER
      REARRANGE PRIORITIES
      CHECK FOR ONE AND ONLY ONE WINNER
    NEXT TRY
    NEXT INTERLEAVE ADDRESS
  NEXT BANK
  CHECK FOR EVEN DISTRIBUTION

```

8.2.9 CACHE FLUSH Test -

The master causes each CPU (making himself a slave when necessary) to write fresh data into a double word and all other CPU's to simultaneously change the data in that double word - thereby creating stale data in the 1st CPU's CACHE. The 1st CPU then insures that stale data exists then initiates a CACHE Flush and insures that the data is no longer stale.

This is done as follows:

```

RATOR          IF CACHE IS NOT DISABLED BY THE OPERATOR
                FOR EACH CPU
                FOR EACH BANK
                    IF BANK IS ACCESSABLE & NOT PROGRAM SPACE
                        SELECTED CPU WRITES FRESH DATA
                        ALL OTHER CPU'S MAKE DATA STALE
                        STALE DATA
                        SELECTED CPU VERIFYS
                        SELECTED CPU DOES CACHE FLUSH
                        SELECTED CPU VERIFYS NON-STALE DATA
NEXT BANK      NEXT CPU

```


BANKS

I

TEST

PROTECTION

POINTER

I

TEST

SYSTEM SIZE

REGISTER

I

I

*

*
* TYPE *
* CONFIGURATION *

I

* LEVEL *

* RUN BANK *

TESTS

I

* RUN SYSTEM *

* LEVEL *

* TESTS *

I

* FLUSH *

* OUT *

* DBE'S *

I

* END OF *

PASS

* CODE *

9.1.1 Initialize -

```

*****
/      SETUP      /
/      STACK      /
TER /              / POIN
*****
I
*****
CLEAR *
*      ZERO      *
*      VARIABLES *
*****
I
*****
*      CLEAR ANY NON *
*      PROGRAM SPACE *
*      IN BANK 0      *
*****
I
*****
*      SET UP      *
*      NON-ZERO    *
*      VARIABLES   *
*****
I
*****
SETUP *
*      VECTORS    *

```

RN *

I

*
* SETUP PATTE
* SCRIPTS FROM *
* APT 'E-TABLE' *

I

* CLEAR *
* CONFIGURATION *
* TABLE *

I

* SETUP, SWITCH *
* REG TYPE *
OR SOFT) * (HARD

I

I

* SETUP FOR *

* APT, ACT, *

* XXDP *

I

PRINT

TITLE

I

*

* GO TO NEXT TEST *

9.1.2 Test Hardcore Hardware -

```

*****
* DETERMINES *
R *
* MK11 CS
* ACCESS METHOD *
*****
I
***
*****
* PROJECT *
* BANK 0 *
*****
I
*****
* MAP *
* KERNEL *
* SUPERVISOR *
*****
I
*****
* SETUP SUPER- *
* VISOR STACK *
* POINTER *
*****
I
*****
* SETUP USER *
*
STACK *
* POINTER *
*****
I

```


9.1.3 Analyze Configuration -

```

*****
/ SELEC
T / / 1ST /
/ CSR /
*****
-I I<-----
-----
NO I / DOES L
I CSR I
L I / I
I EXIST? L / I
I ----- I
I I YES
I I *****
I I * SET BIT *
I I * I I
N I * I
I * 'MKCSRS' * I
I *****
I I I
I I I ----- YES
I I / INTERNAL I
I L----->I I
/ I INTERLEAVE L I I
I / ? L *****
I I ----- * *
I I I NO I * SE
I BIT * I I
I * IN 'MKCSRS' * I

```

```

I
I
I *****
I I<-----I
I
I *****
I I
I *
I I SAVE *
I * INTER I
LEAVE *
I *****
I
I I
I I
I I / L
I I / CALL I
I L / I
I GETSIZE L I
I
I I
I I *****
I I SELECT I
I *
I * I
I 1ST BANK * I
I * OF CSR * I
I
I *****
I I<-----I
I
I *****
I I UPDATE I
I CONFIG * I
I * TABLE OF BANK * I
I * W/INTERLEAVE,MK11 ID* I
I
I *****
I I
I I I
I I I
I I

```


9.1.4 Size & Test 0's & 1's On Each Bank -

```

      I
*****
/ NO * ENABLE * YES /
      * ENABLE CSR *
      * CSR FOR * <-----/
SWO  /-----> * FOR DBE *
      * SBE TRAPS *
      * SET / * TRAPS *
*****
      I
      I
      I-----
-----I
      I
      V
*****
      * SETUP TRAPS *
      * TO COUNT *
      * ERRORS *
*****
      I
*****
      * CLEAR *
      * ERROR *
      * COUNTS *
*****
      I
      *
      * CACHE *
      TURN

```

* OFF *

I

* DO 'TST' *

* INSTRUCTION *

* ON BANK 0 *

I

* TURN *

* CACHE *

* ON *

I

YES

/ ANY /

/ FATAL ERRO /

/ TRAP /

I

I NO

.....
ESSED *

* SET ACC
* BIT FOR *
* BANK 0 *

.....

I

* CLEAR *

* 'BANK', *

* 'LASTBANK' *

.....

I

* SETUP TRAPS *

* TO COUNT *

ERRORS *

I

* MOVE MODIFIED *

0 TO *

* PATTERN
* FASTCITY *

I

A

DEFAULT

I

MAP

* SUPERVISOR

I

* TO BANK *
/ GO TO L

/ NEXT L

/ TEST L
I

* MODIFY PATTERN *
* TO STOP *

* AFTER WRITE *

I

* SETUP

* FOR *
* PATTERN 0 *

I

```

I
*****
*   TURN   *
*   CACHE  *
*   OFF    *
*****

```

```

I
*****
*   ENTER  *
*   SUPERVISOR *
*   MODE   *
*****

```

```

I
*****
*   WRITE  *
*   ZEROS  *
*****

```

```

I
*****
*   EN    *
*   TER  *
*   KERNEL *
*   TURN CACHE *
*   ON    *
*****

```

```

I
-----
/   ANY   L----- YES
-----
/   NON-EXISTENT L
/   I MEM TRAPS? L
-----

```

```

I
-----
THIS THE I NO
L
I
-----

```

```

*****
*   I TURN *
*   I CACHE *
*   I      *

```

```

V
I
/   IS
/   1ST MISSING L---
/   BANK L
-----
I

```



```

I
*****
* ENTER
*   MODE   *
*   TURN CACH
*   ON     *
*****

```

```

I
-----
/ / / ANY L YES
L-----
/ / ERRORS? L
I
-----

```

```

I
I NO
*****
*   SETUP   *
* FOR *
*   PATT   *
*****

```

```

I
I
*****
*   UNMODIFY   *
*   PATTERN   *
*****

```

```

I
*****
*   ENTER   *
* SUPERVISOR *
*   MODE   *
*****

```

ERN 0 *

GO TO "A"

.....

9.1.5 Determine Protected Banks of MK11 -

```

*****
**          /      TURN      /
/  CACHE   /      /      OFF  /
*****
          I
*****
MK11 *      * CLEAR
      *      * PROTECTED *
      * FLAG  *
*****
          I
*****
BANK *      * SET
      *      *
      *      * TO 0 *
*****
          I
-----I----->I
I
I      I      I
I
I      -----
K I L      NO      / IS THIS BAN
I      / ACCESSIBLE BY L----->
-----I-----
I      / CPU?      L
          I
-----
I      I      I
- I      I      I
YES I      -      I
-----
I      / IS THIS      L NO      I

```


9.1
.6 Test MK11 Protection Pointer -

```

*****
ET / / S
/ / BANK /
/ TO 0 /
*****
I<-----
*****
I * I NO *
* CALL * -----
/ BANK * L EXBANK YES *
***** / GRE
ATER THAN L-----
I / 167? L I
----- I
/ I L NO
I / IS T I
HIS BANK MK11 L----- I
I / I L
I ----- I
I I V I
I YES I NO
I I ----- I
/ IS THE BANK L I
*****
ACCESSIBLE L----->*

```

* INCREMENT BANK * BY CPU? L I

I YES

/ IS THE KPFLAG L

I

/ SET? L NO

L NO

L-----

I

----->*****

I

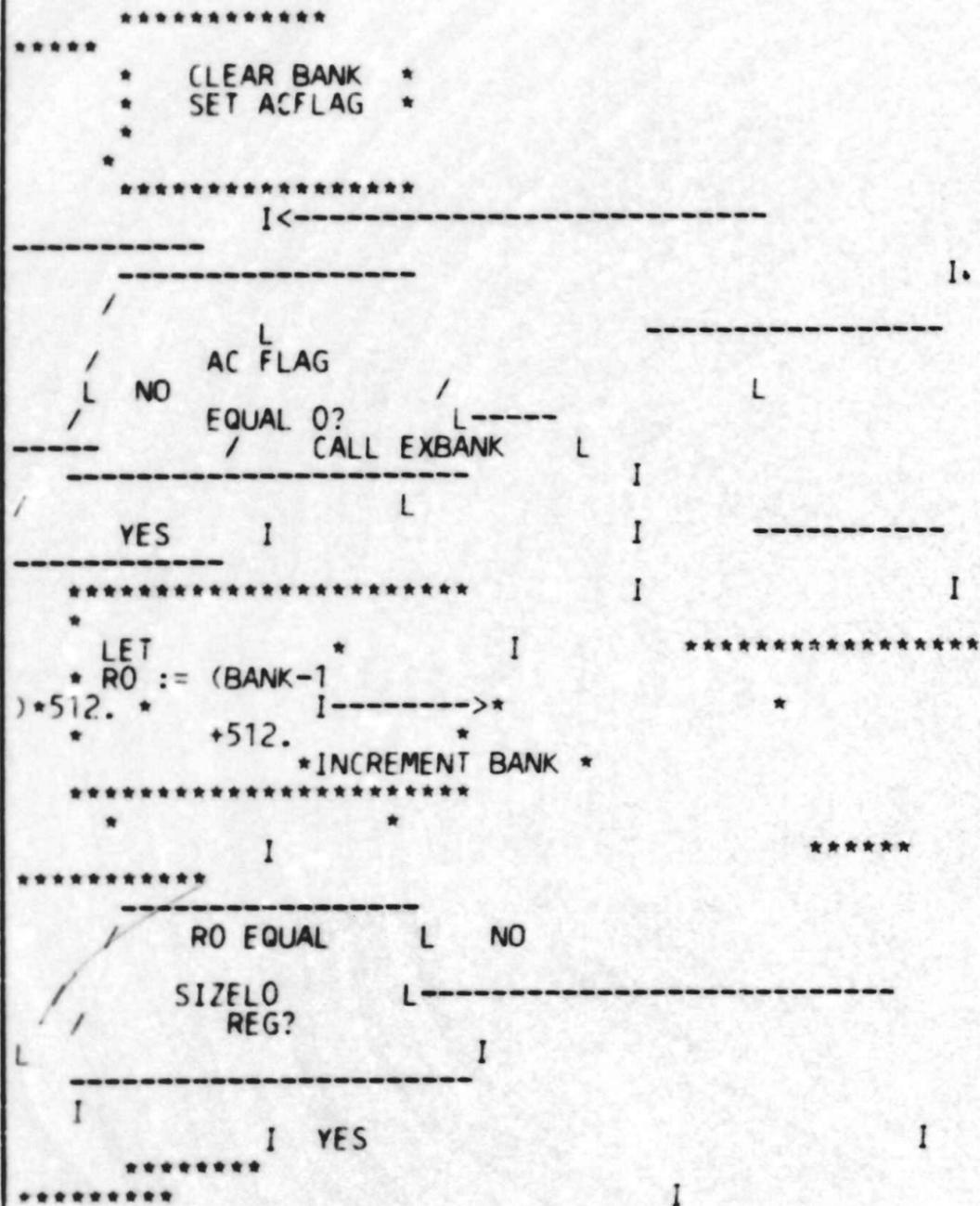
I YES

* GO TO * <-----]

NEXT TEST *

* ERROR TRAP *

9.1.7 Test System Size Register -



•
• I
• GO TO •

• NEXT TEST •
•••••

•••••

• SETUP •

• ERROR INFO •

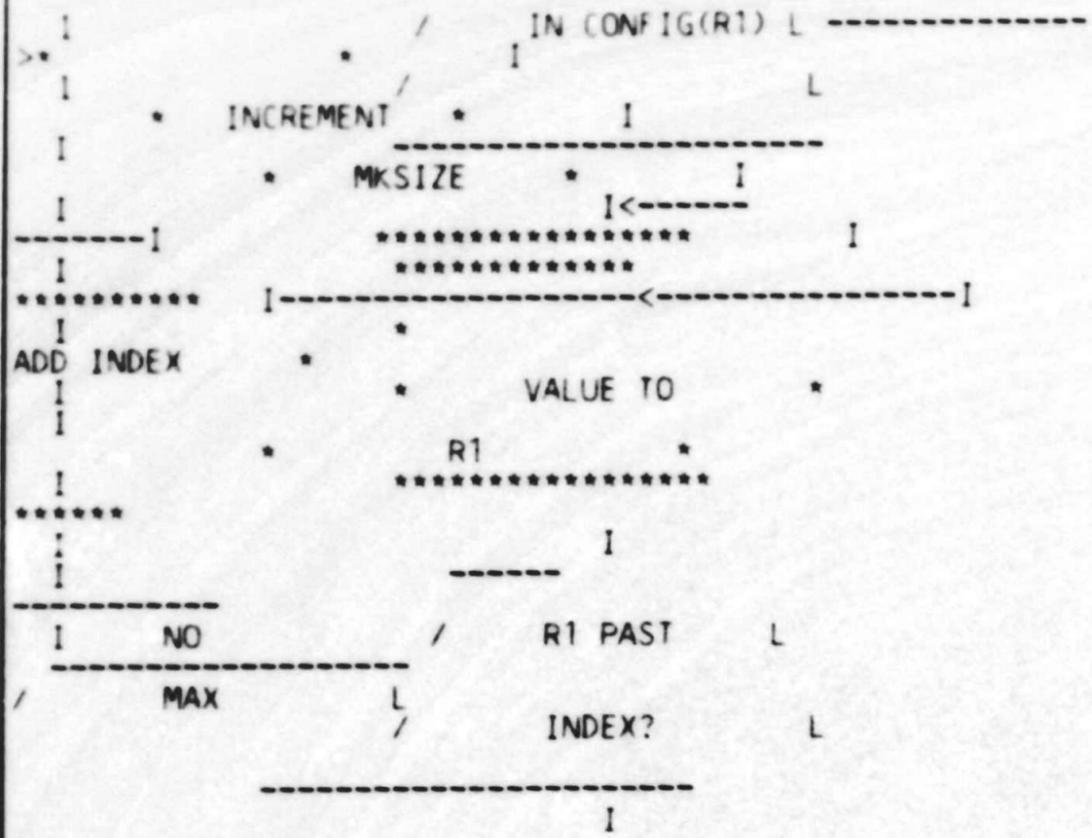
•••••

I

•••••

• ERROR TRAP •

•••••



```

      I
      *****
      *
      *Mysize:=Mysize *
16. *
      *
      *****
      I
      *****
      *
      *Mysize:=Mysize * 16. *
      *
      *****
      I
      *****
      *
      *
      I:=Mysize + Mysize
      *
      *****
      I
      *****
      *
      PRINT
      * I; 'K OF MEMORY' *
      *
      *****
      I
      *****
      *
      PRINT
      * Mysize;'K OF MJ11' *
      *
      *****
      I
      *****
      *
      PRINT
      * Mysize;'K OF MK1
1'' *

```

.....
I

/ L
/ GO TO NEXT TEST L
/

L

9.1.9 Run Bank Level Tests -

```

*****
/      CLEAR      /
/      CONTFLAG   /
/                /
*****
I
*****
*      GET TEST   *
*      MODE FROM  *
*      SWR        *
*****
I
-----
I
I
DISPATCH BY TEST MODE (0 - 7)
I
-----0-----
-1-----2-----3-----4-----5-----6-----
-7-----
I      I      I      I      I      I      I
*****
** I ***** I ***** I *****
* I CALL * * I * * I *
* I BAFFAF * I CALL * I CALL
* I * I CALL * I BAWPAF *
* I * I *

```

```

I * PAFBAF * I * PARBAF * I
***** I ***** I *****
I I I I I I I I I
I I I I I I I I I
I I I I I I I I I
I ***** I *****
I ***** I *****
I * I * I * I *
I * CALL * I * CALL * I * CALL *
I * CALL * I * CALL * I * CALL *
I * PAFPAR * I * BAWPAR * I *
PAFBAW * I * PARBAW * I *
***** I ***** I *****
I I I I I I I I I
I I I I I I I I I
I I I I I I I I I
I I I I I I I I I

```

```

-----
I
I
V
*****
*
*
* GO TO *
* NEXT TEST *
*****
*****

```

9.1.10 Run System Level Tests -

```
*****  
/ /  
/ CALL MT0027 /  
/(UNIQUE BANK TEST)/  
*****
```

```
*****  
I  
*****  
*  
TUP RO FOR *  
DDPAR CALL *  
*****
```

```
****  
*  
* SE  
* A  
****
```

```
I  
*  
*  
*  
*  
*  
*
```

```
*****  
* SETUP PARITY  
* ACTION TO  
* COUNT ERRORS  
*****
```

```
I  
*****  
* MOVE ADDPAR *  
* SUBROUTINE TO *  
* FASTCITY *  
*****
```

```
I  
*****  
*  
*****
```

* CLEAR BANK *

G 1

SEQ 0213

* * *

I

I

I YES
I NO

I

* RESET PARITY *

* ACTION TO *
ERROR L

* DEFAULT *

/ TRAP

L

I

/ L

/ CALL MTO030 L

/ (

FLUSH OUT DBE'S) L

I

/ GO TO L

/ NEXT L

/ TEST

L

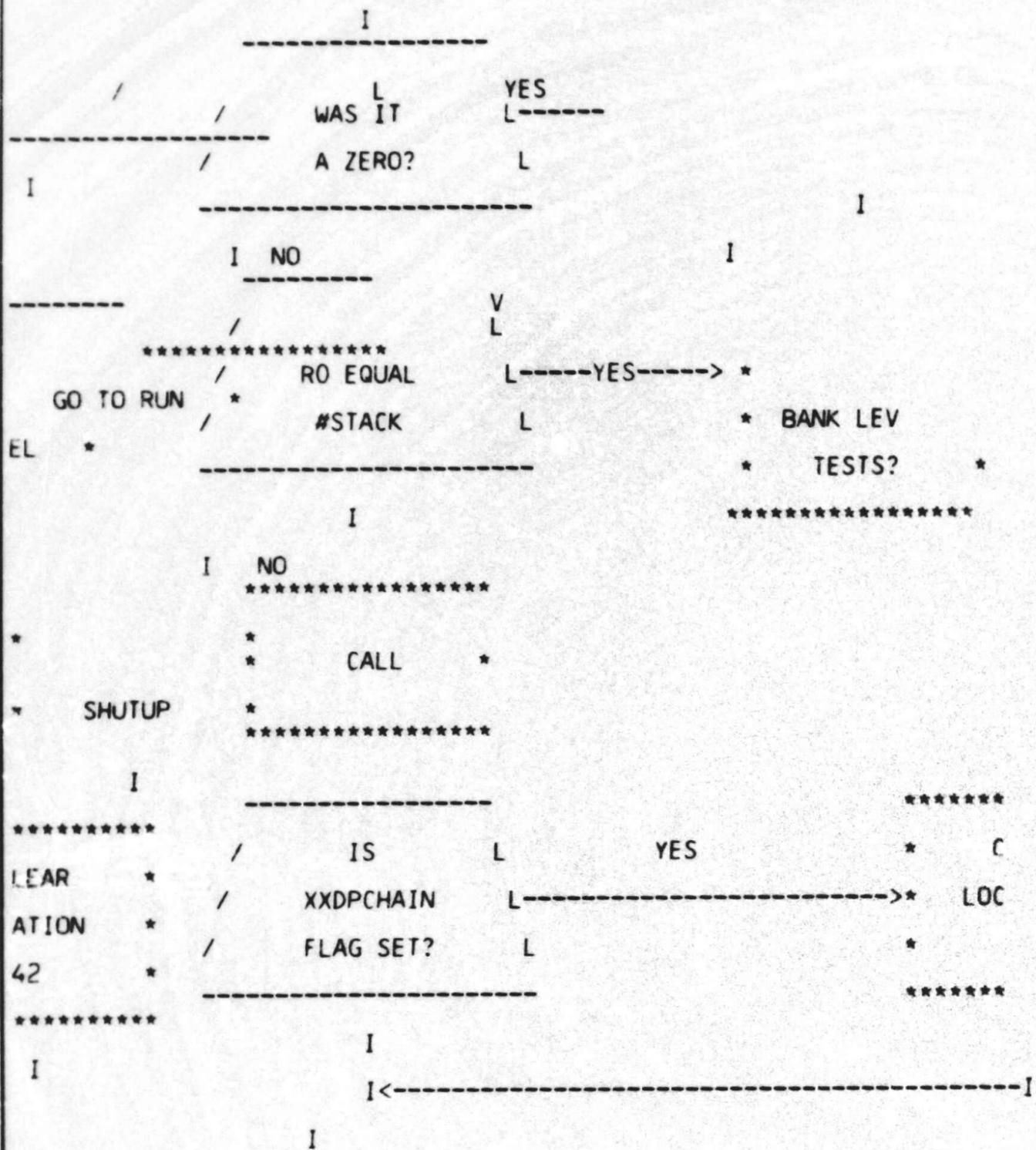
-----I
V

* PRINT \$PASS *
* *****

I

* * MOVE CONTENTS *
* OF 42 TO R0 *

I



```

      I
*****
*
*   RESET   *
*
*****

```

```

*****
      I
      I
-----
NOTE:
THE FOLLOWING CODE IS MODIFIED BY ACT THEREFORE
MAY APPEAR AS 3 TYPES
OF ACTION

```

XXDP CHAIN

ACT QV

ACT AUTO

```

-----
      I           I           I           I           I
      I           I           I           I           I
      I           I           I           I           I
      I           I           I           I           I
      I           I           I           I           I
*****
*
*   CALL RO  *   *   RESET   *
*   *   *   *   *   *   *
*   *   *   *   *   *   *
*   *   *   *   *   *   *
*****
*****

```

```

      I           I           I<-----
*****
*
*   CLEAR IFLAG *   I   *
*
*
*****
      I           I
-----
*****
*
*

```

N 1
* NOP *

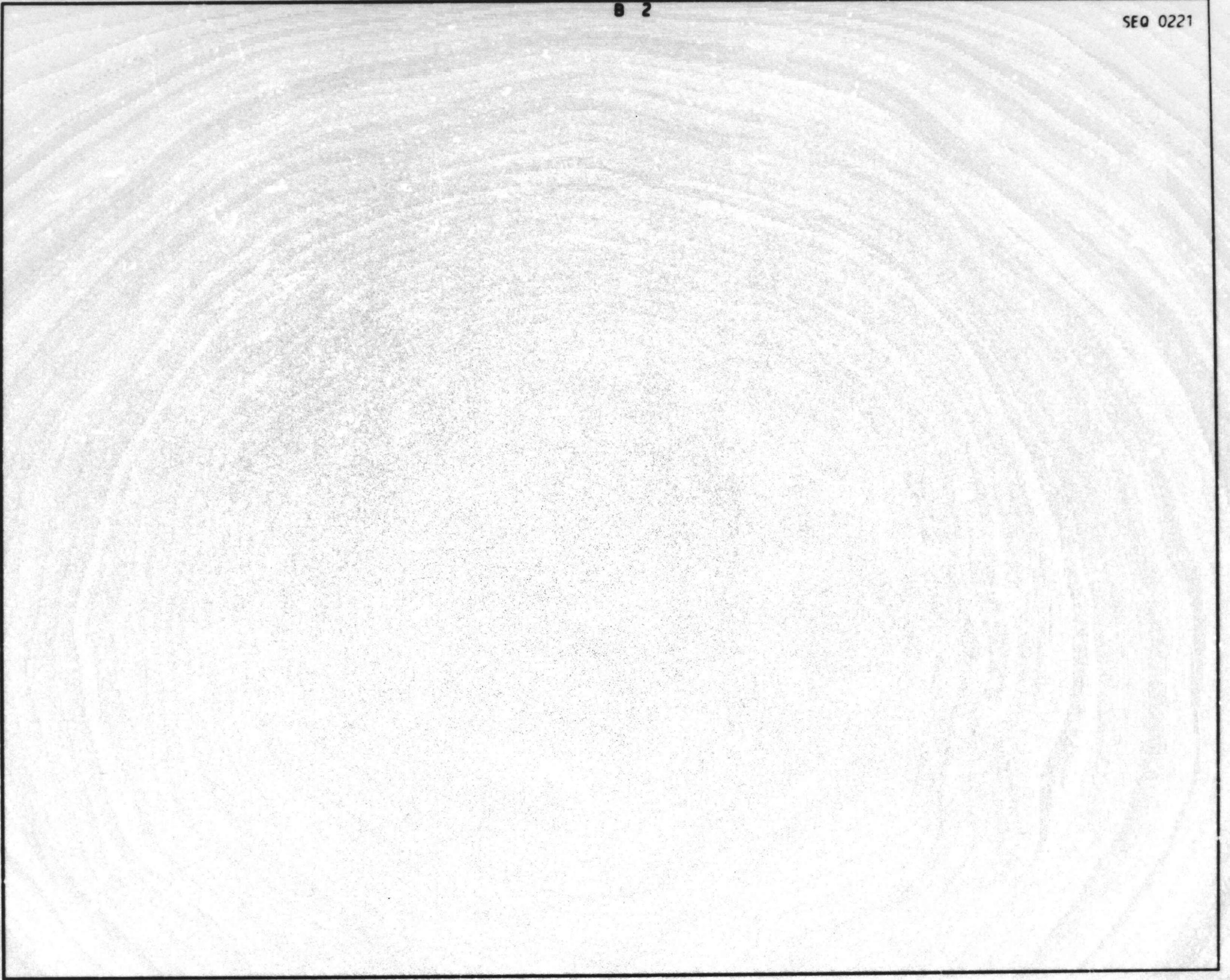
SEQ 0220

I

* GO TO RUN *

* BANK LEVEL *

* TESTS *



5-	90	OPERATIONAL SWITCH SETTINGS	
5-	91	;SWITCH REGISTER DEFINITIONS	
5-	92	.*	
5-	93	.*	
5-	94	.*	
5-	95	.*	
5-	96	.*	
5-	97	.*	
5-	98	.*	
5-	99	.*	
5-	100	.*	
5-	101	.*	
5-	102	.*	
5-	103	.*	
5-	104	.*	
5-	105	.*	
5-	106	.*	
5-	107	.*	
5-	108	.*	
5-	109	.*	
5-	110	.*	
9-	136	DEFINE TRAPS	
11-	252	DEFINE BASIC PDP11 STUFF	
11-	334	DEFINE CACHE REGISTERS	
11-	342	DEFINE CPU REGISTERS	
11-	350	DEFINE MEMORY MANAGEMENT REGISTERS	
13-	480	DEFINE UNIBUS MAP REGISTERS	
13-	548	DEFINE SOFTWARE SWITCH & DISPLAY REGISTERS	
13-	552	DEFINE MK11 REGISTERS	
13-	555	DEFINE PARAMETERS	
13-	560	DEFINE MASTER & SLAVE BITS	
15-	568	MACRO FATAL	
15-	588	MACRO TYPE	
17-	606	MACRO NEWTST	
19-	656	MACRO \$\$NEWTEST	
19-	677	MACRO SUBTST	
19-	696	MACRO \$\$SUBTST	
21-	711	MACRO TYPOCT	
23-	751	MACRO TYPOCS	
25-	807	MACRO TYPDEC	
26-	849	MACRO BMOV	
28-	915	MACRO MAP	
30-	954	MACRO SUPERVISOR	
30-	975	MACRO USER	
32-	997	MACRO SET4 & RES4	
34-	1039	MACRO DLEFT	
36-	1063	TRAP CATCHER	
36-	1071	ACT11 HOOKS	
36-	1090	APT11 HOOKS	
38-	1111	VARIABLES INITIALIZED TO ZERO	
40-	1294	MULTIPOINT VARIABLES	
42-	1314	VARIABLES INITIALIZED TO NON ZERO	
44-	1361	CONFIGURATION TABLE	
45-	1402	***** MAIN *****	
45-	1403	INITIALIZE VARIABLES TO ZERO	
45-	1420	CLEAR NON-PROGRAM SPACE	
47-	1443	INITIALIZE VARIABLES TO NON ZERO	

47-	1456	INITIALIZE VECTORS	
50-	1479	CLEAR THE CONFIGURATION TABLE	
50-	1491	SIZE FOR A HARDWARE SWITCH REGISTER	
50-	1508	FIND OUT IF ANY LINE CLOCK IS PRESENT	
52-	1516	SETUP ACT, APT, & XXDP	
54-	1542	INITIALIZE PATTERNS	
54-	1567	SUBR PLUG IN NULL PATTERNS	
56-	1578	PROTECT PROGRAM & LOADERS	
56-	1584	MULTIPOINT DETERMINATION ROUTINE	
58-	1618	SLAVES START HERE	
60-	1630	DETERMINE MK11 ACCESS METHOD	
60-	1647	SETUP USER & SUPERVISOR STACK	
62-	1664	GET SOFTWARE SWITCH REGISTER IF NECESSARY	
62-	1674	SETUP UNIBUS MAP	
62-	1681	GET MEMORY MANAGEMENT READY	
64-	1689	T1 ANALYZE CONFIGURATION	
65-	1796	T2 TEST BANK 0 ACCESSES	
65-	1825	ENABLE MK11 FOR CORRECT TRAPS	
67-	1833	T3 TEST BANKS 1-167 (OCTAL) FOR ZEROS & ONES	
69-	1937	DETERMINE PROTECTED BANKS	
71-	2031	T4 MK11 PROTECTION POINTER TEST	
73-	2046	CHECK SYSTEM SIZE REGISTER	
81-	2201	MOVE LOADERS IF NECESSARY	
83-	2242	MULTIPOINT HOOK	
85-	2305	SLAVES WAIT FOR MASTER TO COUNT DOWN	
87-	2319	BOOT UP ALL EXISTING CPU'S	
92-	2512	CHECK FOR UNIQUE CPU ID NUMBERS	
92-	2532	PRINT MULTIPOINT DIRECTORY	
92-	2546	LOAD CSR'S FROM E-TABLE \$CDW1 & \$CDW2	
94-	2562	LEGAL CONFIGURATION CHECK	
96-	2626	PRINT CONFIGURATION DETAILS	
98-	2664	CHECK APT SIZING	
99-	2710	T5 DIAGNOSTIC MODE DISPATCH ROUTINE	
102-	2730	T6 UNIQUE BANK TEST	
102-	2740	T7 FORCE ADDRESS PARITY ERRORS	
104-	2784	FLUSH OUT DBE'S	
107-	2789	END OF PASS ROUTINE	
109-	2843	WRITE BACKGROUND PATTERNS	
111-	2857	MTEST MODES	
111-	2859	BANKS FORWARD,PATTERNS FORWARD	**RECURSIVE**
113-	2893	BANKS FORWARD,PATTERNS REVERSE	**RECURSIVE**
115-	2927	BANKS WORST FIRST,PATTERNS FORWARD	**RECURSIVE**
117-	2968	BANKS WORST FIRST,PATTERNS REVERSE	**RECURSIVE**
120-	3010	PATTERNS FORWARD,BANKS FORWARD	**RECURSIVE**
122-	3052	PATTERNS FORWARD,BANKS WORST FIRST	**RECURSIVE**
124-	3101	PATTERNS REVERSE,BANKS FORWARD	**RECURSIVE**
126-	3143	PATTERNS REVERSE,BANKS WORST FIRST	**RECURSIVE**
129-	3193	SUBR SETUP MEMORY TEST	
131-	3214	SUBR TEST MK11 CONTROLLER LOGIC DISPATCH	
133-	3284	IS ECC ON?	
135-	3306	CHECK FOR SBE FREE LOCATIONS	
137-	3384	SETUP INTERLEAVE INFO	
139-	3421	SUBR GET SIZE FROM CSR	
142-	3448	CSR PATTERN CASE STATEMENT	
144-	3490	DETERMINE ADDRESSING ACCORDING TO INTERLEAVE	
146-	3624	DETERMINE IF BANK BELONGS TO CORRECT SIDE	
149-	3653	SUBR MK11 TEST DISPATCH	

151-	3724	SUBR	MJ11 TEST DISPATCH
152-	3778		PATTERNS
152-	3780	MEMORY	TEST SETUP ROUTINES
152-	3781	MT0000	SETUP DATA PATTERN TEST
152-	3789	MT0001	SETUP ADDRESS TEST
152-	3798	MT0002	SETUP COMPLEMENT ADDRESS TEST
154-	3811	MT0003	SETUP 3 XOR 9 WORST CASE NOISE TEST
154-	3836	MT0004	SETUP ROTATING ZEROS TEST
154-	3847	MT0005	SETUP ROTATING ONES TEST
156-	3860	MT0006	SETUP INITIAL DATA TEST
156-	3867	MT0007	SETUP ADDRESS BIT TEST
156-	3877	MT0010	SETUP BYTE ADDRESSING TEST
158-	3886	MT0011	SETUP CREATE SINGLE BIT ERROR TEST
158-	3895	MT0012	SETUP WRITE BYTE CLEARS SBE TEST
158-	3904	MT0013	SETUP CREATE DOUBLE BIT ERROR TEST
158-	3913	MT0014	SETUP WRITE INHIBIT DURING DATIP WITH DBE
160-	3924	MT0015	SETUP WRITE INHIBIT OF BYTE WITH DBE
160-	3933	MT0016	SETUP WRITE INHIBIT OF WORD WITH DBE
160-	3942	MT0017	SETUP HOLDING 1'S & 0'S
162-	3949	MT0020	SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST
164-	4021	MT0021	SETUP MARCHING 0'S & 1'S TEST
164-	4046	MT0022	SETUP REFRESH & SHIFTING DIAGONAL TEST
164-	4054	MT0023	SHIFTING DIAGONAL TEST
165-	4064	MT0024	SETUP FAST GALLOPING PATTERN TEST
165-	4090	MT0025	SETUP INTERRUPT ENABLE TEST
167-	4100	MT0026	SETUP RANDOM DATA TEST
169-	4123	MT0027	UNIQUE BANK TEST
171-	4179	MT0030	SETUP FLUSH OUT DBE'S TEST
173-	4218	MT0031	SETUP SOB-A-LONG TEST
175-	4238	MT0033	SETUP WRITE RECOVERY TEST
177-	4276	MT0034	SETUP BRANCH GOBBLE TEST
177-	4297	MT0035	SOFT ERROR - BACKGROUND PATTERN TEST
179-	4320	MT0999	SETUP NULL TEST
179-	4325		CHECK FOR KAMIKAZE MODE
182-	4334	SUBR	EXECUTE PATTERN IN SUPERVISOR
188-	4403	MEMORY	TEST PATTERN ROUTINES
188-	4413	MTP000	BASIC DATA TEST
188-	4424	MTP001	ADDRESS TEST
188-	4436	MTP002	COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
191-	4451	MTPA03	3 XOR 9 WORST CASE NOISE TEST (WRITE)
191-	4476	MTPB03	3 XOR 9 WORST CASE NOISE TEST (READ)
193-	4494	MTPC03	TEST DATA SUBPROGRAM
193-	4503	MTPD03	TEST DATA SUBSUBPROGRAM
195-	4513	MTPA04	ROTATING ZEROS TEST
195-	4526	MTPB04	SUBR ROTATING BIT
195-	4535	MTP005	ROTATION ONES TEST
198-	4550	MTP006	INITIAL DATA TEST
201-	4594	MTP007	ADDRESS BIT TEST
203-	4634	MTP010	BYTE ADDRESSING TEST
206-	4671	MPT011	SINGLE BIT ERROR TEST
209-	4782	MTP012	WRITE BYTE CLEARS SBE TEST
212-	4860	MTP013	CREATE DOUBLE BIT ERROR TEST
217-	4929	MTP014	WRITE INHIBIT DURING DATIP WITH DBE TEST
220-	5010	MTP015	WRITE INHIBIT OF BYTE WITH DBE
223-	5087	MTP016	WRITE INHIBIT OF WORD WITH DBE
228-	5167	MTP017	HOLDING 1'S & 0'S TEST
231-	5201	MTP020	MARCHING 1'S & 0'S IN CHECK BITS TEST

236-	5279	MTPA20	SLOW MARCHING 1'S & 0'S IN CHECK BITS TEST
243-	5416	MTPA21	MARCHING 1'S & 0'S PATTERN TEST
248-	5487	MTP022	REFRESH & SHIFTING DIAGONAL TEST
250-	5559	SUBR	REFRESH DELAY
253-	5582	MTPA24	FAST GALLOPING PATTERN TEST
255-	5626	MTPB24	FAST GALLOP PART B
255-	5634	MTPC24	FAST GALLOP PART C
258-	5645	MTP025	INTERRUPT ENABLE TEST
263-	5738	MTPA26	RANDOM DATA (WRITE)
263-	5745	MTPB26	RANDOM DATA (READ)
263-	5763	RANDOM	NUMBER SUBPROGRAM
263-	5776	RANDOM	NUMBER SUBSUBPROGRAM
265-	5784	MT0030	FLUSH OUT DBE'S
265-	5790	MTP031	SOB-A-LONG TEST
267-	5816	MTPA32	SKEWED UP ADDRESS TEST (WRITE)
267-	5826	MTPB32	SKEWED UP ADDRESS TEST (READ)
267-	5839	MTP033	WRITE RECOVERY TEST
269-	5859	MTP034	BRANCH GOBBLE TEST
270-	5905		MISC SUBROUTINES
270-	5907	SUBR	COPY R0 TO R4, R1 TO R3, & R2 TO R5
270-	5913	FLIP	WARNING CONSTANTS IN WORST CASE NOISE TESTS
270-	5940	SUBR	WRITE BACKGROUND
273-	5953	SUBR	PRINT CONFIGURATION MAP
276-	6002	SUBR	TYPE CONFIGURATION
281-	6154	TRAP	PARITY ERROR HANDLER
283-	6200	TRAP	NON-EXISTANT MEMORY (HOLES) HANDLER
283-	6222	TRAP	TIMEOUT (TRAP TO 4) HANDLER
283-	6226	TRAP	MEMORY MANAGEMENT (TRAP TO 250) HANDLER
283-	6229	TRAP	RESERVED INSTRUCTION HANDLER
283-	6233		UNEXPECTED IIST TRAP HANDLER
283-	6237	FIND	BAD SP, PC, & PSW FROM STACK
285-	6245	TRAP	KERNEL TRAP HANDLER
285-	6253	TRAP	ENERGIZE TRAP HANDLER
285-	6257	TRAP	DEENERGIZE TRAP HANDLER
285-	6261	TRAP	CACHON TRAP HANDLER
285-	6266	TRAP	CACHOFF TRAP HANDLER
288-	6273	TRAP	LOAD CSR TRAP HANDLER
290-	6302	TRAP	READ CSR TRAP HANDLER
290-	6321	TRAP	TEST (R1) & READ CSR CAREFULLY
293-	6354	TRAP	ECC DISABLE ALL CSR'S TRAP HANDLER
293-	6360	TRAP	ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
293-	6366	TRAP	MK11 INITIALIZE ALL CSR'S TRAP HANDLER
293-	6371	TRAP	MK11 INITIALIZE 1 SELECTED CSR TRAP HANDLER
293-	6376	TRAP	ENABLE SBE PARITY TRAPS ON ALL CSR'S
293-	6382	TRAP	ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
293-	6388	TRAP	WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
293-	6395	TRAP	WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
296-	6405	TRAP	WAS THERE A SBE ON ANY CSR TRAP HANDLER
296-	6430	TRAP	WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
298-	6440	TRAP	WAS THERE A DBE ON ANY CSR TRAP HANDLER
298-	6465	TRAP	WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
300-	6479	TRAP	CLEAR ALL MK11 CSR'S TRAP HANDLER
300-	6483	TRAP	CLEAR 1 SELECTED MK11 CSR TRAP HANDLER
300-	6487	TRAP	ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
300-	6494	TRAP	ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
302-	6503	SUBR	WRITE IN ALL CSR'S
302-	6518	TRAP	INVALIDATE BACKGROUND PATTERN

305-	6528	SUBR	GENERATE CHECK BITS
310-	6595	SUBR	MAPPER
310-	6627	TRAP	MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
313-	6649		RELOCATE PROGRAM
317-	6718		UNRELOCATE PROGRAM
317-	6754		SETUP LOWER 16K OF UNIBUS MAP
319-	6767		MOVE BANKS
321-	6815	SUBR	MAP USER TO NEW BANK
321-	6835	SUBR	SETUP KERNEL PAR'S FOR NEW BANK
321-	6848	SUBR	SETUP KERNEL PAR'S FOR NEW LOADER BANK
324-	6861	SUBR	EXAMINE BANK
327-	6918	SUBR	BANK OK?
327-	6929	SUBR	INCREMENT PATTERN TESTING MK11'S
327-	6937	SUBR	SET HIGHEST PATTERN TESTING TYPE
327-	6941	SUBR	INCREMENT BANK & TEST
329-	6948	SUBR	INCREMENT MARGINS & TEST
329-	6976		DECREMENT MARGINS & TEST
331-	6989	SUBR	SET HIGHEST MARGIN
331-	7006	SUBR	REWRITE CORE HISTORY
333-	7024		BOOTSTRAP ROUTINE
335-	7053		HALT PROGRAM
335-	7062		SHUTDOWN DIAGNOSTIC
335-	7085		APT SHUTDOWN SEQUENCE
337-	7095		BLOCK MOVE SUBROUTINE
338-	7122		FIELD SERVICE MODE
338-	7124	SUBR	FIELD SERVICE COMMAND MODE
340-	7169	COMMAND 0	EXIT
340-	7187	FS	COMMAND 1 READ CSR
342-	7202	FS	COMMAND 2 LOAD CSR
344-	7231	FS	COMMAND 3 EXAMINE MEMORY
346-	7270	FS	COMMAND 4 MODIFY MEMORY
348-	7322	FS	COMMAND 5 SELECT BANK, MARGIN, & PATTERN
348-	7440	FS	COMMAND 6 TYPE CONFIGURATION MAP
351-	7447	FS	COMMAND 7 BATTERY BACKUP TEST
353-	7496	FS	COMMAND 8 SOB-A-LONG TEST
355-	7550	FS	COMMAND 9 SUPER TIGHT SCOPE LOOP
355-	7618	FS	COMMAND 10 ERROR SUMMARY
357-	7645	FS	COMMAND 11 REFRESH TEST
359-	7700	FS	COMMAND 12 SET FILL COUNT
359-	7710	FS	COMMAND 13 ENTER KAMIKAZE MODE
359-	7715	FS	COMMAND 14 EXIT KAMIKAZE MODE
359-	7721	FS	COMMAND 15 TURN CACHE OFF
359-	7727	FS	COMMAND 16 TURN CACHE ON
359-	7733	FS	COMMAND 17 RUN ONLY MULTIPOINT TESTS
361-	7739	FS	COMMAND 18 RESUME RUNNING BOTH SINGLE & MULTIPOINT TESTS
361-	7744	FS	COMMAND 19 TEST ONLY SELECTED BANKS
361-	7763	FS	COMMAND 20 RESUME TESTING ALL BANKS
363-	7775	SUBR	DETERMINE CORRECT CSR
365-	3		MULTIPROCESSOR SLAVE DISPATCHER
367-	66		MOVE PROGRAM TO NEW BANK
367-	80		WRITE ONLY PATTERN IN HI BYTE
369-	100		READ ONLY PATTERN IN HI BYTE
369-	130		SLAVE PATTERNS
371-	145		EXECUTION CONTENTION TEST
373-	164		PORT ARBITRATION SYMMETRY TEST
375-	219		STOP SLAVE PROGRAM
375-	226		WRITE ONLY ADDRESS PATTERN IN HI BYTE

375-	230	READ ONLY ADDRESS PATTERN IN HI BYTE
377-	262	RUN FIVE PATTERNS IN DESIGNATED 1/4 BANK
377-	279	WRITE & READ A SKEWED UP ADDRESS TEST
379-	301	SETUP FOR ASRB TEST
381-	338	RECOVER FROM TASK13 - TRAP FROM IIST
383-	356	RESET FROM ASRB TEST
383-	361	RUN SLAVE IN CSR TESTS
383-	365	SET MARGIN 6, CPU HAS I/O PRIORITY
383-	370	CREATE STALE DATA TEST
385-	385	CACHE FLUSH TEST
385-	422	CACHE FLUSH TEST (FOR PAR'S)
387-	434	SETUP EXECUTION CONTENTION TEST
389-	458	MULTIPORT ASRB TEST (IN SUPERVISOR SPACE)
391-	541	BALANCE TEST
393-	568	ERROR DATA (SUPERVISOR) SETUP STUFF
393-	582	DATA WAS 3 WORDS
395-	614	GET DATA FROM ABORTED AREA IF POSSIBLE
397-	634	POWER FAIL AUTO RESTART
397-	635	ROUTINE POWER DOWN AND UP
403-	788	MULTIPLE CPU'S POWERING UP
403-	795	POWER FAIL WHILE RELOCATED
405-	820	POWER UP FROM BANK 0 TO RELOCATION
407-	858	IO SUBROUTINES
407-	860	ROUTINE TYPE
408-	949	MULTIPORT SLAVE STARTER
410-	1032	HAVE SLAVES RUN CSR TESTS
412-	1064	MOVE SLAVES TO THERE OWN BANKS
414-	1111	RUN NORMAL MULTIPROCESSOR TESTS
416-	1187	INITIALIZE THE IIST
416-	1207	INIT IIST & DISABLE BOOTS & INTERRUPTS
418-	1221	SLAVES MAP
420-	1241	MULTIPORT ACCESS PATHS TEST
423-	1306	MULTIPORT ADDRESS TEST
425-	1369	MULTIPORT ASYNCHRONOUS ADDRESS UNIQUENESS TEST
427-	1417	MULTIPORT SKEWED UP ADDRESS TEST
429-	1458	MULTIPORT PORT ARBITRATION SYMMETRY TEST
431-	1506	MULTIPORT PORT I/O PRIORITY TEST
435-	1575	CHECK FOR PROPER SKEWING OF ACCESSES
435-	1600	WAIT FOR 5 SECONDS FOR A SLAVE
437-	1615	MULTIPORT EXECUTION CONTENTION TEST
437-	1662	MULTIPORT ASRB TEST
439-	1666	MULTIPORT CACHE FLUSH TEST
441-	1717	STOP SLAVES
443-	1739	ERROR DATA SETUP
448-	1988	DATA WAS A WORD
448-	2000	DATA WAS A BYTE
450-	2013	DATA WAS A 7 BIT BYTE
450-	2028	DETERMINE XOR OF GOOD & BAD
452-	2037	LOG ERROR ON BAD BANK
456-	2120	ROUTINE SCOPE HANDLER
457-	2176	SUBR DISPLAY
459-	2199	ROUTINE ERROR HANDLER
462-	2282	ROUTINE ERROR MESSAGE TYPEOUT
475-	2581	SUBR WHO'S BOX IS THIS?
477-	2645	SUBR DETAILED ERROR REPORT
483-	2754	QUE MESSAGES FROM SLAVE CPU'S
485-	2789	ROUTINE BINARY TO OCTAL (ASCII) AND TYPE

486- 2867	ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
487- 2924	ROUTINE TTY INPUT
489- 3007	CONTROL T
489- 3033	CONTROL S & CONTROL Q
491- 3147	ROUTINE READ AN OCTAL NUMBER FROM THE TTY
491- 3196	ROUTINE READ A DECIMAL NUMBER FROM THE TTY
492- 3255	ROUTINE SAVE AND RESTORE R0-R5
493- 3291	ROUTINE RANDOM NUMBER GENERATOR
495- 3321	ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
496- 3363	TABLES
496- 3365	APT MAILBOX-ETABLE
497- 3444	ROUTINE TRAP DECODER
499- 3471	TRAP TABLE
501- 3571	ODT (CEMKA) DATA AREA
548- 4403	TABLE ERROR POINTER
560- 4699	ERROR DATA TAGS (DT)
562- 4728	ERROR DATA FORMATS (DF)
564- 4746	ERROR MESSAGES (EM)
566- 4806	ERROR DATA HEADERS (DH)
568- 4838	MESSAGES

.TITLE CEMKABO 1170 MAIN MEMORY DIAG 1
.IDENT /X01.60/

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

IDENTIFICATION

PRODUCT CODE: AC-C644B-MC
PRODUCT NAME: CEMKABO 1170 MAIN MEMORY DIAG
PRODUCT DATE: 31 JUL 1979
MAINTAINER: PRODUCT ENHANCEMENT

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 DIGITALS 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) 1978,1979 DIGITAL EQUIPMENT CORPORATION

36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88

REVISION HISTORY

=====

REVISION	VERSION	DATE	AUTHOR	CHANGES
=====	=====	=====	=====	=====
CEMKAA	V03.00	24-FEB-78	KAY R. FISHER	NONE = NEW PROGRAM
CEMKAB	V09.00	9-OCT-78	KAY R. FISHER	1. ADD MULTIPROCESSOR CAPABILITIES 2. ADD SOFT ERROR TEST FOR ALPHA PARTICLES IN MOS CHIPS 3. ADD THE FOLLOWING FIELD SERVICE COMMANDS A. KAMIKAZE MODE B. CACHE ON/OFF C. SELECT ONLY MULTIPOINT TESTS D. TEST ONLY SELECTED BANKS 4. EXPAND DETAILED ERROR REPORT TO INCLUDE ALL CSR'S AND SP AT TIME OF ERROR 5. SPEED UP GALLOPING PATTERN BY EXECUTING OUT OF THE PAR'S 6. OPTIMIZE THE SHIFTING DIAGONAL PATTERN BY SHARING CODE WITH THE REFRESH PATTERN 7. ADD BRANCH GOBBLE PATTERN 8. ADD WRITE RECOVERY PATTERN 9. REPAIRED THE FOLLOWING BUGS A. UNDER APT FIELD SERVICE COMMANDS 1 & 2 B. UNDER APT USE OF ENVIRONMENTAL MODE DID NOT MEET APT SPEC C. FATAL PARITY ERRORS AT LOAD TIME THAT WERE CAUSED BY .BLKW STATEMENTS D. RESTART AT 202 CLEARS ERROR ACCOUNTING E. INTERMITTANT SINGLE BIT ERRORS IN CHECKBITS NOT DETECT F. ERROR COUNT WAS MOD 32K G. ODT HANGS AFTER 256 OPENS H. WRONG PC PRINTED ON SOME ERROR I. HIADRS LOADRS REG'S NOT CLEARE AFTER SIZING J. ACT MEMORY SIZE DEPENDENT BIT DEFINED WRONG K. UNABLE TO RUN WITHOUT A LINE CLOCK

	.SBTTL	OPERATIONAL SWITCH SETTINGS		
	.SBTTL	:SWITCH REGISTER DEFINITIONS		
	.SBTTL	:*		
	.SBTTL	:*	SWITCH	USE
	.SBTTL	:*	-----	-----
90	.SBTTL	:*	15	HALT ON ERROR
91	.SBTTL	:*	14	LOOP ON TEST
92	.SBTTL	:*	13	INHIBIT ERROR TYPEOUTS
93	.SBTTL	:*	12	INHIBIT RELOCATION
94	.SBTTL	:*	11	QUICK VERIFY / INHIBIT MARGIN TESTING
95	.SBTTL	:*	10	BELL ON ERROR
96	.SBTTL	:*	9	LOOP ON ERROR
97	.SBTTL	:*	8	HALT PROGRAM (UNRELOCATED & RESTORE LOADERS)
98	.SBTTL	:*	7	DETAILED ERROR REPORTS
99	.SBTTL	:*	6	PRINT CONFIGURATION MAP
100	.SBTTL	:*	5	LIMIT MAX ERRORS PER BANK
101	.SBTTL	:*	4	FAT TERMINAL (132 COLUMNS OR BETTER)
102	.SBTTL	:*	3	TEST MODE - SEE DOCUMENT
103	.SBTTL	:*	2	TEST MODE - SEE DOCUMENT
104	.SBTTL	:*	1	TEST MODE - SEE DOCUMENT
105	.SBTTL	:*	0	DETECT SINGLE BIT ERRORS
106	.SBTTL	:*		
107	.SBTTL	:*		
108	.SBTTL	:*		
109	.SBTTL	:*		
110	.SBTTL	:*		

```
113 000000      .ENABL ABS
114             .ENABL AMA
115             .DSABL GBL
116             :NOTE: CEMKAB.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
117             :THIS PROGRAM. ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
118             .MCALL SMACIT,..PUSH,..POP,..TAG,..BRAN,..EMIT,..EMITN,..EMITL,..EMITR
119             .MCALL .IFOPR,..IS,..GENBR,..OPADD,..OPSUB,CLEAR,SET,CLEARB,SETB
120             .MCALL RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
121             .MCALL IF,..OR,..IFARI,..LEAVE,..GOTO,OR,AND,THEN,ELSE,WHILE,CASE
122             .MCALL FOR,TO,DOWNTO,REPEAT,UNTIL,THRU,END,BEGIN
123             .MCALL $$END,LEAVE,JUMPTO,GOTO,PUSH,POP,LET
124             .MCALL .SIMPLE,..ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
125             .MCALL $CALL,$RETURN
126
127             .NLIST TTM                ; I WANT FAT PAPER!
128             .LIST MC,SYM              ; LIST MACRO CALLS, SYMBOL TABLE
129             .NLIST MD,CND,ME          ; DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
130             :LST$$= 0                 ; DEFINED TO LIST SUPERMAC EXPANSIONS
131             $SWR= 163000              ; USE THESE SYSMAC SWITCHES
132             $TN= 1                    ; FIRST TEST NUMBER TO ONE(1)
133 000000      SMACIT
```

136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192

```
.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
IS 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      ;;TTY 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
```

104401
104402
104403

104405

104407
104410

104411
104412
104413
104414

104415
104416

104417

104420
104421

104422

104423

```

193      104424      CACHOFF=104424      ;TURN OFF CACHE
194
195      104425      LOADCSR=104425      ;LOAD CORRECT CSR
196      104426      READCSR=104426      ;READ CORRECT CSR
197
198      104427      PERR01= 104427      ;PROGRAM DETECTED ERROR
199      104430      PERR02= 104430      ;PROGRAM DETECTED ERROR
200      104431      PERR03= 104431      ;PROGRAM DETECTED ERROR
201      104432      PERR04= 104432      ;PROGRAM DETECTED ERROR
202      104433      PERR07= 104433      ;PROGRAM DETECTED ERROR
203      104434      PERR10= 104434      ;PROGRAM DETECTED ERROR
204      104435      PERR11= 104435      ;PROGRAM DETECTED ERROR
205      104436      PERR12= 104436      ;PROGRAM DETECTED ERROR
206      104437      PERR13= 104437      ;PROGRAM DETECTED ERROR
207      104440      PERR14= 104440      ;PROGRAM DETECTED ERROR
208      104441      PERR15= 104441      ;PROGRAM DETECTED ERROR
209      104442      PERR16= 104442      ;PROGRAM DETECTED ERROR
210      104443      PERR17= 104443      ;PROGRAM DETECTED ERROR
211      104444      PERR20= 104444      ;PROGRAM DETECTED ERROR
212      104445      PERR21= 104445      ;PROGRAM DETECTED ERROR
213      104446      PERR22= 104446      ;PROGRAM DETECTED ERROR
214      104447      PERR23= 104447      ;PROGRAM DETECTED ERROR
215      104450      PERR24= 104450      ;PROGRAM DETECTED ERROR
216      104451      PERR25= 104451      ;PROGRAM DETECTED ERROR
217      104452      PERR26= 104452      ;PROGRAM DETECTED ERROR
218      104453      PERR27= 104453      ;PROGRAM DETECTED ERROR
219      104454      PERR30= 104454      ;PROGRAM DETECTED ERROR
220      104455      PERR31= 104455      ;PROGRAM DETECTED ERROR
221      104456      PERR32= 104456      ;PROGRAM DETECTED ERROR
222      104457      PERR33= 104457      ;PROGRAM DETECTED ERROR
223      104460      PERR34= 104460      ;PROGRAM DETECTED ERROR
224      104461      PERR35= 104461      ;PROGRAM DETECTED ERROR
225      104462      PERR36= 104462      ;PROGRAM DETECTED ERROR
226      104463      PERR37= 104463      ;PROGRAM DETECTED ERROR
227      104464      PERR40= 104464      ;PROGRAM DETECTED ERROR
228      104465      PERR41= 104465      ;PROGRAM DETECTED ERROR
229      104466      PERR42= 104466      ;PROGRAM DETECTED ERROR
230      104467      PERR43= 104467      ;PROGRAM DETECTED ERROR
231
232      104470      ECCDIS= 104470      ;DISABLE ECC ON ALL CSR'S
233      104471      ECC1DIS=104471      ;DISABLE ECC ON 1 SELECTED CSR
234      104472      ECCINIT=104472      ;INITIALIZE ALL MK11 CSR'S
235      104473      ECC1INIT=104473      ;INITIALIZE 1 SELECTED MK11 CSR
236      104474      CBCSR= 104474      ;WRITE GENERATED CHECKBITS IN ALL CSR'S
237      104475      CB1CSR= 104475      ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
238      104476      WASSBE= 104476      ;WAS THERE A SBE ON ANY CSR?
239      104477      WAS1SBE=104477      ;WAS THERE A SBE ON 1 SELECTED CSR?
240      104500      WASDBE= 104500      ;WAS THERE A DBE ON ANY CSR?
241      104501      WAS1DBE=104501      ;WAS THERE A DBE ON 1 SELECTED CSR?
242      104502      CLRCSR= 104502      ;CLEAR ALL CSR'S
243      104503      CLR1CSR=104503      ;CLEAR 1 SELECTED CSR
244      104504      CHKDIS= 104504      ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
245      104505      CHK1DIS=104505      ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
246      104506      ENASBE= 104506      ;ENABLE TRAPS ON SBE'S FROM ALL CSR'S
247      104507      ENA1SBE=104507      ;ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
248      104510      TSTREAD=104510      ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
249      104511      INVALID=104511      ;INVALIDATE BACKGROUND PATTERN ON "BANK"
    
```

```
.SBTTL DEFINE BASIC PDP11 STUFF

252
253
254      :*INITIAL ADDRESS OF THE STACK POINTER
255      002000      STACK= 2000      ;;FIRST ADDRESS OF THE STACK
256      002000      KERSTK= STACK  ;;KERNEL STACK
257      000740      SUPSTK= 740    ;;SUPERVISOR STACK
258      000700      USESTK= 700    ;;USER STACK
259      104000      ERROR=EMT      ;;BASIC DEFINITION OF ERROR CALL
260      000004      SCOPE=IOT      ;;BASIC DEFINITION OF SCOPE CALL
261      177776      PSW= 177776    ;;PROCESSOR STATUS WORD
262      :STKLMT=177774      ;;STACK LIMIT REGISTER
263      :PIRQ= 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
264      177570      DSWR= 177570   ;;HARDWARE SWITCH REGISTER
265      177570      DDISP= 177570  ;;HARDWARE DISPLAY REGISTER
266      177546      LKS= 177546    ;;LINE CLOCK (KW11-L) STATUS REGISTER
267
268      :*MISCELLANEOUS DEFINITIONS
269      000011      HT= 11          ;;CODE FOR HORIZONTAL TAB
270      000012      LF= 12          ;;CODE LINE FEED
271      000015      CR= 15          ;;CODE CARRIAGE RETURN
272      000200      CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
273
274      :*GENERAL PURPOSE REGISTER DEFINITIONS
275      :SP=R6        ;;STACK POINTER
276      :KSP=SP      ;;KERNEL STACK POINTER
277      000006      SSP=SP         ;;SUPERVISOR STACK POINTER
278      000006      USP=SP         ;;USER STACK POINTER
279      :PC=R7        ;;PROGRAM COUNTER
280
281      :*'SWITCH REGISTER' SWITCH DEFINITIONS
282      100000      SW15= 100000
283      040000      SW14= 40000
284      020000      SW13= 20000
285      010000      SW12= 10000
286      004000      SW11= 4000
287      002000      SW10= 2000
288      001000      SW9= 1000
289      000400      SW8= 400
290      000200      SW7= 200
291      000100      SW6= 100
292      000040      SW5= 40
293      000020      SW4= 20
294      000010      SW3= 10
295      000004      SW2= 4
296      000002      SW1= 2
297      000001      SW0= 1
298
299      :*DATA BIT DEFINITIONS (BIT00 TO BIT15)
300      100000      BIT15= 100000
301      040000      BIT14= 40000
302      020000      BIT13= 20000
303      010000      BIT12= 10000
304      004000      BIT11= 4000
305      002000      BIT10= 2000
306      001000      BIT9= 1000
307      000400      BIT8= 400
308      000200      BIT7= 200
```

```

309      000100      BIT6= 100
310      000040      BIT5= 40
311      000020      BIT4= 20
312      000010      BIT3= 10
313      000004      BIT2= 4
314      000002      BIT1= 2
315      000001      BIT0= 1
316
317      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
318      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
319      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
320      ;TBITVEC=14      ;;'T' BIT
321      ;TRTVEC=      14      ;;TRACE TRAP
322      ;BPTVEC=      14      ;;BREAKPOINT TRAP (BPT)
323      000020      IOTVEC= 20     ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
324      000024      PWRVEC= 24     ;;POWER FAIL
325      000030      EMTVEC= 30     ;;EMULATOR TRAP (EMT) **ERROR**
326      000034      TRAPVEC=34     ;;"TRAP" TRAP
327      000060      TKVEC= 60     ;;TTY KEYBOARD VECTOR
328      ;TPVEC= 64      ;;TTY PRINTER VECTOR
329      ;LKVEC= 100     ;;LINE CLOCK (KW11-L) VECTOR
330      000114      CACHVEC=114    ;;CACHE ERROR INTERRUPT VECTOR
331      000114      PARVEC=CACHVEC
332      ;PIRQVEC=240    ;;PROGRAM INTERRUPT REQUEST VECTOR
333      000250      MMVEC= 250     ;;MEMORY MANAGEMENT VECTOR
334      ;SBTTL DEFINE CACHE REGISTERS
335      177740      LOADRS = 177740 ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
336      177742      HIADRS = 177742 ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
337      177744      MEMERR = 177744 ;;CACHE ERROR REGISTER
338      177746      CONTRL = 177746 ;;MEMORY CONTROL REGISTER
339      177750      MAINT = 177750 ;;MEMORY MAINTENANCE REGISTER
340      ;HITMIS = 177752 ;;HIT MISS REGISTER "1" IMPLIES HIT IN CACHE
341
342      ;SBTTL DEFINE CPU REGISTERS
343      177760      SIZELO = 177760 ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
344      ;SIZEHI = 177762 ;;TO GET TO THE LAST 32 WORDS OF MEMORY
345      ;SIZEHI = 177762 ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
346      ;CURRENTLY ALL ZERO
347      177764      SYSTID = 177764 ;;SYSTEM ID REGISTER
348      177766      CPUERR = 177766 ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
349
350      ;SBTTL DEFINE MEMORY MANAGEMENT REGISTERS
351      ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
352      177572      MMRO= 177572
353      177574      MMR1= 177574
354      177576      MMR2= 177576
355      172516      MMR3= 172516
356
357      ;*USER "I" PAGE DESCRIPTOR REGISTERS
358      177600      UIPDR0= 177600
359      ;UIPDR1= 177602
360      ;UIPDR2= 177604
361      ;UIPDR3= 177606
362      ;UIPDR4= 177610
363      ;UIPDR5= 177612
364      ;UIPDR6= 177614
365      ;UIPDR7= 177616
    
```

```

366
367          : *USER 'D' PAGE DESCRIPTOR REGISTORS
368          : UDPDR0= 177620
369          : UDPDR1= 177622
370          : UDPDR2= 177624
371          : UDPDR3= 177626
372          : UDPDR4= 177630
373          : UDPDR5= 177632
374          : UDPDR6= 177634
375          177636 UDPDR7= 177636
376
377          : *USER 'I' PAGE ADDRESS REGISTERS
378          FASTCITY=UIPAR0
379          177640 UIPAR0= 177640 ; PATTERN PROGRAM SPACE
380          : UIPAR1= 177642 ; PATTERN PROGRAM SPACE
381          177644 UIPAR2= 177644 ; PATTERN PROGRAM SPACE
382          177646 UIPAR3= 177646 ; PATTERN PROGRAM SPACE
383          177650 UIPAR4= 177650 ; PATTERN PROGRAM SPACE
384          177652 UIPAR5= 177652 ; PATTERN PROGRAM SPACE
385          177654 UIPAR6= 177654 ; PATTERN PROGRAM SPACE
386          177656 UIPAR7= 177656 ; PATTERN PROGRAM SPACE
387
388          : *USER 'D' PAGE ADDRESS REGISTERS
389          177660 UDPAR0= 177660 ; PATTERN PROGRAM SPACE
390          177662 UDPAR1= 177662 ; PATTERN PROGRAM SPACE
391          : UDPAR2= 177664 ; PATTERN PROGRAM SPACE
392          : UDPAR3= 177666 ; PATTERN PROGRAM SPACE
393          : UDPAR4= 177670 ; PATTERN PROGRAM SPACE
394          : UDPAR5= 177672 ; PATTERN PROGRAM SPACE
395          : UDPAR6= 177674 ; PATTERN PROGRAM SPACE
396          177676 UDPAR7= 177676 ; PATTERN PROGRAM SPACE
397
398          : *SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
399          172200 SIPDR0= 172200
400          : SIPDR1= 172202
401          : SIPDR2= 172204
402          : SIPDR3= 172206
403          : SIPDR4= 172210
404          : SIPDR5= 172212
405          : SIPDR6= 172214
406          : SIPDR7= 172216
407
408          : *SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
409          : SDPDR0= 172220
410          : SDPDR1= 172222
411          : SDPDR2= 172224
412          : SDPDR3= 172226
413          : SDPDR4= 172230
414          : SDPDR5= 172232
415          : SDPDR6= 172234
416          172236 SDPDR7= 172236
417
418          : *SUPERVISOR 'I' PAGE ADDRESS REGISTERS
419          172240 SIPAR0= 172240
420          : SIPAR1= 172242
421          : SIPAR2= 172244
422          172246 SIPAR3= 172246 ; TEST AREA
    
```

```
423          :SIPAR4=          172250          :TEST AREA
424          :SIPAR5=          172252          :TEST AREA
425          :SIPAR6=          172254          :TEST AREA
426          :SIPAR7=          172256
427
428          :*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
429          172260      SDPAR0= 172260
430          :SDPAR1=          172262
431          :SDPAR2=          172264
432          :SDPAR3=          172266
433          :SDPAR4=          172270
434          :SDPAR5=          172272
435          :SDPAR6=          172274
436          172276      SDPAR7= 172276
437
438          :*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
439          172300      KIPDR0= 172300
440          :KIPDR1=          172302
441          :KIPDR2=          172304
442          :KIPDR3=          172306
443          :KIPDR4=          172310
444          :KIPDR5=          172312
445          :KIPDR6=          172314
446          :KIPDR7=          172316
447
448          :*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
449          :KDPDR0=          172320
450          :KDPDR1=          172322
451          :KDPDR2=          172324
452          :KDPDR3=          172326
453          :KDPDR4=          172330
454          :KDPDR5=          172332
455          :KDPDR6=          172334
456          172336      KDPDR7= 172336
457
458          :*KERNEL 'J' PAGE ADDRESS REGISTERS
459          172340      KIPAR0= 172340
460          :KIPAR1=          172342
461          :KIPAR2=          172344
462          :KIPAR3=          172346
463          172350      KIPAR4= 172350
464          172352      KIPAR5= 172352
465          172354      KIPAR6= 172354
466          172356      KIPAR7= 172356
467
468          :*KERNEL 'D' PAGE ADDRESS REGISTERS
469          172360      KDPAR0= 172360
470          :KDPAR1=          172362
471          :KDPAR2=          172364
472          :KDPAR3=          172366
473          :KDPAR4=          172370
474          :KDPAR5=          172372
475          :KDPAR6=          172374
476          172376      KDPAR7= 172376
477
```

480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536

170200
170202
170204

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

DEFINE UNIBUS MAP REGISTERS

```
537      :MAPL33 = 170354
538      :MAPH33 = 170356
539      :MAPL34 = 170360
540      :MAPH34 = 170362
541      :MAPL35 = 170364
542      :MAPH35 = 170366
543      170370      :MAPL36 = 170370
544      170372      :MAPH36 = 170372
545      :MAPL37 = 170374
546      170376      :MAPH37 = 170376
547
548      .SBTTL DEFINE SOFTWARE SWITCH & DISPLAY REGISTERS
549      000174      DISPREG=174
550      000176      SWREG= 176
551
552      .SBTTL DEFINE MK11 REGISTERS
553      172100      MK11ADD=172100
554
555      .SBTTL DEFINE PARAMETERS
556      060000      FIRST=60000      ;START OF THE 16K TEST PATTERN AREA
557      157776      LAST=157776      ;END OF THE 16K TEST PATTERN AREA
558      040000      SIZE=40000      ;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)
559      011661      SECONDS=11661      ;CONSTANT FOR LENGTH OF TIME FOR HUNG CPU TEST
560      .SBTTL DEFINE MASTER & SLAVE BITS
561      000020      MASTER=20
562      000040      SLAVE1=40
563      000100      SLAVE2=100
564      000200      SLAVE3=200
```

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

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

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

606
607
608
609
610
611
612
613
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

```
.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
```

656
657
658
659
660
661
662
663
664
665
666
667
668
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
707
708

```
.SBTTL MACRO $$NEWTEST
.MACRO $$NEWTEST A,ASC,COMND
.IRP ASCII,<ASC>
.IF EQ $NWTST
$NWTST=1
.SBTTL T'A' ASCII
.NLIST
.LIST ME
.LIST
:*****
:*TEST A ASCII
.IFF
ASCII
.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 ASCII,<ASC>
.SBTTL ASCII
.NLIST
.LIST ME
.LIST
:*****
:*SUBTEST ASCII
.ENDM
:*****
.NLIST ME
.ENDM $SUBTST
```

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

```
.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
```

751
752
753
754
755
756
757
758
759
760
761
762
763
764
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

```

.SBTTL MACRO TYPOCS
***** TYPOCS *****
TYPOCS IS USED TO CHANGE A BINARY NUMBER TO AN OCTAL
NUMBER AND TYPE 1 TO 6 DIGITS
WITH OR WITHOUT LEADING ZEROS.

ARGUMENTS:
1) NUM NUMBER TO BE TYPED
2) REMARK ALLOWS A COMMENT TO BE MADE
3) N NUMBER OF DIGITS (1 TO 6) TO BE TYPED
4) Z BLANK=SUPPRESS LEADING ZEROS (TYPES SPACES)
NON-BLANK=TYPE LEADING ZEROS

ROUTINES REQUIRED
1) CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
2) TYPE AN ASCII STRING (.$TYPE)

EXAMPLES:
1) TYPOCS #12345,<TYPES '5'>,1
2) TYPOCS #004,<TYPES '04'>,.2,X
3) TYPOCS #004,<TYPES '4'>,.2
*****

.MACRO TYPOCS NUM,REMARK,N,Z
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM,-(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPOS ;;GO TYPE--OCTAL ASCII
.IF NB N
.BYTE N ;;TYPE N DIGIT(S)
.IFF
.BYTE 6 ;;TYPE 6 DIGITS
.ENDC
.IF NB Z
.BYTE 1 ;;TYPE LEADING ZEROS
.IFF
.BYTE 0 ;;SUPPRESS LEADING ZEROS
.ENDC
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPOCS
    
```

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

```
.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
:MUNUS SIGN WILL BE TYPED.
:ARGUMENTS:
:1) NUM NUMBER TO BE TYPED
:2) REMARK ALLOWS A COMMENT TO BE MADE
:ROUTINES REQUIRED
:1) CONVERT BINARY TO DECIMAL AND TYPE (.$TYPDEC)
:2) TYPE AN ASCIZ STRING (.$TYPE)
:EXAMPLES
:1) TYPDEC SIZE,<TYPE THE CONTENTS OF SIZE>
:2) TYPDEC #-10.,<TYPE A MINUS TEN>
:*****
.MACRO TYPDEC NUM,REMARK
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM,-(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPDEC
```

MACRO BMOV

849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
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

```

.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

```

906
907
908
909
910
911
912

TOHERE
FROMHERE
.DSABL CRF
.IIF DF LSTSS .NLIST ME
.ENABL CRF
.ENDC
.ENDM BMOV

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

```
.SBTTL MACRO MAP
***** MAP *****
: THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
: TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-157777)).
: 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
```

954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994

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

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

997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036

```
.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")  
: I USE THE SET4 AND RES4 MACROS AROUND CODE THAT I EXPECT TO TRAP TO 4  
: LIKE LOOKING FOR ALL POSSIBLE CSR'S AND ETC. WHENEVER CODE IS NOT  
: SURROUNDED BY SET4 AND RES4 THEN ANY TRAPS TO 4 WILL CAUSE AN ERROR  
: PRINTOUT THAT SAYS 'UNEXPECTED TRAP TO 4' AND ALL THE ASSOCIATED REGISTER JUNK  
:*****
```

```
.MACRO SET4 ARG  
.NLIST  
.DSABL CRF  
.IIF DF LST$$ .LIST ME  
.ENABL CRF  
.LIST  
MOV ARG,4  
.DSABL CRF  
.IIF DF LST$$ .NLIST ME  
.ENABL CRF  
.ENDM SET4  
  
.MACRO RES4  
.NLIST  
.DSABL CRF  
.IIF DF LST$$ .LIST ME  
.ENABL CRF  
.LIST  
MOV #TIMEOUT,4  
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
```

1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060

```
.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
```

```

1063
1064
1065 000000 000000 000000
1066 000177
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086 000046 000046
1087 000046 015000
1088 000052 000052
1089 000052 040000
1090
1091 000024 000024
1092 000024 000200
1093 000042 000042
1094 000042 002000
1095 000044 000044
1096 000044 075236
1097 000200 000200
1098 000200 000437
1099 000202 000442
1100 000204 000446
1101 000300 000300
1102 000300 005037 002752
1103 000304 000137 004352
1104 000310
1105 000316 000137 004352
1106 000322
1107 000330 000137 000300
1108 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 BIT14 ;: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)
BR START4 ;'RUN MULTIPROCESSOR TESTS WITHOUT THE IIST' START
.=300
START1: CLR RESTART
JMP START
START2: SET RESTART
JMP START
START4: SET NOIIST
JMP START1
.=STACK
  
```

1111			.SBTTL	VARIABLES	INITIALIZED TO ZERO
1112			:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS		
1113			:*USED IN THE PROGRAM.		
1114	002000		\$CMTAG:		:: START OF COMMON TAGS
1115	002000	000000	BADCSR:	0	:: CSR DETERMINED TO HAVE AN ERROR BY BAD BOX ROUTINE
1116	002002	000000	SELONLY:	0	:: SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
1117	002004	000000	MULTIONLY:	0	:: RUN ONLY MULTI-PORT TESTS FLAG
1118	002006	000000	DIAGFLAG:	0	:: SET FOR SHIFTING DIAGONAL TEST
1119	002010	000000	KAMIKAZE:	0	:: SET FOR KAMIKAZE MODE TESTING
1120	002012	000000	SKIPKAMI:	0	:: USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
1121			:NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER		
1122	002014	000	\$PATMAR:	.BYTE 0	:: PATTERN NUMBER & MARGIN VALUE
1123	002015	000	\$BANK:	.BYTE 0	:: BANK & SIGN = MARGIN INDICATOR
1124	002016	000	\$ERFLG:	.BYTE 0	:: CONTAINS ERROR FLAG
1125	002017	000	\$ITEMB:	.BYTE 0	:: CONTAINS ITEM CONTROL BYTE
1126	002020	000000	LASTERROR:	.WORD 0	:: NUMBER OF ERRORS ON LAST PASS
1127	002022	000000	SBEFLAG:	.WORD 0	:: A SINGLE BIT ERROR HAS OCCURED THIS PASS
1128	002024	000000	ERRPC:	.WORD 0	:: CONTAINS PC OF ERROR FOR TYPEOUT
1129	002026	000000	BADPC:	.WORD 0	:: CONTAINS PC OF ERROR
1130	002030	000000	ERRSP:	.WORD 0	:: CONTAINS SP OF ERROR FOR TYPEOUT
1131	002032	000000	BADSP:	.WORD 0	:: CONTAINS SP OF ERROR
1132	002034	000000	ERRPSW:	.WORD 0	:: CONTAINS PSW OF ERROR FOR TYPEOUT
1133	002036	000000	BADPSW:	.WORD 0	:: CONTAINS PSW OF ERROR
1134	002040	000000	ADDRESS:	.WORD 0	:: CONTAINS ADDRESS OF 'BAD' DATA
1135	002042	000000	PADDRESS:	.WORD 0	:: ADDRESS OF PARITY ERROR
1136	002044	000000	PHYADD:	.WORD 0,0	:: 22 BIT PHYSICAL ADDRESS
1137	002050	000000	GOOD:	.WORD 0	:: CONTAINS 'GOOD' DATA
1138	002052	000000	GOOD2:	.WORD 0	:: CONTAINS 'GOOD2' DATA
1139	002054	000000	GOOD3:	.WORD 0	:: CONTAINS 'GOOD3' DATA
1140	002056	000000	BAD:	.WORD 0	:: CONTAINS 'BAD' DATA
1141	002060	000000	BAD2:	.WORD 0	:: CONTAINS 'BAD2' DATA
1142	002062	000000	BAD3:	.WORD 0	:: CONTAINS 'BAD3' DATA
1143	002064	000000	BADXOR:	.WORD 0	:: XOR OF GOOD & BAD = BAD BITS!
1144	002066	000000	\$AUTO:	.WORD 0	:: AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
1145	002070	000000	FATAL\$:	.WORD 0	:: FATAL ERROR INDICATOR
1146	002072	000000	NEMCNT:	0	:: NON-EXISTANT MEMORY COUNTER (HOLES)
1147	002074	000000	PARCNT:	0	:: PARITY ERROR COUNTER
1148	002076	000000	PATERR:	0	:: PATTERN ERROR COUNTER
1149	002100	000000	NOPAR:	0	:: NO PARITY ERROR MODE INDICATOR
1150	002102	000000	NONEM:	0	:: NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
1151	002104	000000	NOCLOCK:	0	:: NO LINE CLOCK IS ON THIS SYSTEM
1152	002106	000000	BANK:	0	:: MEMORY BANK UNDER TEST
1153	002110	000000	BANKINDEX:	0	:: USED TO INDEX INTO CONFIG TABLE
1154	002112	000000	MARGIN:	0	:: INDICATES WHICH MARGIN IS IN PROGRESS
1155	002114	000000	CPUBIT:	0	:: CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
1156	002116	000000	MUT:	0	:: MEMORY UNDER TEST FLAG
1157	002120	000000	PATTERN:	0	:: PATTERN NUMBER UNDER TEST
1158	002122	000000	KPFLAG:	0	:: BANK IS PROTECTED REGION OF MK11
1159	002124	000000	ACFLAG:	0	:: BANK CAN BE ACCESSED BY THIS CPU
1160	002126	000000	MKFLAG:	0	:: IF SET INDICATES MK11 UNDER TEST
1161	002130	000000	PFLAG:	0	:: BANK IS IN PROGRAM SPACE
1162	002132	000000	RRFLAG:	0	:: BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
1163	002134	000000	RLFLAG:	0	:: PROGRAM IS RELOCATED FLAG
1164	002136	000000	BMFLAG:	0	:: 'BANK IS IDENTIFIED AS BAD MEMORY' FLAG
1165	002140	000000	TMFLAG:	0	:: 'TYPE OF MEMORY TO TEST' FLAG; 0 = MK11
1166	002142	000000	ABORTFLAG:	0	:: 'ABORT OCCURED' FLAG
1167	002144	000000	CSR:	.WORD 0,0	:: DATA TO OR FROM CSR

```

1168 002150 000000 000000 SBECRS: .WORD 0,0 ;CSR STORED HERE ON MK11 DETECTED SBE'S
1169 002154 000000 000000 000000 CSR0: .WORD 0,0,0,0,0,0,0,0 ;SAVE CSR'S AFTER GETSIZE CALL
      002162 000000 000000 000000
1170 002170 000000 000000 000000 CSR2: .WORD 0,0,0,0,0,0,0,0 ;SAVE CSR+2'S AFTER GETSIZE CALL
      002174 000000 000000 000000
      002202 000000 000000 000000
1171 002210 000000 000000 000000 CSRFB: .WORD 0,0,0,0,0,0,0,0 ;SAVE FIRSTB'S AFTER GETSIZE CALL
      002214 000000 000000 000000
      002222 000000 000000 000000
      002230 000000 000000 000000
1172 002234 000000 000000 000000 CSRLB: .WORD 0,0,0,0,0,0,0,0 ;SAVE LASTB'S AFTER GETSIZE CALL
      002242 000000 000000 000000
      002250 000000 000000 000000
1173 002254 000000 CSRREL: .WORD 0 ;CSR'S ARE RELOCATED TO SHADOW MEMORY FLAG
1174 002256 000000 CSRNO: 0 ;CSR ADDRESS NUMBER (3 LSB'S)
1175 002260 000000 FIRSTB: 0 ;FIRST BANK CONTROLLED BY THIS CSR
1176 002262 000000 LASTB: 0 ;LAST BANK CONTROLLED BY THIS CSR
1177 002264 000000 INTERL: 0 ;INTERLEAVE FACTOR (0,1,2,3)
1178 :THESE LOCATIONS STORE GPR'S DURING SUPERVISOR TESTS
1179 002266 000000 SUPDR0: 0
1180 002270 000000 SUPDR1: 0
1181 002272 000000 SUPDR2: 0
1182 002274 000000 SUPDR3: 0
1183 002276 000000 SUPDR4: 0
1184 002300 000000 SUPDR5: 0
1185 002302 000000 SUPDR6: 0
1186 002304 000000 DUMMY: 0 ;DUMMY LOCATION FOR ADDRESS PASSING
1187 :THESE LOCATIONS STORE GPR'S & PSW DURING DETAILED ERROR PRINTOUTS
1188 002306 000000 DETR0: 0
1189 002310 000000 DETR1: 0
1190 002312 000000 DETR2: 0
1191 002314 000000 DETR3: 0
1192 002316 000000 DETR4: 0
1193 002320 000000 DETR5: 0
1194 002322 000000 DETSP: 0
1195 002324 000000 DETPSW: 0
1196 002326 000000 DETFLAG: 0 ;DETAILED REPORT FLAG
1197 002330 000000 PENDBOX: 0 ;1 BIT PER EXISTING CSR THAT STARTS AT A BANK ABOVE 166
      ;CSR'S 0,1,2,3,4,5,6,7 REPRESENTED BY BITS
      ;7,6,5,4,3,2,1,0 RESPECTIVELY
      ;MK11 CONTROL'S HAVE BEEN TESTED FLAG
1198
1199
1200 002332 000000 CONTFLAG: 0
1201 002334 000000 MKCSRS: .WORD 0 ;LO BYTE: 1 BIT PER EXISTING CSR
      ;CSR'S 0,1,2,3,4,5,6,7 REPRESENTED BY BITS
      ;7,6,5,4,3,2,1,0 RESPECTIVELY
      ;HI BYTE: 1 BIT PER EXISTING CSR WITH INTERNAL INTERLEAVE
      ;INTERNAL INTERLEAVE REPRESENTED BY BITS
      ;15,14,13,12,11,10,9,8 RESPECTIVELY
      ;FIRST ADDRESS UNDER CONTROL OF THIS CSR
1202
1203
1204
1205
1206
1207 002336 000000 CSRFIRST: 0
1208 002340 000000 000000 DATBUF: .WORD 0,0 ;TWO WORD DATA BUFFER
1209 002344 000000 000000 TSTDAT: .WORD 0,0 ;TWO WORD TEST DATA
1210 002350 000000 000000 SBEMSK: .WORD 0,0 ;TWO WORD SINGLE BIT ERROR MASK
1211 002354 000000 000000 DBEMSK: .WORD 0,0 ;TWO WORD DOUBLE BIT ERROR MASK
1212 002360 000000 SUPDOADD: 0 ;ADDRESS OF SUBROUTINE TO EXECUTE IN SUPERVISOR MODE
1213 002362 000 PASFLG: .BYTE 0 ;LOCAL LOOP PASS CONTROL
1214 002363 000 UPPFLG: .BYTE 0 ;LOCAL LOOP PASS CONTROL
1215 002364 000000 STRTDI: 0 ;LOAD STRTDI WITH THE STARTING ADDRESS OF THE DIAGONAL
1216 002366 000000 REALPAT: 0 ;REAL PATTERN UNDER TEST
    
```

VARIABLES INITIALIZED TO ZERO

1217	002370	000000	OLDCACHE:0	:BACKED UP VALUE OF CACHE CONTROL REGISTER
1218	002372	000000	PARTHERE:0	:PARITY TRAPS SOMETIMES GO TO ADDRESS STORED HERE
1219	002374	000000	FSSTACK:0	:STACK SAVED HERE IF IN FIELD SERVICE MODE
1220	002376	000000	GLDMARGIN:0	:BACKED UP VALUE OF LAST MARGIN SETTING
1221	002400	000000	NEWBANK:0	:USED FOR RELOCATION TO A NEW BANK
1222	002402	000000	SOURCE: .WORD 0	:SOURCE OF DATA WORDS FOR CHECKBIT GENERATION SUBROUTINE
1223	002404	000000	CHECK: .WORD 0	:CHECKBITS TO BE LOADED INTO CSR
1224	002406	000000	PCBUMP: .WORD 0	:VALUE TO BUMP THE PC BY TO RECOVER AFTER A PARITY TRAP
1225	002410	000000	CSRFBANK:0	:FIRST BANK OWNED (AT LEAST IN PART) BY THIS CSR
1226	002412	000000	CSRLBANK:0	:LAST BANK OWNED (AT LEAST IN PART) BY THIS CSR
1227	002414	000000	CSRINC: 0	:VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR
1228	002416	000	CSRINDEX: .BYTE 0	:AN INDEX VARIABLE FOR A CSR LOOP
1229	002417	000	CSROUDEX: .BYTE 0	:AN INDEX VARIABLE FOR A CSR LOOP
1230	002420	000000	CSRLOOP:0	:LOOP CONTROL FOR CSR TESTING
1231	002422	000000	SIDE: 0	:INDEX VARIABLE FOR EACH SIDE (CONTROLLER) OF A CSR
1232	002424	000000	SUCCESS:0	:FLAG SET BY SUCCESSFULL TASK OR SUBROUTINE
1233	002426	000000	ZERO: 0	:FOR AID IN 'MOV' INSTRUCTIONS
1234	002430	000000	ZEROS: 0	:FOR AID IN 'MOV' INSTRUCTIONS
1235	002432	000000	SPECIAL:0	:FLAG FOR SPECIAL CASE OF NO INTERNAL INTERLEAVE
1236	002434	000000	TIME: 0	:SECONDS THAT BATTERIES SHOULD LAST
1237	002436	000000	NULLFLAG:0	:SET WHEN RUNNING NULL PATTERNS
1238	002440	000000	QVFLAG: 0	:FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
1239	002442	000000	ACTFLAG:0	:FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
1240	002444	000000	APTFLAG:0	:FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
1241	002446	000000	XXDPCHAIN:0	:FLAGS XXDP CHAIN MODE PROGRAMMING RULES
1242			:NOTE: THESE TWO BYTES MUST STAY TOGETHER	
1243	002450	000	\$NULL: .BYTE 0	::CONTAINS NULL CHARACTER FOR FILLS
1244	002451	000	\$FILLS: .BYTE 0	::CONTAINS # OF FILL CHARACTERS
1245	002452	000	\$TPFLG: .BYTE 0	::'TERMINAL NOT AVAILABLE' FLAG
1246			.EVEN	
1247	002454	000000	\$ESCAPE:0	::ESCAPE ON ERROR ADDRESS
1248	002456	000000	EVEN: 0	:USED FOR ALTERNATE DATA PATTERNS
1249	002460	000000	STRIPES:0	:COUNTS DIAGONAL STRIPES
1250	002462	000000	COUNT: 0	:BACKED UP COPY OF STRIPES
1251	002464	000000	NOTAB: 0	:NO TABLE BEING PRINTED - NOW
1252	002466	000000	MJSIZE: 0	:SIZE OF MJ11 MEMORY IN K WORDS
1253	002470	000000	MKSIZE: 0	:SIZE OF MK11 MEMORY IN K WORDS
1254	002472	000000	TOOMANY:0	:FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
1255	002474	000000	NOECCFLAG:0	:ECC CAPABILITY DISABLED ON THE CONTROL PANEL
1256	002476	000000	GDSWITCH:0	:GOOD VALUE FOR SOME SWITCHES
1257	002500	000000	BDSWITCH:0	:BAD VALUE FOR SOME SWITCHES
1258	002502	000000	WRITEONLY:0	:FLAG TO PATTERNS TO WRITE ONLY
1259	002504	000000	READONLY:0	:FLAG TO PATTERNS TO READ ONLY
1260	002506	000000	TESTADD:0	:THE ADDRESS TO RUN CSR TESTS ON
1261	002510	000000	ALLCPUS:0	:CPUBIT'S FOR ALL CPU'S PRESENT
1262	002512	000000	HIBANK: 0	:HIGHEST BANK OF MEMORY
1263	002514	000000	HIACBANK: 0	:1ST INACCESSABLE BANK OF MEMORY
1264	002516	000000	UNITOP: 0	:HIGHEST ACCESSABLE BANK OF MEMORY THRU UNIBUS MAP
1265	002520	000000	STOPOK: 0	:FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
1266	002522	000000	APTCORE:0	:AMOUNT OF CORE MEMORY ACCORDING TO APT
1267	002524	000000	APTMOS: 0	:AMOUNT OF MOS MEMORY ACCORDING TO APT
1268	002526	000000	NOFSMODE:0	:FLAG TO DISABLE FIELD SERVICE MODE
1269	002530	000000	NOERROR:0	: 'THIS IS NOT AN ERROR' FLAG
1270	002532	000000	LOADBANK:0	:BANK LOADERS ARE RELOCATED TO
1271	002534	000000	TEMP: 0	:USED FOR JUNK
1272	002536	000000	QUICK: 0	:QUICK STOP FLAG FOR APT POWER FAIL
1273	002540	000000	NOSCOPE:0	: 'NO SCOPE LOOP ALLOWED' FLAG

VARIABLES INITIALIZED TO ZERO

1274 002542 000000
1275 002544 000000
1276 002546 000000
1277 002550 000000
1278 002552 000000
1279 002554 000000
1280 002556 000000
1281 002560 000000
1282 002562 000000
1283 002564 000000
1284 002566 000000
1285 002570 000000
1286 002572 000000
1287 002574 000000
1288 002576 000000
1289 002600 000000
1290 002602 000000
1291 002604 000000

FSINFLAG:0
APTSIZE:0
FS7FLAG:0
KI0: 0
KI1: 0
KI2: 0
KI3: 0
KI4: 0
KI5: 0
KI6: 0
KI7: 0
SLAERROR:0
HUNGMSB:0
HUNGTIME:0
CONFGERRR:0
TWOCRS:0
I: 0
J: 0

;'FIELD SERVICE - NO INTERNAL INTERLEAVE'' FLAG
:APT SIZING INFO FLAG
:TRUE WHEN IN FIELD SERVICE COMMAND 7
:SAVED KIPAR0 TO PASS CORRECT VALUE TO SLAVE
:SAVED KIPAR1 TO PASS CORRECT VALUE TO SLAVE
:SAVED KIPAR2 TO PASS CORRECT VALUE TO SLAVE
:SAVED KIPAR3 TO PASS CORRECT VALUE TO SLAVE
:SAVED KIPAR4 TO PASS CORRECT VALUE TO SLAVE
:SAVED KIPAR5 TO PASS CORRECT VALUE TO SLAVE
:SAVED KIPAR6 TO PASS CORRECT VALUE TO SLAVE
:SAVED KIPAR7 TO PASS CORRECT VALUE TO SLAVE
:SLAVE HAD AN ERROR FLAG
:USED FOR TIMES OVER 13 SECONDS
:TIME TO ALLOW THE SYSTEM TO BE HUNG (MAX = 13. SEC)
:CONFIGURATION ERROR FLAG
:TO FLAG LOADCSR TRAP TO LOAD TWO CSR'S
:USED FOR GENERAL PURPOSE INDEXING
:USED FOR GENERAL PURPOSE INDEXING

```
1294 .SBTTL MULTIPOINT VARIABLES
1295 :MULTIPOINT DIRECTORY
1296 :PORTDIR:MASTER IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1297 :      SLAVE IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1298 :      SLAVE IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1299 :      SLAVE IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1300 002606 000000 000000 000000 PORTDIR:.WORD 0,0,0,0
      002614 000000
1301 002616 000000 000000 000000 TASK: .WORD 0,0,0,0
      002624 000000
1302 002626 000000 000000 000000 PTYFLAG:.WORD 0,0,0,0
      002634 000000
1303 002636 000000 000000 000000 PTYBUF: .WORD 0,0,0,0
      002644 000000
1304 002646 000000 000000 000000 PORTCOUNT:.WORD 0,0,0,0
      002654 000000
1305 002656 000000 000000 000000 SLFLAG: .WORD 0,0,0,0
      002664 000000
1306 002666 000000 MINDEX: 0 ;MASTER CPU'S INDEX # TO TABLES ABOVE
1307 002670 000000 LOCK: 0 ;MULTIPROCESSOR LOCK
1308 002672 000000 ACK: 0 ;ACKNOWLEDGE FLAG (USED FOR STARTING SLAVES)
1309 002674 000000 MASWR: 0 ;MASTER'S SWR
1310 002676 000000 MDISPLAY:0 ;MASTER CPU DISPLAY WHEN RUNNING MULTIPROCESSOR TESTS
1311 002700 $CMTGE: ;*END OF COMMON TAGS
```

1314
 1315 002700 000001
 1316 002702 000012
 1317 002704 000031
 1318 002706 002000
 1319 002710 000001
 1320 002712 177777
 1321 002714 176543
 1322 002716 123456
 1323 002720 176543
 1324 002722 123456
 1325 002724 177777
 1326 002726 177777
 1327 002730 000000
 1328 002732 000000
 1329 002734 000003
 1330 002736 177500
 1331 002740 177502
 1332 002742 000260
 1333 002744 052525
 1334 002746 000000
 1335 002750 000000
 1336 002752 000000
 1337 002754 000000
 1338 002756 000000
 1339 002760 000000
 1340
 1341
 1342 002762 000377
 1343 002764 177400
 1344
 1345
 1346 002766 177570
 1347 002770 177570
 1348 002772 177560
 1349 002774 177562
 1350 002776 177564
 1351 003000 177566
 1352 003002 012
 1353 003003 207 377 377
 003006 000
 1354 003007 077
 1355 003010 015 000
 1356 003011 012
 1357
 1358 003014 000000

SBTTL VARIABLES
 CACHK: 1
 ERRMAX: 10.
 SOBK: 25.
 KSTACK: STACK
 LOADHOME: 1
 WORST: 177777
 SEEDHI: 176543
 SEEDLO: 123456
 MSEEDH: 176543
 MSEEDL: 123456
 HEADER: 177777
 ONES: 177777
 SLAVES: 0
 ONCE: 0
 FLIPLOC: 3
 IISTACR: 177500
 IISTADR: 177502
 IISTVEC: 260
 SOFTPAT: 52525
 \$LPADR: .WORD 0
 \$LPERR: .WORD 0
 RESTART: 0
 \$ERTTL: .WORD 0
 NOIIST: 0
 PWLOCK: 0
 ***** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER *****
 BAKPAT: .WORD 377
 SWAPAT: .WORD 177400

 SWR: .WORD DSWR
 DISPLAY: .WORD DDISP
 \$TKS: 177560
 \$TKB: 177562
 \$TPS: 177564
 \$TPB: 177566
 \$FILLC: .BYTE 12
 \$BELL: .ASCIZ <207><377><377>
 \$QUES: .ASCII /?/
 \$CRLF: .ASCII <15>
 \$LF: .ASCIZ <12>
 .EVEN
 MULTIPORT: 0

INITIALIZED TO NON ZERO
 :CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
 :MAX # OF ERRORS PER BANK WITH SW11
 :SOB CONSTANT
 :STACK BEGINNING (USED TO KEEP SLAVES STACKS SEGREGATED)
 :HOME BANK OF LOADERS
 :SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
 :WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
 :WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
 :MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
 :MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
 :USED TO PRINT HEADINGS ONLY ONCE
 :FOR AID IN 'MOV' INSTRUCTIONS
 :# OF SLAVE CPU'S
 :FLAG FOR TYPING TITLE ONLY ONCE
 :COUNTER FOR FLIPING DATA ON WORST CASE NOISE TEST
 :ADDRESS OF THE IIST ACR (WAS 166000)
 :ADDRESS OF THE IIST ADR (WAS 166002)
 :ADDRESS OF THE IIST VECTOR (WAS 170)
 :PATTERN FOR SOFT ERROR BACKGROUND TESTS
 :CONTAINS SCOPE LOOP ADDRESS
 :CONTAINS SCOPE RETURN FOR ERRORS
 :RESTART (START ADD 202) FLAG
 :CONTAINS TOTAL ERRORS
 :NO IIST IS TO BE USED DURING MULTIPOINT TESTS
 :MULTIPROCESSOR LOCK FOR POWER FAILURE
 :BACKGROUND PATTERN *
 :SWAPPED BAKPAT *

 :ADDRESS OF SWITCH REGISTER
 :ADDRESS OF DISPLAY REGISTER
 :TTY KBD STATUS
 :TTY KBD BUFFER
 :TTY PRINTER STATUS REG. ADDRESS
 :TTY PRINTER BUFFER REG. ADDRESS
 :INSERT FILL CHARS. AFTER A 'LINE FEED'
 :CODE FOR BELL
 :QUESTION MARK
 :CARRIAGE RETURN
 :LINE FEED
 :MULTIPROCESSING FLAG

CONFIGURATION TABLE

1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1398
1399
1400

003016 000170
004336 000000 000000 000000
004344 000000 000000 000000

.SBTTL CONFIGURATION TABLE
:CONFIG:FIRST 16K CONFIGURATION WORDS (3 EACH)
2ND 16K CONFIGURATION WORDS (3 EACH)
167TH 16K CONFIGURATION WORDS (3 EACH)

:CONFIGURATION WORDS:

LOW: BIT 0-2 INTERLEAVE FACTOR
BIT 3 EXTERNAL INTERLEAVE
BIT 4 MASTER CAN ACCESS (MEMORY EXISTS)
BIT 5 SLAVE #1 CAN ACCESS (SHARED)
BIT 6 SLAVE #2 CAN ACCESS (SHARED)
BIT 7 SLAVE #3 CAN ACCESS (SHARED)
BIT 8-10 BOX NUMBER (0-7)
BIT 11 ERRORS PRESENT
BIT 12 PROTECTED REGION OF A MK11 BOX
BIT 13-14 MEMORY TYPE;0=MJ11;1=MK11
BIT 15 PROTECTED (PROGRAM SPACE)
MED: BIT 0-5 NOT USED
BIT 6-8 MK11 CSR ADDRESS
BIT 9-11 NOT USED
BIT 12 INTERNAL INTERLEAVE
BIT 13 'BACKGROUND PATTERN VALID' FLAG
BIT 14 BANK SELECTED FOR TEST BY FIELD SERVICE MODE
BIT 15 LOADERS HOME BANK
HIGH: BIT 0 INDICATES BANK IS ADDRESSED BY CSR 172100
BIT 1 INDICATES BANK IS ADDRESSED BY CSR 172104
BIT 2 INDICATES BANK IS ADDRESSED BY CSR 172110
BIT 3 INDICATES BANK IS ADDRESSED BY CSR 172114
BIT 4 INDICATES BANK IS ADDRESSED BY CSR 172120
BIT 5 INDICATES BANK IS ADDRESSED BY CSR 172124
BIT 6 INDICATES BANK IS ADDRESSED BY CSR 172130
BIT 7 INDICATES BANK IS ADDRESSED BY CSR 172134
BIT 8-15 NUMBER OF ERRORS

CONFIG: .REPT 170
CONFIEND: .WORD 0,0,0,0,0,0

;THESE WORDS SHOULD NEVER GET WRITTEN INTO
;IF THEY DO THE PROGRAM IS OVERFLOWING IT'S
;CONFIGURATION TABLE (SOFTWARE BUG)!

```

1402
1403 004352

1404 004352
1405 004360 000005
1406 004362 013706 002706
1407 004366
1408
1409 004406 005004
1410 004410
1411 004416 077404
1412 004420 000000
1413 004422
1414 004422 000137 005752
1415 004426
1416 004426 012700 002000
1417 004432 005020
1418 004434 022700 002700
1419 004440 001374
1420 004442

1421
1422
1423
1424 004442 012737 000004 002100
1425 004450 012700 001000
1426 004454 012737 000015 177746
1427 004462 012720 000000
1428 004466 020027 002000
1429 004472 103773
1430 004474 012700 117146
1431 004500 012720 000000
1432 004504 020027 137776
1433 004510 103773
1434 004512 005037 002100
1435
1436 004516 005237 002670
1437 004522
1438 004532 000000
1439 004534 000776
1440 004536
  
```

```

.SBTTL ***** MAIN *****
START: SUBTST <<INITIALIZE VARIABLES TO ZERO>>
:*****
:*SUBTEST INITIALIZE VARIABLES TO ZERO
:*****
      SET ACK ;ACKNOWLEDGE THAT I HAVE STARTED (IN CASE I'M A SLAVE)
      RESET
      MOV KSTACK,SP ;;SETUP THE STACK POINTER
      IF MULTIPOINT EQ ONES AND CPUBIT NE #MASTER
      :WAIT FOR ACK TO CLEAR OR TIMEOUT & HALT
      CLR R4
1$: IF ACK IS TRUE
      SOB R4,1$
SHALT: HALT ;THE MASTER WOULD NOT ALLOW THIS SLAVE TO PROCEED
      END ;OF IF ACK IS TRUE
      JMP BEGIN
      END ;OF IF MULTIPOINT
      MOV #SCMTAG,R0 ;;FIRST LOCATION TO BE CLEARED
1$: CLR (R0)+ ;;CLEAR MEMORY LOCATION
      CMP #SCMTGE,R0 ;;DONE?
      BNE 1$ ;LOOP BACK IF NO
      SUBTST <<CLEAR NON-PROGRAM SPACE>>
:*****
:*SUBTEST CLEAR NON-PROGRAM SPACE
:*****
      ;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 #4,NOPAR ;PARITY ACTION = COUNT & IGNORE
      MOV #1000,R0
      MOV #15,CONTRL ;DISABLE CACHE & ABORTS
9$: MOV #0,(R0)+ ;CLEAR THE STACK AREA
      CMP R0,#STACK
      BLO 9$
      MOV #END+2,R0
8$: MOV #0,(R0)+ ;CLEAR FROM THE END OF THE PROGRAM TO THE XXDP LOADE
      CMP R0,#157776-3192.
      BLO 8$
      CLR NOPAR ;RESTORE DEFAULT PARITY ACTION

      INC LOCK ;ONLY ALLOW ONE CPU AT A TIME TO GO THRU HERE
      IF LOCK NE #1
MHALT1: HALT ;MULTIPLE MASTER CPU'S ARE RUNNING
      BR MHALT1
      END ;OF IF LOCK
  
```

1443 004536

SUBTST <<INITIALIZE VARIABLES TO NON ZERO>>

 :*SUBTEST INITIALIZE VARIABLES TO NON ZERO

1444 004536
 1445 004544 012737 000003 002734
 1446 004552
 1447 004560 012737 176543 002720
 1448 004566 012737 123456 002722
 1449 004574 013737 002720 002714
 1450 004602 013737 002722 002716
 1451 004610 012737 000377 002762
 1452 004616 012737 177400 002764
 1453 004624 012737 052525 002744
 1454 004632 012737 005752 002746
 1455 004640 012737 005752 002750
 1456 004646

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
 MOV #52525,SOFTPAT
 MOV #BEGIN,\$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
 MOV #BEGIN,\$LPERR ;;SETUP THE ERROR LOOP ADDRESS
 SUBTST <<INITIALIZE VECTORS>>

 :*SUBTEST INITIALIZE VECTORS

1457 004646 012737 066232 000020
 1458 004654 012737 000340 000022
 1459 004662 012737 066662 000030
 1460 004670 012737 000340 000032
 1461 004676 012737 075252 000034
 1462 004704 012737 000340 000036
 1463 004712 012737 055160 000024
 1464 004720 012737 000340 000026
 1465 004726 012737 036454 000114
 1466 004734 012737 000340 000116
 1467 004742 012737 036752 000010
 1468 004750 012737 000340 000012
 1469 004756 012737 036726 000004
 1470 004764 012737 000340 000006
 1471 004772 012737 036740 000250
 1472 005000 012737 000340 000252
 1473 005006 012737 036764 000170
 1474 005014 012737 000340 000172
 1475 005022 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
 MOV #\$\$IITRAP,170 ;SET UP FOR UNEXPECTED IIST TRAPS
 MOV #340,172
 CACHON ;TURN CACHE ON

1479 005024
 1480
 1481
 1482
 1483 005024
 1484 005032 012700 003016
 1485 005036 005020
 1486 005040 022700 004336
 1487 005044 001374
 1488 005046
 1489
 1490 005046 012737 000020 002114
 1491 005054
 1492
 1493
 1494
 1495 005054
 1496 005062 012737 177570 002766
 1497 005070 012737 177570 002770
 1498 005076
 1499 005106 000403
 1500 005110 012716 005116
 1501 005114 000002
 1502 005116
 1503 005130 012737 000176 002766
 1504 005136 012737 000174 002770
 1505 005144
 1506
 1507
 1508 005144
 1509 005144
 1510 005152 005737 177546
 1511 005156 000403
 1512 005160
 1513 005166

```

SUBTST <<CLEAR THE CONFIGURATION TABLE>>
*****
*SUBTEST CLEAR THE CONFIGURATION TABLE
*****
:THIS ZEROS (UNLESS WE STARTED AT ADDRESS 202) THE CONFIG TABLE
:WHICH IS FULLY DISCRIBED AT LOCATION 'CONFIG'.
.DSABL LSB
IF RESTART IS FALSE
1$: MOV #CONFIG,RO
    CLR (RO)+
    CMP #CONFIEND,RO
    BNE 1$
END :OF IF RESTART
.DSABL LSB
MOV #MASTER,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.
.DSABL 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
3$: BR 2$ ;:BRANCH IF NO TIMEOUT
    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

SUBTST <<FIND OUT IF ANY LINE CLOCK IS PRESENT>>
*****
*SUBTEST FIND OUT IF ANY LINE CLOCK IS PRESENT
*****
SET4 #4$
TST LKS
BR 5$ ;IF NO TRAP SKIP
4$: SET NOCLOCK ;:SET NO LINE CLOCK FLAG
5$: RES4 ;:RESTORE TRAPS TO 4 TO DEFAULT
    
```

1516 005200

SUBAAB: SUBST <<SETUP ACT, APT, & XXDP>>

:SUBTEST SETUP ACT, APT, & XXDP

1517

:THIS SETS UP A BUNCH OF FLAGS TO TELL THE PROGRAM EVERYTHING

1518

:IT CARES TO KNOW ABOUT APT, ACT, & XXDP.

1519 005200 005037 075142

CLR \$PASS ;CLEAR PASS COUNT

1520 005204

IFB #BIT5 SET.IN \$ENV

1521 005214

SET \$TPFLG ;INDICATE NO TERMINAL

1522 005222

END :OF IFB #BIT5

1523 005222

IFB #BIT7 SET.IN \$ENV

1524 005232

SET APTSIZE

1525 005240 012737 075156 002766

MOV #\$\$WREG,SWR ;USE APT SWR

1526 005246

END :OF IFB #BIT7

1527 005246

IFB \$ENV EQ #1

1528 005256

SET APTFLAG,QVFLAG,\$AUTO,QUICK

1529 005306 012737 043640 000024

MOV #APTDOWN,PWRVEC

1530 005314

ELSE

1531 005316

IF 42 NE #STACK AND 42 NE #0

1532 005334

SET QVFLAG,\$AUTO

1533 005350

IF 42 EQ #SENDAD

1534 005360

SET ACTFLAG

1535 005366

ELSE

1536 005370

SET XXDPCHAIN

1537 005376

END :OF IF 42

1538 005376

END :OF IF 42

1539 005376

END :OF IFB \$ENV

1542 005376

SUBTST <<INITIALIZE PATTERNS>>

: *SUBTEST INITIALIZE PATTERNS

1543
1544
1545
1546
1547 005376
1548 005404 012700 075220
1549 005410 012001
1550 005412 012703 020124
1551 005416 004737 005472
1552 005422 012001
1553 005424 004737 005472
1554 005430 012001
1555 005432 012703 021402
1556 005436 004737 005472
1557 005442 012001
1558 005444 004737 005472
1559 005450 012001
1560 005452 012703 021600
1561 005456 004737 005472
1562 005462 012001
1563 005464 004737 005472
1564 005470
1565 005470 000420
1566
1567 005472

: 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.
IF APTFLAG IS TRUE
MOV #SDDW0,R0
MOV (R0)+,R1
MOV #MKCSRT,R3
CALL PATPLUG
MOV (R0)+,R1
CALL PATPLUG
MOV (R0)+,R1
MOV #MKPAT,R3
CALL PATPLUG
MOV (R0)+,R1
CALL PATPLUG
MOV #MJPAT,R3
CALL PATPLUG
MOV (R0)+,R1
CALL PATPLUG
END ;OF IF APTFLAG IS TRUE
BR SUBAAA

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

: *SUBTEST SUBR PLUG IN NULL PATTERNS

1568 005472
1569 005500 006001
1570 005502
1571 005504 012713 025772
1572 005510
1573 005510 062703 000002
1574 005514
1575 005530 000207

FOR I := #1 TO #16.
ROR R1
ON.NOERROR ;IF CARRY CLEAR
MOV #MT0999,(R3)
END ;OF ON.ERROR
ADD #2,R3
END ;OF FOR
RETURN

1578 005532

SUBAAA: SUBTST <<PROTECT PROGRAM & LOADERS>>

 :SUBTEST PROTECT PROGRAM & LOADERS

1579 005532 052737 100000 003016
 1580 005540 052737 100000 003024
 1581 005546
 1582 005564
 1583 005570
 1584 005570

BIS #BIT15,CONFIG ;PROTECT PROGRAM SPACE (BANK 0)
 BIS #BIT15,CONFIG+6 ;PROTECT LOADER SPACE (BANK 1)
 IF #SENDAD NE 42 AND ONCE IS FALSE ;NOT ACT-11 AND NOT DONE ONCE?
 TYPE MSG000 ;TYPE PROGRAM TITLE
 END ;OF IF #SENDAD
 SUBTST <<MULTIPOINT DETERMINATION ROUTINE>>

 :SUBTEST MULTIPOINT DETERMINATION ROUTINE

1585
 1586
 1587
 1588
 1589

:DETERMINE
 :1. IF WE ARE MULTIPROCESSING
 :2. IF WE CAN USE THE IIST INTERFACE
 :3. EACH CPU'S ID (IIST NUMBER OR SYSTEM ID REGISTER)

1590 005570
 1591 005576
 1592 005604
 1593 005610 104413
 1594 005612
 1595 005616
 1596 005616
 1597 005634 000763
 1598 005636
 1599 005636
 1600 005644 013737 177764 002606
 1601 005652
 1602 005654
 1603 005662 012777 100000 175046
 1604 005670 017700 175042
 1605 005674 072027 177770
 1606 005700 042700 177774
 1607 005704 010037 002606
 1608 005710 005777 175024
 1609 005714
 1610 005714
 1611 005722 013706 002706
 1612 005726
 1613 005740
 1614 005746 005037 002670
 1615

ENABL LSB
 IF NOIIST IS TRUE
 IF ONCE IS FALSE
 1\$: TYPE MSG080 ;ENTER # OF SLAVE CPU'S (0-3)?
 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
 POP SLAVES
 END ;OF IF ONCE
 IF SLAVES LT #0 OR SLAVES GT #3
 BR 1\$
 END ;OF IF SLAVES
 IF SLAVES EQ #0 THEN GOTO 2\$
 MOV SYSTID,PORTDIR
 ELSE
 SET4 #2\$;TRAPS TO 4 GOTO 2\$
 MOV #BIT15,@IISTACR ;IS THERE AN IIST?
 MOV @IISTACR,R0 ;YES - GET ID
 ASH #-8,R0
 BIC #^C3,R0
 MOV R0,PORTDIR ;IDENTIFY MASTER
 TST @IISTADR ;SEE IF 2ND IIST REGISTER TRAPS
 END ;OF IF NOIIST
 SET MULTIPOINT ;MULTIPOINTING!
 2\$: MOV KSTACK,SP
 RES4 ;RESET TRAPS TO 4 TO DEFAULT
 SET ONCE
 CLR LOCK
 .DSABL LSB

1618 005752

BEGIN: SUBTST <<SLAVES START HERE>>

:SUBTEST SLAVES START HERE

1619 005752 000240
1620 005754 005237 002670
1621 005760
1622 005770 000000
1623 005772 000776
1624 005774
1625 005774
1626 006004 004737 061244
1627 006010

NOP
INC LOCK ;ONLY ONE CPU AT A TIME IS ALLOWED HERE
IF LOCK NE #1
MHALT2: HALT ;MULTIPLE CPU'S EXECUTING SAME CODE
BR MHALT2
END :OF IF LOCK
IF CPUBIT NE #MASTER ;IF I'M A SLAVE
CALL SLAVEMAP
END :OF IF CPUBIT

1630 006010

```

SUBTST <<DETERMINE MK11 ACCESS METHOD>>
:*****
:*SUBTEST      DETERMINE MK11 ACCESS METHOD
:*****
:DETERMINE HOW WE ACCESS THE MK11 CSR'S.
:IF I CAN'T ACCESS THE CSR'S NORMALLY THEN I ASSUME THAT I MUST
:RELOCATE THRU THE UNIBUS MAP SO I SET THE 'CSR'S RELOCATED' FLAG (CSRREL)
:NOTE:  I DON'T REALLY KNOW AT THIS POINT IF THERE IS ANY MK11 MEMORY YET
:       I ONLY DETERMINE HOW I WILL HAVE TO ACCESS IT IF THERE IS ANY!
:
SET4   #5$           ;TRAPS TO 4 GOTO 5$
MOV    #8,R0        ;TEST FOR 8 CSR'S
MOV    #MK11ADD,R2  ;R2 = CSR ADDRESS
4$:   TST   (R2)     ;ANSWER R0 TRAP
      CLR  CSRREL    ;ANSWER - CSR'S NOT RELOCATED
      BR   6$       ;BRANCH - OUT
5$:   ADD  #4,SP     ;FIX STACK FROM TRAP
      ADD  #4,R2     ;NEXT CSR = LAST CSR + 4
      SOB RO,4$     ;LOOP TILL DONE
      SET  CSRREL
6$:   RES4           ;RESET TRAPS TO 4 TO DEFAULT
      SUBTST <<SETUP USER & SUPERVISOR STACK>>
:*****

```

1631
 1632
 1633
 1634
 1635
 1636 006010
 1637 006016 012700 000010
 1638 006022 012702 172100
 1639 006026 005712
 1640 006030 005037 002254
 1641 006034 000410
 1642 006036 062706 000004
 1643 006042 062702 000004
 1644 006046 077011
 1645 006050
 1646 006056
 1647 006070

```

:*****
:*SUBTEST      SETUP USER & SUPERVISOR STACK
:*****
DEENERGIZE           ;TURN OFF MEMORY MANAGEMENT
:SET PREVIOUS MODE TO SUPERVISOR
BIC    #BIT13!BIT12,PSW
BIS    #BIT12,PSW
PUSH   #SUPSTK
MTPJ   SSP
:SET PREVIOUS MODE TO USER
BIS    #BIT13!BIT12,PSW
PUSH   #USESTK
MTPJ   USP

```

1648 006070 104421
 1649
 1650
 1651 006072 042737 030000 177776
 1652 006100 052737 010000 177776
 1653
 1654 006106
 1655 006112 006606
 1656
 1657
 1658 006114 052737 030000 177776
 1659
 1660 006122
 1661 006126 006606

1664 006130

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

1665 006130

```
      IF CPUBIT EQ #MASTER AND $AUTO IS FALSE ;IF MASTER & NOT (APT OR ACT)
```

1666 006146

```
      IF SWR EQ #SWREG ;IF SOFTWARE SWITCH REG SELECTED
```

1667 006156 104407

```
      GTSWR ;;GET SOFT-SWR SETTINGS
```

1668 006160

```
      END ;OF IF SWR
```

1669 006160

```
      END ;OF IF CPUBIT
```

1670

1671 006160 012737 000020 172516

```
      MOV #20,MMR3 ;SET 22 BIT MODE
```

1672 006166 005037 177750

```
      CLR MAINT ;CLEAR MARGINS
```

1673

1674 006172

```
      SUBTST <<SETUP UNIBUS MAP>>
```

```
:*****  
:*SUBTEST   SETUP UNIBUS MAP  
:*****
```

1675 006172 012737 000077 170372

```
      MOV #77,MAPH36 ;MAP36 IS USED TO ACCESS MK11 CSR'S ON PDP11-70'S
```

1676 006200 012737 160000 170370

```
      MOV #160000,MAPL36 ;THAT DO NOT HAVE THE MULTIPROCESSOR MODIFICATIONS.
```

1677 006206

```
      CLEAR MAPH0,MAPL0 ;MAP0 IS USED FOR APT TO LOOK AT THE MAILBOX THRU.
```

1678 006216 004737 042026

```
      CALL LOWMAP ;SETUP LOWER UNIBUS MAP
```

1679 006222 052737 000040 172516

```
      BIS #BITS,MMR3 ;ENERGIZE UNIBUS MAP
```

1680

1681 006230

```
      SUBTST <<GET MEMORY MANAGEMENT READY>>
```

```
:*****  
:*SUBTEST   GET MEMORY MANAGEMENT READY  
:*****
```

1682 006230

```
      IF CPUBIT EQ #MASTER
```

1683 006240 104422

```
      KMAP ;MAP KERNEL SPACE 1 TO 1
```

1684 006242

```
      END ;OF IF CPUBIT
```

1685 006242

```
      MAP ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1
```

1686 006256 104420

```
      ENERGIZE ;TURN ON MEMORY MANAGEMENT
```

1689 006260

006260 000004

1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702
 1703 006262 005037 002256
 1704 006266
 1705 006272
 1706 006300 010637 006754
 1707 006304 006337 002334
 1708 006310 006337 002330
 1709 006314 104426
 1710 006316 052737 000001 002334
 1711 006324 005001
 1712 006326 013700 002146
 1713 006332 072027 177765
 1714 006336 042700 177770
 1715 006342
 1716 006356
 1717 006366 006756
 1718 006370 006760
 1719 006372 006760
 1720 006374 006766
 1721 006376 006766
 1722 006400 006774
 1723 006402 006766
 1724 006404 006774
 1725 006406
 1726 006414 010137 002264
 1727 006420 004737 017736
 1728
 1729 006424 013701 002256
 1730 006430 006201
 1731 006432 013761 002144 002154
 1732 006440 013761 002146 002174
 1733 006446 013761 002260 002214
 1734 006454
 1735 006464 052737 000001 002330
 1736 006472 000137 006712
 1737 006476
 1738 006476 013761 002262 002234
 1739 006504 013701 002260
 1740 006510 070127 000006
 1741 006514 010137 002110

```

NEWST <<ANALIZE CONFIGURATION>>
*****
*TEST 1 ANALIZE CONFIGURATION
*****
TST1: SCOPE
;THIS CODE DOES THE FOLLOWING FOR MK11 CSR'S
;1. FINDS ANY EXISTING CSR'S - SETTING BITS IN LOC 'MKCSRS' FOR
; EACH ONE PRESENT.
;2. SETS A BIT IN LOC 'MKCSRS' FOR ANY MK11 THAT IS INTERNALLY INTERLEAVED
;3. DETERMINES THE INTERLEAVE FACTOR & STARTING ADDRESS. USING THIS
; WE DETERMINE THE FIRST & LAST BANK INCLUDED IN THIS CSR. USING THAT
; THE CONFIGURATION TABLE IS UPDATED TO
; A. IDENTIFY BANKS WITHIN THE RANGE AS MK11 MEMORY.
; B. IDENTIFY BANKS WITHIN THE RANGE WITH INTERLEAVE FACTOR.
; C. IF NO EXTERNAL INTERLEAVE IDENTIFY BANKS WITHIN THE RANGE
; AS CONTROLLED BY THIS CSR
; D. IDENTIFY BANKS WITHIN THE RANGE BY A SPECIFIC BIT POSITION
; FOR THIS CSR
CLR CSRNO ;1ST CSR IS 0
PUSH CONTRL ;SAVE CACHE STATUS
SET4 #CONFGT ;TRAPS TO 4 GOTO CONFGT
MOV SP,CONFGT ;SAVE STACK
CONFG1: ASL MKCSRS
ASL PENDBOX
READCSR ;TRAP IF IT DOES NOT EXIST
BIS #1,MKCSRS ;MARK CSR AS PRESENT
CLR R1
MOV CSR+2,R0
ASH #-11.,R0
BIC #^C7,R0
IF #BIT0 SET.IN R0 THEN LET MKCSRS := MKCSRS SET.BY #BIT8
CASE R0
INTER0
INTER1
INTER2
INTER3
INTER4
INTER5
INTER6
INTER7
END ;OF CASE R0
MOV R1,INTERL ;SET INTERLEAVE FACTOR
CALL GETSIZE
;THE NEXT 6 LINES SAVE THE READ STATUS OF THE BOX CAPACITY
MOV CSRNO,R1
ASR R1
MOV CSR,CSR0(R1)
MOV CSR+2,CSR2(R1)
MOV FIRSTB,CSRFB(R1)
IF FIRSTB GT #167
BIS #BIT0,PENDBOX
JMP CONFG5
END ;OF IF FIRSTB
MOV LASTB,CSRFB(R1)
MOV FIRSTB,R1
MUL #6,R1 ;R1 = BANK * 6
MOV R1,BANKINDEX
    
```

```

1742 006520 013701 002110          CONFIG3: MOV     BANKINDEX,R1          ;GET NEXT INDEX (SAME FIRST TIME)
1743 006524 053761 002264 003016  BIS     INTERL,CONFIG(R1)        ;SETUP INTERLEAVE BITS
1744 006532 052761 020000 003016  BIS     #BIT13,CONFIG(R1)        ;IDENTIFY AS MK11
1745 006540 042761 040000 003016  BIC     #BIT14,CONFIG(R1)
1746 006546          IF #BIT8 SET.IN MKCSRS
1747 006556 052761 010000 003020  BIS     #BIT12,CONFIG+2(R1)
1748 006564          END ;OF IF #BIT8
1749 006564 013700 002256          MOV     CSRNO,R0                ;GET CSR LSB'S
1750 006570 072027 177776          ASH     #-2,R0                  ;MAKE (0-7)
1751 006574          IF #BIT13!BIT12 SET.IN CSR+2 ;ANY EXTERNAL INTERLEAVE?
1752 006604 052761 000010 003016  BIS     #BIT3,CONFIG(R1)        ;IDENTIFY IT
1753 006612          ELSE
1754 006614          PUSH  R0
1755 006616 072027 000010          ASH     #8,R0                  ;MOVE TO BITS 10-8
1756 006622 050061 003016          BIS     R0,CONFIG(R1)          ;IDENTIFY AS CSR & BOX NUMBER
1757 006626          POP   R0
1758 006630          END ;OF IF #BIT13!BIT12
1759 006630 012702 000001          MOV     #1,R2
1760 006634 072200          ASH     R0,R2                  ;R2 = 1,2,4,10,20,40,100, OR 200
1761 006636 050261 003022          BIS     R2,CONFIG+4(R1)        ;SETUP CSR WORD
1762 006642 062737 000006 002110  ADD     #6,BANKINDEX           ;NEXT TABLE ENTRY
1763 006650 005237 002260          INC     FIRSTB
1764 006654 023737 002260 002262  CMP     FIRSTB,LASTB           ;DONE?
1765 006662 101716          BLOS   CONFIG3                ;NO - LOOP
1766 006664 000412          BR     CONFIG5
1767 006666 013706 006754          CONFIG4: MOV     CONFSP,SP      ;FIX STACK FROM MULTIPLE TRAP
1768 006672          IF #BIT4 OFF.IN CPUERR
1769 006702          LET GOOD := #BIT4
1770 006710 104037          ERROR +37
1771 006712          END ;OF IF CPUERR
1772 006712 062737 000004 002256  CONFIG5: ADD     #4,CSRNO         ;BUMP CSR
1773 006720          IF CSRNO NE #40 THEN JUMPTO CONFIG1
1774 006734          RES4
1775 006746          POP   CONTRL                ;RESET TRAPS TO 4 TO DEFAULT
1776 006752 000413          BR     SUBAAS                 ;RESTORE CACHE STATUS
1777 006754 000000          CONFSP: 0                    ;STACK SAVED HERE!
1778
1779          ;TOTAL INTERLEAVE (0, 2-1, 4-1, OR 8-1)
1780 006756 000207          INTER0: RETURN                ;NO INTERLEAVE
1781
1782 006760          INTER1:
1783 006760 012701 000001          INTER2: MOV     #1,R1          ;2 WAY INTERLEAVE
1784 006764 000207          RETURN
1785
1786 006766          INTER3:
1787 006766          INTER4:
1788 006766 012701 000002          INTER6: MOV     #2,R1          ;4 WAY INTERLEAVE
1789 006772 000207          RETURN
1790
1791 006774          INTER5:
1792 006774 012701 000004          INTER7: MOV     #4,R1          ;8 WAY INTERLEAVE
1793 007000 000207          RETURN
  
```

```

1795 007002 104472
1796 007004

007004 000004
1797
1798
1799
1800
1801
1802
1803 007006 005037 002074
1804 007012 012737 000001 002100
1805 007020 005037 002072
1806 007024 012737 000001 002102
1807 007032
1808 007040 005000
1809 007042 012701 040000
1810 007046 104424
1811 007050 005720
1812 007052 077102
1813 007054 104423
1814
1815 007056 005737 002074
1816 007062 001403
1817 007064
1818 007072 005737 002072
1819 007076 001406
1820 007100 162737 000002 002040
1821 007106
1822 007114 053737 002114 003016
1823 007122
1824
1825 007134

007134
1826 007134
1827 007152 104506
1828 007154
1829 007156 104472
1830 007160
  
```

```

SUBAAS: ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
        NEWTST <<TEST BANK 0 ACCESSES>>
:*****
:*TEST 2 TEST BANK 0 ACCESSES
:*****
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 RO
        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

        SUBTST <<ENABLE MK11 FOR CORRECT TRAPS>>
:*****
:*SUBTEST ENABLE MK11 FOR CORRECT TRAPS
:*****
        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
  
```

1833 007160
 007160 000004
 1834
 1835
 1836
 1837
 1838
 1839
 1840
 1841
 1842 007162 005037 002106
 1843 007166 012737 000001 002100
 1844 007174 012737 000002 002102
 1845 007202
 1846 007210
 1847 007216 012737 005237 177654
 1848 007224 012737 002076 177656
 1849 007232 005237 002106
 1850 007236 022737 000170 002106
 1851 007244 001450
 1852 007246 013701 002106
 1853 007252 070127 000006
 1854 007256 010137 002110
 1855 007262 005037 002076
 1856 007266 005037 002074
 1857 007272 005037 002072
 1858 007276
 1859 007312 005761 003016
 1860 007316 100533
 1861 007320 012737 000207 177644 WARN1:
 1862 007326 012700 060000
 1863 007332 010004
 1864 007334 012701 040000
 1865 007340 010103
 1866 007342 005002
 1867 007344 104424
 1868 007346
 1869 007354 004737 177640
 1870 007360 104417
 1871 007362 104423
 1872 007364 000412
 1873 007366 005037 002106
 1874 007372
 1875 007404 005037 002100
 1876 007410 000543
 1877 007412 005737 002072
 1878 007416 001401
 1879 007420 000704
 1880 007422 104424
 1881 007424
 1882 007432 004737 177646
 1883 007436 104417
 1884 007440 104423
 1885 007442 005737 002076

```

NEWST <<TEST BANKS 1-167 (OCTAL) FOR ZEROS & ONES>>
:*****
:*TEST 3 TEST BANKS 1-167 (OCTAL) FOR ZEROS & ONES
:*****
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.
      ;ANY ERROR HERE WILL FORCE THE CONFIGURATION TABLE TYPEOUT!
      ;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
      BMOV MTP000 ;PUT IN FAST MEMORY
      MOV #005237,UIPAR6 ;PUT 'INC ' INSTRUCTION IN SUBROUTINE
      MOV #PATERR,UIPAR7 ;PUT IN ADDRESS OF OPERAND FOR ABOVE
TAG9$: INC BANK
      CMP #120,BANK ;DONE?
      BEQ TAG2$ ;YES - SKIP TO NEXT TEST
      MOV BANK,R1
      MUL #6,R1 ;BANK*3*2
      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
      TST CONFIG(R1) ;IS THIS BANK PROTECTED?
      BMI TSTBANK ;YES - GO TEST BANK SPECIAL
      MOV #207,UIPAR2 ;PUT 'RETURN' INSTRUCTION AFTER WRITE ROUTINE
      MOV #FIRST,R0
      MOV R0,R4
      MOV #SIZE,R1
      MOV R1,R3
      CLR R2 ;DATA IS ZEROS
      CACHOFF ;TURN CACHE OFF
      SUPERVISOR ;ENTER SUPERVISOR MODE
      CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
      KERNEL ;ENTER KERNEL MODE
      CACHON ;TURN CACHE ON
      BR TAG3$ ;SKIP NEXT INSTRUCTION
TAG2$: CLR BANK
      RES4 ;RESET TRAPS TO 4 TO DEFAULT
      CLR NOPAR ;INDICATE DEFAULT PARITY ACTION
      BR SUBAAQ
TAG3$: TST NEMCNT ;ANY TRAPS?
      BEQ 1$ ;NO - SKIP
      BR TAG9$ ;NOW - TRY NEXT BANK
1$: CACHOFF ;TURN CACHE OFF
      SUPERVISOR ;ENTER SUPERVISOR MODE
      CALL UIPAR3 ;FINISH PATTERN
      KERNEL ;ENTER KERNEL MODE
      CACHON ;TURN CACHE ON
      TST PATERR ;ANY PATTERN ERRORS
    
```

```

1886 007446 001031          BNE      2$                ;YES - SKIP
1887 007450 005737 002074  TST      PARCNT           ;ANY PARITY ERRORS
1888 007454 001026          BNE      2$                ;YES - SKIP
1889 007456 005737 002072  TST      NEMCNT           ;ANY NON EXISTANT MEMORY
1890 007462 001023          BNE      2$                ;YES - SKIP
1891 007464 012700 060000  MOV      #FIRST,R0
1892 007470 010004          MOV      R0,R4
1893 007472 012701 040000  MOV      #SIZE,R1
1894 007476 010103          MOV      R1,R3
1895 007500 013702 002726  MOV      ONES,R2          ;DATA IS ONES
1896 007504 012737 000240 177644 MOV      #000240,UIPAR2  ;PUT 'NOP' INSTRUCTION BACK IN SUBROUTINE
1897 007512 104424          CACHOFF                    ;TURN CACHE OFF
1898 007514          SUPERVISOR                ;ENTER SUPERVISOR MODE
1899 007522 004737 177640  CALL     FASTCITY          ;CALL TO THE USER INSTRUCTION PAR'S
1900 007526 104417          KERNEL                    ;ENTER KERNEL MODE
1901 007530 104423          CACHON                    ;TURN CACHE ON
1902 007532 013700 002110 2$: MOV      BANKINDEX,R0
1903 007536 005737 002076  TST      PATERR           ;ANY PATTERN ERRORS?
1904 007542 001006          BNE      3$                ;YES - SKIP
1905 007544 005737 002074  TST      PARCNT           ;ANY PARITY ERRORS?
1906 007550 001003          BNE      3$                ;YES - SKIP
1907 007552 005737 002072  TST      NEMCNT           ;ANY HOLES?
1908 007556 001406          BEQ      4$                ;NONE - SKIP
1909 007560 052760 004000 003016 3$: BIS      #BIT11,CONFIG(R0) ;SET ERROR BIT IN THIS BANK
1910 007566          SET      CONFGERROR      ;FORCE PRINTING OF CONFIGURATION TABLE
1911 007574 053760 002114 003016 4$: BIS      CPUBIT,CONFIG(R0) ;SET ACCESSED BIT
1912 007602 000137 007232  JMP      TAG9$
1913
1914          ;TEST A PROTECTED BANK
1915 007606          TSTBANK: PUSH     R1
1916 007610 012737 000001 002102 MOV      #1,NONEM          ;SET NON-EXISTANT MEMORY TO COUNT
1917 007616 012700 060000  MOV      #FIRST,R0
1918 007622 012701 040000  MOV      #SIZE,R1
1919 007626 104424          CACHOFF                    ;TURN CACHE OFF
1920 007630          SUPERVISOR                ;ENTER SUPERVISOR MODE
1921 007636 005720          4$: TST      (R0)+
1922 007640 077102          SOB      R1,4$
1923 007642 104417          KERNEL                    ;ENTER KERNEL MODE
1924 007644 104423          CACHON                    ;TURN CACHE ON
1925 007646 012737 000002 002102 MOV      #2,NONEM          ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
1926 007654          POP      R1
1927 007656          IF PARCNT NE #0
1928 007664 052761 004000 003016 BIS      #BIT11,CONFIG(R1) ;ERROR BANK
1929 007672          SET      CONFGERROR      ;FORCE PRINTING OF CONFIGURATION TABLE
1930 007700          END ;OF IF PARCNT
1931 007700          IF NEMCNT EQ #0
1932 007706 053761 002114 003016 BIS      CPUBIT,CONFIG(R1) ;ACCESSED BANK
1933 007714          END ;OF IF NEMCNT
1934 007714 000137 007232  JMP      TAG9$
    
```

```

1937 007720
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956 007720 000004
1957 007722 104424
1958 007724 005037 002122
1959 007730 012701 060000
1960 007734 005004
1961 007736
1962 007742 013703 002106
1963 007746 070327 000006
1964 007752 010337 002110
1965 007756
1966 007766
1967 010006
1968 010022 005037 002144
1969 010026
1970 010034
1971 010044 104504
1972 010046 010411
1973 010050 010461 000002
1974 010054 104502
1975 010056 005711
1976 010060 104500
1977
1978 010062
1979 010064 005037 002144
1980 010070 104504
1981 010072 013711 002726
1982 010076 013761 002726 000002
1983 010104 104502
1984 010106 005711
1985 010110 104500
1986 010112
1987 010114 012737 007400 002144
1988 010122 104504
1989 010124 010411
1990 010126 010461 000002

```

```

SUBAAQ: SUBTST <<DETERMINE PROTECTED BANKS>>
:*****
:*SUBTEST      DETERMINE PROTECTED BANKS
:*****
:THE MK11 INHIBITS ECC DISABLE (AND OTHER THINGS) ON THE BOTTOM
:16K WORDS OF EACH CONTROLLER (2 PER BOX).
:THIS IS CONSIDERED TO BE A PROTECTED BANK BY THE PROGRAM.
:USUALLY (ON A SINGLE BOX SYSTEM WITH 4K MOS RAMS) THIS IS BANKS #0 & #2
:HOWEVER IT MAY BE QUITE COMPLEX TO DETERMINE ON A GIVEN SYSTEM
:CONFIGURATION (MIXED MK11,MJ11; MIXED 4K MOS, 16K MOS; MULTIPLE MK11
:BOXES; MIXED INTERLEAVE SCHEMES) WHICH BANKS ARE PROTECTED.
:SO
:THIS ROUTINE ATTEMPS TO CREATE A DOUBLE BIT ERROR IN ADDRESS #0 & #2
:OF EVERY MK11 BANK. MK11 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!
SCOPE
CACHOFF          ;TURN CACHE OFF
CLR      KPFLAG
MOV      #FIRST,R1
CLR      R4
FOR BANK := #0 TO #167
  MOV      BANK,R3
  MUL      #6,R3
  MOV      R3,BANKINDEX
  IF (PUBIT SET.IN CONFIG(R3) ;IF ACCESSABLE
  IF #BIT13 SET.IN CONFIG(R3) AND #BIT14 OFF.IN CONFIG(R3) ;IF MK11
  MAP BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
  CLR      CSR
  SUPERVISOR      ;ENTER SUPERVISOR MODE
  PUSH      FIRST,FIRST+2 ;SAVE TWO TEST LOCATIONS
  CHKDIS          ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
  MOV      R4,(R1)          ;WRITE CHECKBITS (ALL ZEROS)
  MOV      R4,2(R1)
  CLRCSR          ;CLEAR MK11 CSRS
  TST      (R1)          ;READ CHECKBITS INTO REAL CSR
  WASDBE          ;WAS THERE A DOUBLE BIT ERROR
  ;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
  ON.NOERROR :1
  CLR      CSR
  CHKDIS          ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
  MOV      ONES,(R1)
  MOV      ONES,2(R1)
  CLRCSR          ;CLEAR MK11 CSRS
  TST      (R1)
  WASDBE          ;WAS THERE A DOUBLE BIT ERROR
  ON.NOERROR :2
  MOV      #7400,CSR
  CHKDIS          ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
  MOV      R4,(R1)
  MOV      R4,2(R1)

```

1991	010132	104502			CLRCR		;CLEAR MK11 CSRS
1992	010134	005711			TST (R1)		
1993	010136	104500			WASDBE		;WAS THERE A DOUBLE BIT ERROR
1994	010140				ON.NOERROR ;3		
1995	010142	012737	074000	002144	MOV #74000,CSR		
1996	010150	104504			CHKDIS		;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR*
1997	010152	010411			MOV R4,(R1)		
1998	010154	010461	000002		MOV R4,2(R1)		
1999	010160	104502			CLRCR		;CLEAR MK11 CSRS
2000	010162	005711			TST (R1)		
2001	010164	104500			WASDBE		;WAS THERE A DOUBLE BIT ERROR
2002	010166				END ;OF ON.NOERROR ;3		
2003	010166				END ;OF ON.NOERROR ;2		
2004	010166				END ;OF ON.NOERROR ;1		
2005	010166				ON.NOERROR		
2006					;MARK PROTECTED BANK		
2007	010170	013703	002110		MOV BANKINDEX,R3		
2008	010174	052763	010000	003016	BIS #BIT12,CONFIG(R3)		;IDENTIFY AS PROTECTED BY MK11
2009	010202				END ;OF ON.NOERROR		
2010	010202	104470			ECCDIS		;DISABLE ERROR CORRECTION
2011	010204	010411			MOV R4,(R1)		;CLEAR OUT DOUBLE BIT ERROR!
2012	010206	010461	000002		MOV R4,2(R1)		;CLEAR OUT DOUBLE BIT ERROR!
2013	010212	104502			CLRCR		
2014	010214				POP FIRST+2,FIRST		;RESTORE TWO TEST LOCATIONS
2015	010224	104417			KERNEL		;ENTER KERNEL MODE
2016	010226				END ;OF IF #BIT3		
2017	010226				END ;OF IF CPUBIT		
2018	010226				END ;OF FOR BANK		
2019	010242				MAP		;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
2020	010256	005037	002106		CLR BANK		
2021							
2022	010262				IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE		
2023	010300	104506			ENASBE		;TRAP ON SINGLE BIT ERRORS
2024	010302				ELSE		
2025	010304	104472			ECCINIT		;TRAP ON DOUBLE BIT ERRORS (NORMAL)
2026	010306				END ;OF IF #SWO		
2027							
2028	010306	104423			CACHON		;TURN CACHE ON

2031 010310

NEWST <<MK11 PROTECTION POINTER TEST>>

: *TEST 4 MK11 PROTECTION POINTER TEST

2032 010310 000004
2033 010312
2034 010312
2035 010316 004737 042422
2036 010322
2037 010336
2038 010344 104040
2039 010346
2040 010346
2041 010350
2042 010350
2043 010364

TST4: SCOPE
:INSURE THAT BOTTOM OF MK11 WAS PROTECTED WITH PROTECTION POINTER CLEAR
BEGIN PROTECTIONTEST
FOR BANK := #0 TO #167
CALL EXBANK
IF MKFLAG IS TRUE AND ACFLAG IS TRUE
IF KPFLAG IS FALSE
ERROR +40 ;MK11 PROTECTION POINTER FAILURE - DID NOT PROTECT BANK
END :OF IF KPFLAG
LEAVE PROTECTIONTEST
END :OF IF MKFLAG
END :OF FOR BANK
END PROTECTIONTEST

2046 010364

SUBTST <<CHECK SYSTEM SIZE REGISTER>>

:SUBTEST CHECK SYSTEM SIZE REGISTER

2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079

:THIS CHECKS THE SYSTEM SIZE REGISTER TO INSURE IT IS CORRECT
:AND PROGRAMS ANY MK11 BOXES ON LINE THAT ARE NOT INHIBITED.
:
:NOTE: IF THE SYSTEM SIZE REGISTER IS SET TO INDICATE A VERY SMALL
:MEMORY THE PROGRAM MAY NOT BE ABLE TO FUNCTION.
: IF THE MK11 BOX STARTING ADDRESS LOGIC MALFUNCTIONS THE PROGRAM
: WILL EITHER CRASH BECAUSE OF DUAL ADDRESSING PROBLEMS OR
: WILL HAVE HOLES IN MEMORY (NON-EXISTANT BANKS).
:
:THE TEST FUNCTIONS AS FOLLOWS
:
:1. THE PROGRAM CHECKS FOR MEMORY ON 16K WORD (BANK) BOUNDRIES
:THRU THE UNIBUS MAP (SO THAT THE SYSTEM SIZE REGISTER IS BYPASSED).
:LOCATION 'UNITOP' IS ASSIGNED THE NUMBER OF THE LAST BANK THAT WAS
:ACCESSABLE VIA THE UNIBUS MAP.
:
:2. LOCATION 'HIBANK' IS SET TO THE HIGHEST BANK THAT IS EITHER
:(1) DIRECTLY ACCESSABLE VIA THE MEMORY BUS OR
:(2) AN MK11 CSR CLAIMS THAT IT EXISTS.
:
:3. LOCATION 'HIACBANK' IS ASSIGNED THE NUMBER OF THE HIGHEST BANK
:THAT IS DIRECTLY ACCESSABLE VIA THE MEMORY BUS.
:
:4. IF THE HIGHEST ACCESSABLE BANK IS LESS THAN #170
:IF ANY BOX STARTING ADDRESSES WERE ABOVE BANK #167
:TRY TO PROGRAM THE STARTING ADDRESS OF THE PENDING BOX
:IF IT DOES PROGRAM DOWN AND THEREFORE ADD MEMORY TO THE SYSTEM
:RESTART THE PROGRAM
:NEXT PENDING BOX
:
:5. R0 IS SET TO THE HIGHEST BANK NUMBER OF HIBANK OR UNITOP
:
:6. IF THE SYSTEM SIZE REGISTER DOES NOT EQUAL R0'S BANK THEN PRINT THE ERROR

2082						:FIND UNITOP	
2083	010364	104424				CACHOFF	:TURN CACHE OFF
2084	010366	005037	170372			CLR MAPH36	:SET MAP TO BANK 1
2085	010372	012737	100000	170370		MOV #100000,MAPL36	
2086	010400					SET4 #2\$:TRAPS TO 4 GOTO 2\$
2087	010406	012737	000004	002100		MOV #4,NOPAR	:PARITY ACTION = COUNT & IGNORE
2088	010414	012700	140000			MOV #140000,R0	:PAR6 SELECTED
2089	010420	005710			1\$:	TST (R0)	:TRAP IF NON-EXISTANT
2090	010422	062737	100000	170370		ADD #100000,MAPL36	:BUMP 1 BANK
2091	010430	005537	170372			ADC MAPH36	
2092	010434	022737	000073	170372		CMP #73,MAPH36	:INTO 128K SHADOW MEMORY?
2093	010442	001366				BNE 1\$:NO - LOOP
2094	010444	000402				BR 3\$	
2095	010446	062706	000004		2\$:	ADD #4,SP	:FIX STACK FROM TRAP
2096	010452	162737	100000	170370	3\$:	SUB #100000,MAPL36	
2097	010460	005637	170372			SBC MAPH36	
2098	010464					RES4	:RESET TRAPS TO 4 TO DEFAULT
2099	010476	005037	002100			CLR NOPAR	:RESTORE DEFAULT PARITY ACTION
2100	010502	006137	170370			ROL MAPL36	
2101	010506	006137	170372			ROL MAPH36	
2102	010512	013737	170372	002516		MCV MAPH36,UNITOP	
2103							
2104						:FIND HIBANK	
2105	010520	012737	000077	170372		MOV #77,MAPH36	:RESTORE MAP36
2106	010526	012737	160000	170370		MOV #160000,MAPL36	
2107	010534	104423				CACHON	:TURN CACHE ON
2108	010536					PUSH SLAVES	
2109	010542	012737	000003	002730		MOV #3,SLAVES	:FAKE OUT THE EXBANK ROUTINE
2110	010550					BEGIN MEMSIZELOOP	
2111	010550					FOR BANK := #0 TO #167	
2112	010554	004737	042422			CALL EXBANK	
2113	010560					IF ACFLAG IS FALSE AND MKFLAG IS FALSE	
2114	010574					LEAVE MEMSIZELOOP	
2115	010576					END :OF IF ACFLAG	
2116	010576					END :OF FOR BANK	
2117	010612					END MEMSIZELOOP	
2118	010612	013700	002106			MOV BANK,R0	
2119	010616	005300				DEC R0	
2120	010620	010037	002512			MOV R0,HIBANK	

2123					:FIND HIACBANK
2124	010624				CLEAR BANK,HIACBANK
2125	010634				SET ACFLAG
2126	010642				WHILE ACFLAG IS TRUE AND BANK LT #167
2127	010660	005237	002106		INC BANK
2128	010664	004737	042422		CALL EXBANK
2129	010670				END ;OF WHILE ACFLAG
2130	010672				POP SLAVES ;RESTORE FROM FAKE OUT
2131	010676	013737	002106	002514	MOV BANK,HIACBANK
2132					
2133					:PROGRAM ON ANY BOXES SETUP FOR ALLOW PROGRAM
2134					:& HI (100) STARTING ADDRESSES
2135	010704				IF HIACBANK LT #170
2136	010714				IF PENDBOX IS TRUE
2137	010722	013701	002330		MOV PENDBOX,R1
2138	010726				FOR CSRNO := #0 TO #34 BY #4
2139	010732	106301			ASLB R1
2140	010734				ON.ERROR
2141	010736				SET TWOCSRS
2142	010744	012737	000001	002144	MOV #1,CSR
2143	010752	013737	002514	002146	MOV HIACBANK,CSR+2
2144	010760	000241			CLC
2145	010762	006037	002146		ROR CSR+2
2146	010766	005537	002146		ADC CSR+2 ;IF NOT 32K WORD BOUND LEAVE A WHOLE IN MEM
2147	010772	052737	002000	002146	BIS #BIT10,CSR+2 ;LOAD START ADD & INTERLEAVE BIT
2148	011000	104425			LOADCSR
2149	011002				CLEAR CSR,CSR+2,TWOCSRS
2150	011016	012737	000010	002144	MOV #BIT3,CSR
2151	011024	104425			LOADCSR
2152	011026	104426			READCSR
2153	011030	013702	002146		MOV CSR+2,R2
2154	011034	042702	177000		BIC #177000,R2
2155	011040	006302			ASL R2
2156	011042				IF R2 LT #167 ;IF I CHANGED IT'S STARTING ADDRESS
2157	011050	005037	002670		CLR LOCK
2158	011054	000137	004352		JMP START
2159	011060				END ;OF IF R2
2160	011060				END ;OF ON.ERROR
2161	011060				END ;OF FOR CSRNO
2162	011076				END ;OF IF PENDBOX
2163	011076				END ;OF IF HIACBANK

CHECK SYSTEM SIZE REGISTER

```

2166
2167 011076
2168
2169
2170 011110 072027 000011
2171 011114 052700 000777
2172 011120 020037 177760
2173 011124 001503
2174 011126 012737 177760 002040
2175 011134 010037 002050
2176 011140 013737 177760 002056
2177 011146 000300
2178 011150 004737 011210
2179 011154 010137 002476
2180 011160 013700 177760
2181 011164 000300
2182 011166 004737 011210
2183 011172 010137 002500
2184 011176 104023
2185 011200 013737 002754 002020
2186 011206 000452
2187
2188
2189 011210 005001
2190 011212
2191 011224
2192 011236
2193 011250
2194 011262
2195 011274
2196 011306
2197 011320
2198 011332 000207

:RO := MAX(HIBANK,UNITOP)
IF RO LT UNITOP THEN LET RO := UNITOP

:CHECK SYSTEM SIZE REGISTER
ASH #9,RO
BIS #777,RO ;RO = LAST GOOD ADDRESS (LESS 6 LSB'S)
CMP RO,SIZELO ;IS IT RIGHT?
BEQ SUBAAT
MOV #SIZELO,ADDRESS
MOV RO,GOOD
MOV SIZELO,BAD
SWAB RO
CALL SYSSIZE
MOV R1,GDSWITCH
MOV SIZELO,RO
SWAB RO
CALL SYSSIZE
MOV R1,BDSWITCH
ERROR +23
MOV $ERTTL,LASTERROR ;DON'T COUNT AS 'DETECTED BUT NOT ISOLATED'
BR SUBAAT

:DETERMINE SWITCH SETTINGS IN R1 FROM ADDRESS IN RO
SYSSIZE:CLR R1
IF #BIT0 SET.IN RO THEN LET R1 := R1 SET.BY #BIT2
IF #BIT1 SET.IN RO THEN LET R1 := R1 SET.BY #BIT0
IF #BIT2 SET.IN RO THEN LET R1 := R1 SET.BY #BIT4
IF #BIT3 SET.IN RO THEN LET R1 := R1 SET.BY #BIT7
IF #BIT4 SET.IN RO THEN LET R1 := R1 SET.BY #BIT3
IF #BIT5 SET.IN RO THEN LET R1 := R1 SET.BY #BIT1
IF #BIT6 SET.IN RO THEN LET R1 := R1 SET.BY #BIT5
IF #BIT7 SET.IN RO THEN LET R1 := R1 SET.BY #BIT6
RETURN
    
```

2201 011334

SUBAAT: SUBTST <<MOVE LOADERS IF NECESSARY>>

:SUBTEST MOVE LOADERS IF NECESSARY

2202
2203
2204

:THE TOP 16K OF THE PROGRAM (WHICH INCLUDES LOADERS) WILL BE MOVED
:TO AN MK11 PROTECTED AREA IF IT IS NOW IN MK11 MEMORY.
:IF IT IS IN CORE IT WILL BE LEFT ALONE.

2205 011334

IF CPUBIT EQ #MASTER

2206 011344 042737 100000 003024

BIC #BIT15,CONFIG+6 ;UNPROTECT OLD LOADER SPACE

2207 011352

BEGIN FIRSTPROTECT

2208 011352

FOR BANK := #1 TO #167

2209 011360 004737 042422

CALL EXBANK

2210 011364

IF ACFLAG IS TRUE AND BMFLAG IS FALSE

2211 011400

IF KPFLAG IS TRUE OR MKFLAG IS FALSE

2212 011414

LEAVE FIRSTPROTECT

2213 011416

END ;OF IF KPFLAG

2214 011416

END ;OF IF ACFLAG

2215 011416

END ;OF FOR BANK

2216 011432 012737 000001 002106

MOV #1,BANK

2217 011440 004737 042422

CALL EXBANK

2218 011444

END FIRSTPROTECT

2219 011444 013737 002106 002710

MOV BANK,LOADHOME

2220 011452 013700 002110

MOV BANKINDEX,R0

2221 011456 052760 100000 003016

BIS #BIT15,CONFIG(R0)

2222 011464

IF LOADHOME NE #1

2223 011474 012737 000004 002100

MOV #4,NOPAR

;PARITY ACTION = COUNT & IGNORE

2224 011502 013700 002710

MOV LOADHOME,R0

;DESTINATION BANK

2225 011506 012701 000001

MOV #1,R1

;SOURCE BANK

2226 011512 004737 042072

CALL BANKMOV

2227 011516 004737 042370

CALL NEWLOAD

;MAP KERNEL TO NEW LOADER BANK

2228 011522 005037 002100

CLR NOPAR

2229 011526 005037 002074

CLR PARCNT

2230 011532

END ;OF IF LOADHOME

2231

;COPY MAPPING FOR ANY SLAVES

2232 011532 012700 172340

MOV #KIPAR0,R0

2233 011536 012701 002550

MOV #KIO,R1

2234 011542 012705 000010

MOV #8,R5

2235 011546 012021

MOV (R0)+,(R1)+

1\$:

2236 011550 077502

SOB R5,1\$

2237 011552

END ;OF IF CPUBIT

2238

SET STOPOK

2239 011552

2242 011560

SUBTST <<MULTIPOINT HOOK>>

```

:*****
:*SUBTEST      MULTIPOINT HOOK
:*****
    
```

2243
 2244
 2245
 2246
 2247
 2248
 2249
 2250
 2251
 2252
 2253
 2254
 2255 011560 104424
 2256 011562
 2257 011574
 2258 011604 012700 000002
 2259 011610 000137 011656
 2260 011614
 2261 011614
 2262 011624 012700 000004
 2263 011630 000137 011656
 2264 011634
 2265 011634
 2266 011644 012700 000006
 2267 011650 000137 011656
 2268 011654
 2269 011654 005000
 2270
 2271 011656
 2272 011664 013702 177764
 2273 011670
 2274 011672 012777 100000 171036
 2275 011700 017702 171032
 2276 011704 072227 177770
 2277 011710 042702 177774
 2278 011714
 2279 011714 010260 002606
 2280
 2281
 2282 011720 010003
 2283 011722
 2284 011730
 2285 011734 052760 100000 002616
 2286 011742
 2287 011750 005337 002670
 2288 011754 062700 002636
 2289
 2290
 2291 011760 000137 177640
 2292 011764
 2293 011766 004737 061074
 2294 011772
 2295 011776 012777 000001 170732

```

:THIS IS WHERE WE SEPARATE THE MASTER FROM THE SLAVES.
:THE MASTER IS THE PROCESSOR RUNNING WITH LOCATION 'CPUBIT'
:EQUAL TO #MASTER (BIT4).
:
:SLAVES ARE STOPPED HERE.
:IF THERE IS AN IIST AND FIELD SERVICE IS ALLOWING US TO USE IT
: THE SLAVE LOWERS HIS PRIORITY AND WAITS FOR AN INTERRUPT
: FROM THE MASTER'S IIST.
:ELSE
: THE SLAVE MOVES INTO THE UIPAR'S AND EXECUTES A LITTLE PROGRAM
: THAT LOOKS FOR A COUNT PATTERN '10,9,8,7,6,5,4,3,2,1,0' IN HIS
: PSEUDO TTY BUFFER (PTYBUF+ID).
CACHOFF          :TURN CACHE OFF
IF MULTIPOINT IS FALSE THEN JUMPTO SUBAAR
IF CPUBIT EQ #SLAVE1
    MOV #2,R0
    JMP CPUID
END ;OF IF CPUBIT
IF CPUBIT EQ #SLAVE2
    MOV #4,R0
    JMP CPUID
END ;OF IF CPUBIT
IF CPUBIT EQ #SLAVE3
    MOV #6,R0
    JMP CPUID
END ;OF IF CPUBIT
CLR R0          ;MUST BE MASTER THEN
    
```

```

CPUID: IF NOIIST IS TRUE
    MOV SYSTID,R2
ELSE
    MOV #BIT15,@IISTACR
    MOV @IISTACR,R2
    ASH #-8,R2
    BIC #^C3,R2
END ;OF IF NOIIST
MOV R2,PORTDIR(R0)
    
```

```

:THIS IS WHERE THE SLAVES COME TO STOP
STOPCPU:MOV R0,R3
IF NOIIST IS TRUE
    IF R0 EQ #0 THEN GOTO MP1      :SKIP IF I'M THE MASTER
    BIS #BIT15,TASK(R0)
    BMOV WAITSTUFF
    DEC LOCK                       :ALLOW OTHERS TO RUN
    ADD #PTYBUF,R0                 :R0 POINTS TO COUNT DOWN LOCATION
:WAIT FOR MASTER TO COUNT DOWN!
:CALL TO THE USER INSTRUCTION PAR'S
    JMP FASTCITY
ELSE
    CALL IISTINIT
    IF R0 EQ #0 THEN GOTO MP1      :SKIP IF I'M THE MASTER
    MOV #1,@IISTACR
    
```

2296 012004 012777 000004 170726
2297 012012 000234
2298 012014 052760 100000 002616
2299 012022 005337 002670
2300 012026 000001
2301 012030 000137 050726
2302 012034

MOV #BIT2,@IISTADR ;ENABLE INTERRUPTS
SPL 4
BIS #BIT15,TASK(R0)
DEC LOCK ;ALLOW OTHERS TO RUN
WAIT ;WAIT IF I'M A SLAVE
JMP TASKDO
END ;OF IF NOIIST

2305 012034

WAITSTUFF:SUBTST <<SLAVES WAIT FOR MASTER TO COUNT DOWN>>
:*****
:*SUBTEST SLAVES WAIT FOR MASTER TO COUNT DOWN
:*****

2306 012034 012702 000014
2307 012040 012701 000013
2308 012044 005302
2309 012046 005301
2310 012050 020210
2311 012052 001776
2312 012054 020110
2313 012056 001366
2314 012060 005701
2315 012062 001370
2316 012064 000137 061216

1\$: MOV #12.,R2 ;V177640
MOV #11.,R1 ;V177644
2\$: DEC R2 ;V177650
DEC R1 ;V177652
3\$: CMP R2,(R0) ;V177654
BEQ 3\$;V177656
CMP R1,(R0) ;V177660
BNE 1\$;V177662
4\$: TST R1 ;V177664
BNE 2\$;V177666
JMP SLAVUP ;V177670

2319 012070

2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363

```
MP1:  SUBST <<BOOT UP ALL EXISTING CPU'S>>
:*****
:*SUBTEST  BOOT UP ALL EXISTING CPU'S
:*****
:ALL SLAVES ARE BOOTED AS FOLLOWS
:
:SETUP A SOFTWARE SWITCH REGISTER.
:IF WE ARE USING IIST'S.
:  BEGIN BOOTLOOP
:    FOR EACH OF 4 POSSIBLE CPU'S.
:      SETUP STACK VALUE LESS THAN PREVIOUS CPU.
:      CLEAR ACKNOWLEDGE.
:      POINT IIST INTERRUPT VECTOR TO START.
:MP2:  WAIT FOR THE MASTER'S IIST TO BE READY (SHOULD ALREADY BE).
:      INITIALIZE THE MASTER'S IIST (DISABLE BOOTS & INTERRUPTS).
:      INITIATE THE IIST BOOT OF THE SELECTED CPU.
:      WAIT FOR THE WINDOW IN THE BOOT THAT ALLOWS INTERRUPTS.
:      INITIATE THE IIST INTERRUPT OF THE SELECTED CPU.
:      WAIT FOR THE SLAVE TO ACKNOWLEDGE STARTING.
:      IF THE CPU DOES NOT ACKNOWLEDGE
:        IF THE CPU SELECTED IS NOT THE MASTER
:          LEAVE BOOTLOOP.
:        ELSE
:          GOTO MP2.
:      END OF IF THE CPU SELECTED IS NOT THE MASTER.
:    ELSE
:      WAIT FOR TASK DONE FROM SLAVE (TASK=START).
:      UPDATE SLAVE COUNT (THIS SLAVE IS THE HIGHEST # SO FAR).
:      IF THIS WAS THE 4TH CPU THEN LEAVE BOOTLOOP.
:    END OF IF THE CPU DOES NOT ACKNOWLEDGE.
:  END OF FOR EACH OF 4 POSSIBLE CPU'S.
:END OF BOOTLOOP.
:IF THERE WERE NO SLAVES
:  CLEAR THE MULTIPROCESSING FLAG 'MULTIPOINT'.
:  SET THE IIST VECTOR TO LOOK FOR UNEXPECTED TRAPS.
:END OF IF THERE WERE NO SLAVES.
:ELSE
:  FOR EACH SLAVE
:    SETUP STACK VALUE LESS THAN PREVIOUS CPU.
:    TYPE 'START SLAVE #N AT ADDRESS 200'.
:    WAIT FOR TASK DONE FROM SLAVE (TASK=START).
:  END OF FOR EACH SLAVE.
:END OF IF WE ARE USING IIST'S.
:RESTORE THE MASTER TO THE NORMAL SWITCH REGISTER
:WAIT A SECOND.
:MAKE SURE I'M THE ONLY PROCESSOR RUNNING.
:SETUP STACK VALUE BACK TO MASTER'S DEFAULT.
:TURN CACHE ON.
```

2366						
2367						.ENABL LSB
2368	012070	005337	002670			DEC LOCK ;ALLOW ONE SLAVE TO RUN
2369						;PASS PSEUDO SWR TO SLAVE
2370	012074	013737	002766	002674		MOV SWR,MASWR ;SAVE REAL SWR
2371	012102	012737	000176	002766		MOV #SWREG,SWR ;SETUP SOFTWARE SWR
2372	012110	017777	170560	170650		MOV @MASWR,@SWR ;FILL IT WITH REAL SWR STUFF
2373	012116					IF NOIIST IS TRUE THEN JUMPTO MP4
2374	012130	012702	000400			MOV #BIT8,R2 ;BOOT = BITS 11,10,9,8
2375	012134	012703	000001			MOV #1,R3 ;INTERRUPT = BITS 3,2,1,0
2376	012140					BEGIN BOOTLOOP
2377	012140					FOR R1 := #1 TO #3
2378	012144	162737	000200	002706		SUB #200,KSTACK ;SLAVES - PLEASE USE YOUR OWN STACK
2379	012152	013737	002706	000J00		MOV KSTACK,0 ;PUT IN LOC 0 FOR BOOTSTRAP
2380	012160	012700	000020			MOV #MASTER,R0 ;SET FIRST CPU ID BIT
2381	012164	072001				ASH R1,R0 ;SHIFT IT TO CORRECT SPOT
2382	012166	010037	002114			MOV R0,CPUBIT ;CPUBIT = BIT5 OR BIT6 OR BIT7
2383	012172	010100				MOV R1,R0 ;CPU # ---> R0
2384	012174	006300				ASL R0 ;R0 := 2 OR 4 OR 6 ;INDEX VALUE
2385	012176	005037	002672			CLR ACK ;ALLOW ONLY ONE CPU AT A TIME
2386	012202	013704	002742			MOV IISTVEC,R4
2387	012206	012724	004352			MOV #START,(R4)+ ;INTERRUPT VECTOR TO START
2388	012212	012714	000340			MOV #340,(R4) ;HIGH PRIORITY
2389						
2390						;WAIT FOR IIST READY
2391	012216	012737	061165	002574		MOV #5*SECONDS,HUNGTIME
2392	012224	012777	000001	170504	MP2:	MOV #1,@IISTACR
2393	012232	004737	072154			CALL QUE
2394	012236					ON.ERROR
2395	012240	104061				ERROR +61 ;IIST IS HANGING SYSTEM!
2396	012242	005037	002730			CLR SLAVES
2397	012246					LEAVE BOOTLOOP
2398	012250					END ;OF ON.ERROR
2399	012250	032777	004000	170462		BIT #BIT11,@IISTADR
2400	012256	001762				BEQ MP2
2401						
2402	012260	004737	061174			CALL IISTOFF ;INITIALIZE THE IIST & DISABLE BOOTS & INTERRUPTS
2403	012264	012777	000000	170444		MOV #0,@IISTACR ;SELECT THE ACCESS CONTROL REGISTER
2404	012272	010277	170442			MOV R2,@IISTADR ;BOOT SOME CPU
2405	012276	012777	000001	170434		MOV #1,@IISTADR ;NOW!
2406						;THIS DELAY IS GOOD ONLY FOR THE M9312 MULTIPROCESSOR BOOT
2407						;THIS DELAY ALSO ASSUMES THAT CACHE IS OFF
2408	012304	012704	140000			MOV #140000,R4 ;THIS DELAY TAKES APPROX 3/4 SECOND
2409	012310	000240			3\$:	NOP
2410	012312	000240				NOP
2411	012314	000240				NOP
2412	012316	000240				NOP
2413	012320	005237	000110			INC 110 ;CHANGE MOVING LOCATION - SEE BOOT LISTING
2414						
2415						
2416	012324	077407				SQB R4,3\$;THIS STOPS THE BOOT FROM RUNNING THE MEMORY ;DIAGNOSTIC AND BLOWING UP MY SOFTWARE!

D 7

```

2418 012326 012777 000000 170402
2419 012334 010377 170400
2420 012340 012777 000001 170372
2421
2422 012346 012705 000003
2423 012352 005004
2424 012354
2425 012362 005237 000110
2426
2427
2428 012366 077406
2429 012370 077507
2430 012372
2431 012372
2432 012400 006302
2433 012402 006303
2434 012404
2435 012412
2436 012414
2437 012416
2438 012420
2439 012420
2440 012422 005037 002672
2441 012426 012737 000005 002572
2442 012434 012737 177777 002574
2443 012442 004737 072154
2444 012446
2445 012450 162737 000001 002572
2446 012456
2447 012460 010137 002056
2448 012464 104060
2449 012466 005037 002730
2450 012472
2451 012474
2452 012474
2453 012474 005760 002616
2454 012500 100360
2455 012502 010137 002730
2456 012506
2457 012514
2458 012514 006302
2459 012516 006303
2460 012520
2461 012530
2462 012530
2463 012536 005037 003014
2464 012542 012777 036764 170172
2465 012550
2466 012550 000462
2467 012552
2468 012556 012700 000020
2469 012562 072001
2470 012564 010037 002114
2471 012570 162737 000200 002706
2472 012576 010100
2473 012600 006300
2474 012602
    
```

1\$:

2\$:

MP4:

```

MOV #0,@IISTACR ;SELECT THE ACCESS CONTROL REGISTER
MOV R3,@IISTADR ;INTERRUPT SOME CPU
MOV #1,@IISTADR ;NOW!
;WAIT FOR ACK OR TIMEOUT
MOV #3,R5 ;THIS IS APPROX A 3 SECOND DELAY
CLR R4
IF ACK IS FALSE
    INC 110 ;CHANGE MOVING LOCATION - SEE BOOT LISTING
                ;THIS STOPS THE BOOT FROM RUNNING THE MEMORY
                ;DIAGNOSTIC AND BLOWING UP MY SOFTWARE!
SOB R4,1$
SOB R5,1$
END ;OF IF ACK IS FALSE ;IF THERE IS NO SLAVE AT THIS IISTID #
IF ACK IS FALSE
    ASL R2
    ASL R3
    IF R3 EQ #BIT4
        LEAVE BOOTLOOP
    ELSE
        GOTO MP2
    END ;OF IF R3
ELSE
    CLR ACK ;ALLOW SLAVE TO PROCEED
    MOV #5,HUNGMSB
    MOV #-1,HUNGTIME
    CALL QUE
    ON.ERROR
        SUB #1,HUNGMSB
        ON.ERROR
            MOV R1,BAD
            ERROR +60 ;SLAVE IS HANGING SYSTEM!
            CLR SLAVES
            LEAVE BOOTLOOP
        END ;OF ON.ERROR
    END ;OF ON.ERROR
    TST TASK(R0)
    BPL 2$
    MOV R1,SLAVES ;COUNT OF SLAVES WAS IN R1
    IF R3 EQ #BIT3 THEN LEAVE BOOTLOOP
END ;OF IF ACK
ASL R2
ASL R3
END ;OF FOR R1
END BOOTLOOP
IF SLAVES EQ #0
    CLR MULTIPOINT
    MOV #IITRAP,@IISTVEC ;RESTORE UNEXPECTED IIST TRAP
END ;OF IF SLAVES
BR MP3
FOR R1 := #1 TO SLAVES
    MOV #BIT4,R0
    ASH R1,R0
    MOV R0,CPUBIT ;CPUBIT = BITS OR BIT6 OR BIT7
    SUB #200,KSTACK
    MOV R1,R0
    ASL R0 ;R0 = 2 OR 4 OR 6
    TYPE MSG081 ;START SLAVE #
    
```

```

2475 012606          TYPOCS          R1,,1
2476 012614          TYPE MSG082          ;AT ADDRESS 200
2477 012620 012737 000031 002572      MOV #25,,HUNGMSB
2478 012626 012737 177777 002574      MOV #-1,HUNGTIME
2479 012634 005037 002672          5$: CLR ACK          ;ALLOW SLAVE TO PROCEED
2480 012640 004737 072154          CALL QUE
2481 012644          ON.ERROR
2482 012646 162737 000001 002572      SUB #1,HUNGMSB
2483 012654          ON.ERROR
2484 012656 010137 002056          MOV R1,BAD
2485 012662 104060          ERROR +60          ;SLAVE IS HANGING SYSTEM!
2486 012664          CLEAR SLAVES,MULTIPORT
2487 012674 000137 012716          JMP MP3
2488 012700          END ;OF ON.ERROR
2489 012700          END ;OF ON.ERROR
2490 012700 005760 002616          TST TASK(R0)
2491 012704 100353          BPL 5$
2492 012706          END ;OF FOR R1
2493          ;POINT THE MASTER BACK TO THE NORMAL SWR
2494 012716 013737 002674 002766 MP3: MOV MASWR,SWR
2495 012724 012737 000020 002114      MOV #MASTER,CPUBIT
2496 012732 052737 000020 002510      BIS #MASTER,ALLCPUS
2497          ;WAIT A SECOND
2498 012740 005000          CLR R0
2499 012742 077001          4$: SOB R0,4$
2500          ;STOP ANY CONFUSED MASTER (ME)
2501 012744 005237 002670          INC LOCK
2502 012750          IF LOCK NE #1
2503 012760 000000          MHALT3: HALT
2504 012762 000776          BR MHALT3
2505 012764          END ;OF IF LOCK
2506 012764 012737 000020 002114 MP5: MOV #MASTER,CPUBIT          ;BACK TO MASTER
2507 012772 012737 002000 002706      MOV #STACK,KSTACK
2508 013000 104423          CACHON          ;TURN CACHE ON
2509          .DSABL LSB
    
```

2512 013002

2513 013002 104424
2514 013004
2515 013004 013702 002730
2516 013010 006302
2517 013012
2518 013014
2519 013016
2520 013022
2521 013032 104041
2522 013034
2523 013040
2524 013042 005037 003014
2525 013046
2526 013046
2527 013046
2528 013056
2529 013066
2530 013066 104423
2531
2532 013070

2533 013070
2534 013076
2535 013102
2536 013112 112737 000060 115753
2537 013120
2538 013124 010001
2539 013126 006301
2540 013130 105237 115753
2541 013134
2542 013140
2543 013150
2544 013160
2545
2546 013160

2547 013160
2548 013174 013701 002334
2549 013200 013737 075214 002144
2550 013206 013737 075216 002146
2551 013214
2552 013220 106301
2553 013222
2554 013224 104425
2555 013226
2556 013226
2557 013244 005237 075142
2558 013250 000137 015050
2559 013254

```

SUBTST <<CHECK FOR UNIQUE CPU ID NUMBERS>>
:*****
:*SUBTEST CHECK FOR UNIQUE CPU ID NUMBERS
:*****
CACHOFF ;TURN CACHE OFF
BEGIN SAMESAME
MOV SLAVES,R2
ASL R2
FOR RO := #0 TO R2 BY #2
FOR R1 := #0 TO R2 BY #2
IF RO NE R1
IF PORTDIR(RO) EQ PORTDIR(R1)
ERROR +41
TYPE MSG084 ;PROCEEDING AS A SINGLE PORT TEST
LEAVE SAMESAME
CLR MULTIPOINT
END ;OF IF PORTDIR(RO)
END ;OF IF RO
END ;OF FOR R1
END ;OF FOR RO
END SAMESAME
CACHON ;TURN CACHE ON

SUBTST <<PRINT MULTIPOINT DIRECTORY>>
:*****
:*SUBTEST PRINT MULTIPOINT DIRECTORY
:*****
IF MULTIPOINT IS TRUE
TYPE MSG086
TYPOCS PORTDIR
MOVB #'0,MSG087
FOR RO := #1 TO SLAVES
MOV RO,R1
ASL R1
INCB MSG087
TYPE MSG087
TYPOCS PORTDIR(R1)
END ;OF FOR RO
END ;OF IF MULTIPOINT

SUBAAR: SUBTST <<LOAD CSR'S FROM E-TABLE $CDW1 & $CDW2>>
:*****
:*SUBTEST LOAD CSR'S FROM E-TABLE $CDW1 & $CDW2
:*****
IF $CDW1 NE #0 OR $CDW2 NE #0
MOV MKCSRS,R1
MOV $CDW1,CSR
MOV $CDW2,CSR+2
FOR CSRNO := #0 TO #34 BY #2
ASLB R1
ON.ERROR
LOADCSR
END ;OF ON.ERROR
END ;OF FOR CSRNO
INC $PASS
JMP APTHANG ;GO LOOP FOREVER KEEPING APT ALIVE
END ;OF IF $CDW1
```

```

2562 013254
2563 013254
2564 013264
2565 013270 004737 042422
2566 013274 013700 002110
2567
2568
2569 013300
2570
2571 013306
2572 013316 013737 002510 002114
2573 013324 004737 042422
2574 013330 012737 000020 002114
2575
2576 013336
2577 013352
2578 013360
2579 013400
2580 013416
2581 013434
2582 013452
2583 013470 104063
2584 013472
2585 013472
2586 013472
2587
2588
2589 013472 004737 042422
2590 013476
2591 013512 016001 003016
2592 013516 042701 177770
2593 013522
2594 013532 006301
2595 013534
2596 013534
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609 013542 005004
2610 013544 116002 003022
2611 013550 012703 000010
2612 013554 006002
2613 013556 005504
2614 013560 077303
2615
    
```

```

SUBTST <<LEGAL CONFIGURATION CHECK>>
*****
*SUBTEST      LEGAL CONFIGURATION CHECK
*****
IF CPUBIT EQ #MASTER
FOR BANK := #0 TO #167
CALL      EXBANK
MOV BANKINDEX,R0

:MAKE SURE THAT EVERYBODY CAN ACCESS EVERY BANK
IF MKFLAG IS TRUE
:IF ANYBODY CAN ACCESS
IF #MASTER!SLAVE1!SLAVE2!SLAVE3 SET.IN CONFIG(R0)
MOV      ALLCPUS,CPUBIT
CALL      EXBANK
MOV      #MASTER,CPUBIT
:IF EVERYBODY CAN'T ACCESS
IF ACFLAG IS FALSE AND CONFGERROR IS FALSE
SET CONFGERROR
CLEAR SLFLAG,SLFLAG+2,SLFLAG+4,SLFLAG+6
IF #MASTER SET.IN CONFIG(R0) THEN LET SLFLAG := ONES
IF #SLAVE1 SET.IN CONFIG(R0) THEN LET SLFLAG+2 := ONES
IF #SLAVE2 SET.IN CONFIG(R0) THEN LET SLFLAG+4 := ONES
IF #SLAVE3 SET.IN CONFIG(R0) THEN LET SLFLAG+6 := ONES
ERROR +63
END :OF IF ACFLAG
END :OF IF #MASTER!SLAVE1!SLAVE2!SLAVE3
END :OF IF MKFLAG IS TRUE

:MAKE SURE (IF I CAN) THAT EACH BOX HAS ONE & ONLY ONE CSR
CALL      EXBANK
IF ACFLAG IS TRUE AND MKFLAG IS TRUE
MOV      CONFIG(R0),R1
BIC      #^C7,R1          ;R1 = 0 OR 1 OR 2 OR 4
IF #BIT12 OFF.IN CONFIG+2(R0) ;IF NO INTERNAL INTERLEAVE
ASL      R1              ;R1 = 0 OR 2 OR 4
END :OF IF #BIT12
IF R1 EQ #0 THEN LET R1 := R1 + #1
:NOW
:INTERLEAVE          R1          INTERLEAVE FACTOR
-----
:4 TO 1 EXT & INTERNAL      4          8
:4 TO 1 EXT & NO INT        4          4
:2 TO 1 EXT & INTERNAL      2          2
:2 TO 1 EXT & NO INT        2          2
:      NO EXT & INTERNAL    1          1
:      NO EXT & NO INT      1          0
:SO
:R1 = 1 OR 2 OR 4 = '# OF CSR'S ON THIS BANK'
CLR      R4
MOVB     CONFIG+4(R0),R2
MOV      #8.,R3
ROR      R2
ADC      R4
SOB     R3,1$          ;COUNT CSR'S IN THIS BANK
    
```

1\$:

2616
2617 013562
2618 013574
2619 013602 104064
2620 013604
2621 013604
2622 013604
2623 013620

:IF # OF CSR'S IS WRONG!
IF R4 NE R1 AND CONFGERROR IS FALSE
SET CONFGERROR
ERROR +64
END :OF IF R4
END :OF IF ACFLAG
END :OF FOR BANK
END :OF IF CPUBIT

2626 013620

SUBSTST <<PRINT CONFIGURATION DETAILS>>

: *SUBTEST PRINT CONFIGURATION DETAILS

2627 013620 005037 002466
2628 013624 005037 002470
2629 013630
2630 013632
2631 013642
2632 013656
2633 013676
2634 013702
2635 013702
2636 013702
2637 013714 006337 002466
2638 013720 006337 002466
2639 013724 006337 002466
2640 013730 006337 002466
2641 013734 006337 002470
2642 013740 006337 002470
2643 013744 006337 002470
2644 013750 006337 002470
2645 013754 005237 002516
2646 013760 006337 002516
2647 013764 006337 002516
2648 013770 006337 002516
2649 013774 006337 002516
2650 014000
2651 014014
2652 014032
2653 014036
2654 014044
2655 014050
2656 014056
2657 014062
2658 014070
2659 014074
2660 014112 004737 035402
2661 014116

```
CLR MJSIZE
CLR MKSIZE
FOR R1 := #0 TO #167*3*2 BY #6
  IF CPUBIT SET.IN CONFIG(R1)
    IF #BIT14!BIT13 OFF.IN CONFIG(R1) THEN LET MJSIZE := MJSIZE + #1
    IF #BIT14 OFF.IN CONFIG(R1) AND #BIT13 SET.IN CONFIG(R1)
      LET MKSIZE := MKSIZE + #1
    END ;OF IF MK11 MEMORY
  END ;OF IF ACCESSABLE BY CPU
END ;OF FOR ALL BANKS IN TABLE
ASL MJSIZE
ASL MJSIZE
ASL MJSIZE
ASL MJSIZE ;MJSIZE := MJSIZE * 16.
ASL MKSIZE
ASL MKSIZE
ASL MKSIZE
ASL MKSIZE ;MKSIZE := MKSIZE * 16.
INC UNITOP
ASL UNITOP
ASL UNITOP
ASL UNITOP
ASL UNITOP ;UNITOP := UNITOP * 16.
LET I := MJSIZE + MKSIZE
IF I LT UNITOP THEN LET I := UNITOP
TYPE $CRLF
TYPDEC I
TYPE MSG070
TYPDEC MJSIZE
TYPE MSG071
TYPDEC MKSIZE
TYPE MSG072
IF #SW6 SET.IN @SWR OR CONFGERROR IS TRUE
  CALL PCONFIG
END ;OF IF #SW6
```

2664 014116

2665 014116

2666 014132 005037 002534

2667 014136 012700 075164

2668 014142

2669 014144

2670 014152 111001

2671 014154 042701 177400

2672 014160

2673 014166 000261

2674 014170

2675 014172 000241

2676 014174

2677 014174 006101

2678 014176 005201

2679 014200 006301

2680 014202 006301

2681 014204 006301

2682 014206 006301

2683 014210 163701 002534

2684 014214 010137 002534

2685 014220

2686 014230 060137 002522

2687 014234

2688 014234

2689 014244 060137 002524

2690 014250

2691 014250 062700 000004

2692 014254

2693 014254

2694 014264

2695 014304 104051

2696 014306

2697

2698

2699 014306 013700 002730

2700 014312 005200

2701 014314

2702 014322 010037 002050

2703 014326 013737 075234 002056

2704 014334 104065

2705 014336

2706 014336

2707

2708 014336 004737 015104

```

SUBSTST <<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
      MOV  (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
      SUB  TEMP,R1
      MOV  R1,TEMP
      IFB 1(R0) EQ #1
        ADD  R1,APTCORE
      END ;OF IFB 1(R0)
      IFB 1(R0) EQ #4
        ADD  R1,APTMS
      END ;OF IFB 1(R0)
      ADD  #4,R0
    END ;OF IFB 1(R0)
  END ;OF FOR R2
  IF APTCORE NE MJSIZE OR APTMS NE MKSIZE
    ERROR +51
  END ;OF IF APTCORE

;CHECK FOR CORRECT NUMBER OF CPU'S
MOV  SLAVES,R0
INC  R0 ;R0 = 1, 2, 3, OR 4
IF $DDW6 NE R0
  MOV R0,GOOD
  MOV $DDW6,BAD
  ERROR +65
END ;OF $DDW6
END ;OF IF APTFLAG

CALL  DOBACK ;WRITE BACKGROUND PATTERN
    
```

2710 014342

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

014342 000004
2711 014344 005037 002332
2712 014350 017700 166412
2713 014354 042700 177761
2714 014360 004770 014370
2715 014364 000137 014410
2716 014370 015204
2717 014372 015320
2718 014374 015434
2719 014376 015572
2720 014400 015730
2721 014402 016066
2722 014404 016246
2723 014406 016404
2724
2725

TST5: SCOPE
CLR CONTFLAG
MOV @SWR,R0 ;GET SWITCHES
BIC #^C16,R0 ;MASK TO ONLY MODE BITS
CALL @DISPTBL(R0) ;DISPATCH TO ROUTINE THROUGH NEXT TABLE
JMP MEMDONE ;GO TO NEXT TEST
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
:**NOTE** MARGINS FOLLOW PATTERNS (FORWARD IS (0,2,3,4,5))
:(REVERSE IS (5,4,3,2,0))

2729 014410 004737 015104
 2730 014414
 014414 000004
 2731
 2732
 2733 014416
 2734 014424
 2735 014440 004737 024370
 2736 014444
 2737 014452 005037 002116
 2738 014456
 2739
 2740 014456

014456 000004
 2741
 2742
 2743
 2744
 2745
 2746
 2747
 2748 014460
 2749 014466 012700 060000
 2750 014472 012737 000004 002100
 2751 014500
 2752 014506
 2753 014512 004737 042422
 2754 014516
 2755 014524 005037 002074
 2756 014530 004737 026042
 2757 014534
 2758 014544 104035
 2759 014546
 2760 014546
 2761 014546
 2762 014562 005037 002100
 2763 014566
 2764 014566 000405
 2765
 2766
 2767
 2768 014570 012737 000002 177750
 2769 014576 005710
 2770 014600 000207

```

MEMDONE:CALL DOBACK ;CHECK BACKGROUND PATTERN
NEWSTST<<UNIQUE BANK TEST>>
:*****
:*TEST 6 UNIQUE BANK TEST
:*****
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 MTO027
SET HEADER
CLR MUT
END ;OF IF SELONLY

NEWSTST <<FORCE ADDRESS PARITY ERRORS>>
:*****
:*TEST 7 FORCE ADDRESS PARITY ERRORS
:*****
TST7: SCOPE
:THIS MAKES SURE THAT ALL ACCESSABLE MEMORY LOCATIONS WILL GIVE A
:PARITY TRAP IF I FORCE A MEMORY BUSS ADDRESS PARITY ERROR BY
:USING THE MAINTENANCE REGISTER.
:
:WHEN DONE EVERY ADDRESS PARITY ERROR LED IN EVERY MEMORY BOX
:(MK11'S, MKA11'S, AND MJ11'S) SHOULD BE LITE.
:YOU CAN ONLY TURN THEM OFF BY TURNING OFF POWER TO THE BOX'S.
IF SELONLY IS FALSE
MOV #FIRST,R0
MOV #4,NOPAR ;INDICATE PARITY ACTION
BMOV ADDPAR
FOR BANK := #0 TO #167
CALL EXBANK
IF ACFLAG IS TRUE
CLR PARCNT
CALL SUPD01
IF PARCNT NE #1
ERROR +35
END ;OF IF PARCNT
END ;OF IF ACFLAG
END ;OF FOR BANK
CLR NOPAR ;INDICATE PARITY ACTION
END ;OF IF SELONLY
BR SUBAAU

;THESE 3 INSTRUCTIONS ARE MOVED TO THE USER PAR'S IN ORDER TO
;CAUSE ABORTS FROM THE BANK UNDER TEST VICE THE PROGRAM SPACE
ADDPAR: MOV #BIT1,MAINT ;V177640 ;CREATE ADDRESS PARITY ERROR
TST (R0) ;V177646 ;THIS WILL ABORT & INCREMENT PARCNT
RETURN ;V177650
    
```

2773 014602
2774 014610
2775 014620 104050
2776 014622
2777 014622
2778
2779 014622
2780 014630 005137 002744
2781 014634 004737 015104
2782 014640 005037 002502
2783 014644
2784 014656

```
SUBAAU: IF SBFLAG IS TRUE
        IF LASTERROR EQ $ERTTL
            ERROR +50
        END ;OF IF LASTERROR
    END ;OF IF SBFLAG

    SET WRITEONLY
    COM SOFTPAT
    CALL DOBACK ;RESTORE BACKGROUND PATTERN
    CLR WRITEONLY
    IF MULTIPOST IS TRUE THEN JUMPTO MAST01
```

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

2785 014656 004737 024754

2789
 2790
 2791
 2792
 2793
 2794
 2795
 2796 014662 005037 002022
 2797 014666 005037 002542
 2798 014672 013737 002754 002020
 2799 014700 005237 075142
 2800 014704 042737 100000 075142
 2801 014712
 2802 014716
 2803 014734
 2804 014740 005037 002440
 2805 014744
 2806 014744
 2807 014752 013700 000042
 2808 014756 001430
 2809 014760 022700 002000
 2810 014764 001425
 2811
 2812 014766
 2813 014770 004737 043554
 2814 014774
 2815 014776 000005
 2816 015000 004710
 2817 015002 000240
 2818 015004 000240
 2819 015006 000240
 2820 015010
 2821
 2822
 2823
 2824
 2825 015010 013706 002706
 2826 015014 052737 000060 172516
 2827 015022 104420
 2828 015024 013700 002710
 2829 015030 012701 000001
 2830 015034 004737 042072
 2831 015040
 2832 015050 012701 000050
 2833 015054 077001
 2834 015056 062737 000001 075144
 2835 015064 005537 075146
 2836 015070 077107
 2837 015072 005237 075142
 2838 015076 000764
 2839 015100
 2840 015100 000137 014342

```

.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 SBFLAG
      CLR FSINFLAG
      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, R0 ;; GET MONITOR ADDRESS
      BEQ $DOAGAIN ;; BRANCH IF NO MONITOR
      $ZAP42: CMP #STACK, R0 ;; ARE WE UNDER RT11
             BEQ $DOAGAIN ;; YES - BRANCH
             ;WE ARE UNDER (HEAVEN HELP US) XXDP!
             PUSH R0
             CALL SHUTUP
             POP R0
             RESET ;; CLEAR THE WORLD
             $SENDAD: CALL (R0) ;; 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
                     BIS #BITS!BIT4, MMR3
                     ENERGIZE ; TURN ON MEMORY MANAGEMENT
                     MOV LOADHOME, R0 ; DESTINATION BANK
                     MOV #1, R1 ; SOURCE BANK
                     CALL BANKMOV
             $DOAGAIN: IF $USWR EQ $PASS ; IF APT LIMITING # OF PASSES
             APTHANG: MOV #50, R1
             2$: SOB R0, 2$
                  ADD #1, $DEVCT
                  ADC $UNIT
                  SOB R1, 2$
                  INC $PASS
                  BR APTHANG
             END ;OF IF $USWR
             JMP LOOP ;RETURN
    
```

2843 015104

2844 015104 005037 002120
2845 015110
2846 015114 004737 042422
2847 015120
2848 015134
2849 015150 004737 021222
2850 015154 005037 002116
2851 015160
2852 015166
2853 015166
2854 015202 000207

```
DOBACK: SUBTST <<WRITE BACKGROUND PATTERNS>>
:*****
:*SUBTEST WRITE BACKGROUND PATTERNS
:*****
      CLR PATTERN
      FOR BANK := #0 TO #167
      CALL EXBANK
      IF ACFLAG IS TRUE AND PFLAG IS FALSE
      SET HEADER,MUT
      CALL MKTEST ;CALL MJTEST WOULD ALSO WORK
      CLR MUT
      SET HEADERR
      END ;OF IF ACFLAG
      END ;OF FOR BANK
      RETURN
```

2857
2858
2859 015204

2860 015204 005037 002106
2861
2862 015210 004737 042422
2863 015214 005737 002124
2864 015220 001417
2865 015222 005737 002132
2866 015226 001014
2867 015230 005037 002120
2868
2869 015234 005037 002112
2870
2871 015240 004737 016564
2872
2873 015244 004737 043006
2874 015250 001373
2875
2876 015252 004737 042746
2877 015256 001366
2878
2879 015260 004737 042772
2880 015264 001351
2881
2882 015266 005737 002134
2883 015272 001401
2884 015274 000207
2885 015276 004737 041130
2886 015302
2887
2888 015306 004737 015204
2889 015312 004737 041604
2890 015316 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$: CLR MARGIN ;SET MARGIN TO 0
      :START OF MARGIN LOOP
3$: CALL MTEST ;GO TEST CORRECT MEMORY
      :TERMINATION OF MARGIN LOOP
      CALL INCMAR ;NEXT HIGHER MARGIN
      BNE 3$ ;IF NOT DONE - LOOP ON THIS MARGIN
      :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$: CALL INCBNK ;NEXT HIGHER BANK
      BNE 1$ ;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
```

2893 015320

2894 015320 005037 002106

2895

2896 015324 004737 042422

2897 015330 005737 002124

2898 015334 001417

2899 015336 005737 002132

2900 015342 001014

2901 015344 004737 042762

2902

2903 015350 004737 043212

2904

2905 015354 004737 016564

2906

2907 015360 004737 043152

2908 015364 100373

2909

2910 015366 005337 002120

2911 015372 100366

2912

2913 015374 004737 042772

2914 015400 001351

2915

2916 015402 005737 002134

2917 015406 001401

2918 015410 000207

2919 015412 004737 041130

2920 015416

2921

2922 015422 004737 015320

2923 015426 004737 041604

2924 015432 000207

```

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
2$: CALL SETMAR ;SET HIGH MARGINS (IF NOT INHIBITED)
      ;START OF MARGIN LOOP
3$: CALL MTEST ;GO TEST CORRECT MEMORY
      ;TERMINATION OF MARGIN LOOP
      CALL DECMAR ;NEXT LOWER MARGIN
      BPL 3$ ;IF NOT DONE - LOOP ON THIS MARGIN
      ;TERMINATION OF PATTERN LOOP
      DEC PATTERN ;IS THIS THE LAST PATTERN?
      BPL 2$ ;NO - LOOP ON THIS PATTERN
      ;TERMINATION OF BANK LOOP
4$: CALL INCBNK ;NEXT HIGHER BANK
      BNE 1$ ;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

```

2927 015434
 2928 015434 005037 002106
 2929
 2930 015440 004737 042422
 2931 015444 005737 002124
 2932 015450 001422
 2933 015452 005737 002136
 2934 015456 001417
 2935 015460 005737 002132
 2936 015464 001014
 2937 015466 005037 002120
 2938
 2939 015472 005037 002112
 2940
 2941 015476 004737 016564
 2942
 2943 015502 004737 043006
 2944 015506 001373
 2945
 2946 015510 004737 042746
 2947 015514 001366
 2948
 2949 015516 004737 042772
 2950 015522 001346
 2951
 2952 015524 005137 002712
 2953 015530 001003
 2954
 2955 015532 004737 015434
 2956 015536 000207
 2957 015540 005737 002134
 2958 015544 001401
 2959 015546 000207
 2960 015550 004737 041130
 2961 015554
 2962
 2963 015560 004737 015434
 2964 015564 004737 041604
 2965 015570 000207

```

BAWPAF: SUBTST <<BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST     BANKS WORST FIRST,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     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$:      CLR     MARGIN         ;SET MARGIN TO 0
          ;START OF MARGIN LOOP
3$:      CALL    MTEST          ;GO TEST CORRECT MEMORY
          ;TERMINATION OF MARGIN LOOP
          CALL    INCMAR        ;NEXT HIGHER MARGIN
          BNE     3$             ;IF NOT DONE - LOOP ON THIS MARGIN
          ;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$:      CALL    INCBNK        ;NEXT HIGHER BANK
          BNE     1$             ;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
    
```

2968 015572

2969 015572 005037 002106
2970
2971 015576 004737 042422
2972 015602 005737 002124
2973 015606 001422
2974 015610 005737 002136
2975 015614 001417
2976 015616 005737 002132
2977 015622 001014
2978 015624 004737 042762
2979
2980 015630 004737 043212
2981
2982 015634 004737 016564
2983
2984 015640 004737 043152
2985 015644 100373
2986
2987 015646 005337 002120
2988 015652 100366
2989
2990 015654 004737 042772
2991 015660 001346
2992
2993 015662 005137 002712
2994 015666 001003
2995
2996 015670 004737 015572
2997 015674 000207
2998 015676 005737 002134
2999 015702 001401
3000 015704 000207
3001 015706 004737 041130
3002 015712
3003
3004 015716 004737 015572
3005 015722 004737 041604
3006 015726 000207

```

BAWPAR: SUBTST <<BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**>>
:*****
:*SUBTEST      BANKS WORST FIRST,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      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     SETPAT        ;SET HIGH PATTERN FOR CORRECT MEMORY
          ;START OF PATTERN LOOP
2$:      CALL     SETMAR        ;SET HIGH MARGINS (IF NOT INHIBITED)
          ;START OF MARGIN LOOP
3$:      CALL     MTEST         ;GO TEST CORRECT MEMORY
          ;TERMINATION OF MARGIN LOOP
          CALL     DECMAR        ;NEXT LOWER MARGIN
          BPL      3$            ;IF NOT DONE - LOOP ON THIS MARGIN
          ;TERMINATION OF PATTERN LOOP
          DEC      PATTERN       ;IS THIS THE LAST PATTERN?
          BPL      2$            ;NO - LOOP ON THIS PATTERN
          ;TERMINATION OF BANK LOOP
4$:      CALL     INCBNK        ;NEXT HIGHER BANK
          BNE      1$            ;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     BAWPAR        ;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     BAWPAR        ;CALL SELF
          CALL     UNRELOCATE    ;UNMOVE & UNMAP PROGRAM
          RETURN

```

```

3010 015730 PAFBAF: SUBTST <<PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**
:*****
3011 015730 005037 002120 CLR PATTERN ;SET PATTERN TO 0
3012 :START OF PATTERN LOOP
3013 015734 005037 002106 1$: CLR BANK ;SET BANK TO 0
3014 :START OF BANK LOOP
3015 015740 004737 042422 2$: CALL EXBANK ;EXAMINE BANK
3016 015744 004737 042730 CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
3017 015750 001015 BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
3018 015752 005737 002124 TST ACFLAG ;CAN WE ACCESS THIS BANK?
3019 015756 001412 BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
3020 015760 005737 002132 TST RRFLAG ;RELOCATION REQUIRED?
3021 015764 001007 BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
3022 015766 005037 002112 CLR MARGIN ;SET MARGIN TO 0
3023 :START OF MARGIN LOOP
3024 015772 004737 016564 3$: CALL MTEST ;GO TEST CORRECT MEMORY
3025 :TERMINATION OF MARGIN LOOP
3026 015776 004737 043006 CALL INCMAR ;NEXT HIGHER MARGIN
3027 016002 001373 BNE 3$ ;IF NOT DONE - LOOP ON THIS MARGIN
3028 :TERMINATION OF BANK LOOP
3029 016004 004737 042772 4$: CALL INCBNK ;NEXT HIGHER BANK
3030 016010 001353 BNE 2$ ;IF NOT DONE - LOOP ON THIS BANK
3031 :TERMINATION OF PATTERN LOOP
3032 016012 004737 042746 CALL INCRPT ;NEXT HIGHER PATTERN
3033 016016 001346 BNE 1$ ;OK - LOOP; ELSE CONTINUE
3034 :END OF LOOPS
3035 016020 005137 002140 COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
3036 :IS THIS AN EVEN NUMBER PASS?
3037 016024 001403 BEQ 5$ ;YES - SKIP
3038 :**NOTE** RECURSIVE CALL
3039 016026 004737 015730 CALL PAFBAF ;CALL SELF
3040 016032 000207 RETURN
3041 016034 005737 002134 5$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
3042 016040 001401 BEQ 6$ ;NO - SKIP
3043 016042 000207 RETURN ;YES - RETURN
3044 016044 004737 041130 6$: CALL RELOCATE ;MOVE & MAP PROGRAM
3045 016050 ON.ERROR THEN $RETURN
3046 :**NOTE** RECURSIVE CALL
3047 016054 004737 015730 CALL PAFBAF ;CALL SELF
3048 016060 004737 041604 CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
3049 016064 000207 RETURN
    
```

3052 016066
 3053 016066 005037 002120
 3054
 3055 016072 005037 002106
 3056
 3057 016076 004737 042422
 3058 016102 004737 042730
 3059 016106 001020
 3060 016110 005737 002124
 3061 016114 001415
 3062 016116 005737 002136
 3063 016122 001412
 3064 016124 005737 002132
 3065 016130 001007
 3066 016132 005037 002112
 3067
 3068 016136 004737 016564
 3069
 3070 016142 004737 043006
 3071 016146 001373
 3072
 3073 016150 004737 042772
 3074 016154 001350
 3075
 3076 016156 004737 042746
 3077 016162 001343
 3078
 3079 016164 005137 002140
 3080
 3081 016170 001403
 3082
 3083 016172 004737 016066
 3084 016176 000207
 3085 016200 005137 002712
 3086 016204 001003
 3087
 3088 016206 004737 016066
 3089 016212 000207
 3090 016214 005737 002134
 3091 016220 001401
 3092 016222 000207
 3093 016224 004737 041130
 3094 016230
 3095
 3096 016234 004737 016066
 3097 016240 004737 041604
 3098 016244 000207

```

PAFBAW: SUBSTST <<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
      CLR     MARGIN     ;SET MARGIN TO 0
      ;START OF MARGIN LOOP
3$:   CALL     MTEST      ;GO TEST CORRECT MEMORY
      ;TERMINATION OF MARGIN LOOP
      CALL     INCMAR     ;NEXT HIGHER MARGIN
      BNE     3$          ;IF NOT DONE - LOOP ON THIS MARGIN
      ;TERMINATION OF BANK LOOP
4$:   CALL     INCBANK    ;NEXT HIGHER BANK
      BNE     2$          ;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
      BEQ     5$          ;IS THIS AN EVEN NUMBER PASS?
      ;**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
  
```

3101 016246
 3102 016246 004737 042762
 3103
 3104 016252 005037 002106
 3105
 3106 016256 004737 042422
 3107 016262 004737 042730
 3108 016266 001015
 3109 016270 005737 002124
 3110 016274 001412
 3111 016276 005737 002132
 3112 016302 001007
 3113 016304 004737 043212
 3114
 3115 016310 004737 016564
 3116
 3117 016314 004737 043152
 3118 016320 100373
 3119
 3120 016322 004737 042772
 3121 016326 001353
 3122
 3123 016330 005337 002120
 3124 016334 100346
 3125
 3126 016336 005137 002140
 3127
 3128 016342 001403
 3129
 3130 016344 004737 016246
 3131 016350 000207
 3132 016352 005737 002134
 3133 016356 001401
 3134 016360 000207
 3135 016362 004737 041130
 3136 016366
 3137
 3138 016372 004737 016246
 3139 016376 004737 041604
 3140 016402 000207

```

PARBAF: SUBTST <<PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**>>
:*****
:*SUBTEST      PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**
:*****
CALL      HIPAT      ;SET HIGHEST PATTERNS
;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     SETMAR     ;SET HIGH MARGINS (IF NOT INHIBITED)
;START OF MARGIN LOOP
3$: CALL     MTEST    ;GO TEST CORRECT MEMORY
;TERMINATION OF MARGIN LOOP
CALL     DECMAR     ;NEXT LOWER MARGIN
BPL     3$           ;IF NOT DONE - LOOP ON THIS MARGIN
;TERMINATION OF BANK LOOP
4$: CALL     INCBNK   ;NEXT HIGHER BANK
BNE     2$           ;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     PARBAF     ;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     PARBAF     ;CALL SELF
CALL     UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN
    
```

3143 016404

PARBAW: SUBSTST <<PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**>>
 :*****
 :*SUBTEST PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**
 :*****

3144 016404 004737 042762
 3145
 3146 016410 005037 002106
 3147
 3148 016414 004737 042422
 3149 016420 004737 042730
 3150 016424 001020
 3151 016426 005737 002124
 3152 016432 001415
 3153 016434 005737 002136
 3154 016440 001412
 3155 016442 005737 002132
 3156 016446 001007
 3157 016450 004737 043212
 3158
 3159 016454 004737 016564
 3160
 3161 016460 004737 043152
 3162 016464 100373
 3163
 3164 016466 004737 042772
 3165 016472 001350
 3166
 3167 016474 005337 002120
 3168 016500 100343
 3169
 3170 016502 005137 002140
 3171
 3172 016506 001403
 3173
 3174 016510 004737 016404
 3175 016514 000207
 3176 016516 005137 002712
 3177 016522 001003
 3178
 3179 016524 004737 016404
 3180 016530 000207
 3181 016532 005737 002134
 3182 016536 001401
 3183 016540 000207
 3184 016542 004737 041130
 3185 016546
 3186
 3187 016552 004737 016404
 3188 016556 004737 041604
 3189 016562 000207

```

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 SETMAR ;SET HIGH MARGINS (IF NOT INHIBITED)
;START OF MARGIN LOOP
3$: CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF MARGIN LOOP
CALL DECMAR ;NEXT LOWER MARGIN
BPL 3$ ;IF NOT DONE - LOOP ON THIS MARGIN
;TERMINATION OF BANK LOOP
4$: CALL INCBNK ;NEXT HIGHER BANK
BNE 2$ ;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
    
```

3193 016564

MTEST: SUBST <<SUBR SETUP MEMORY TEST>>

: *SUBTEST SUBR SETUP MEMORY TEST
: *****

3194 016564 013700 002112
3195 016570 006300
3196 016572 010037 177750
3197 016576
3198 016604
3199 016612 005037 002362
3200 016616 005737 002126
3201 016622 001410
3202
3203 016624 005737 002332
3204 016630 001002
3205 016632 004737 016664
3206 016636 004737 021222
3207 016642 000402
3208 016644 004737 021512
3209 016650 005037 002116
3210 016654
3211 016662 000207

MOV MARGIN,RO
ASL RO
MOV RO,MAINT ;SET MARGINS
SET HEADER ;INITIALIZE HEADER MESSAGE TYPEOUT
SET MUT ;INDICATE THERE IS A MEMORY UNDER TEST
CLR PASFLG
TST MKFLAG ;MK11?
BEQ 2\$;NO - SKIP
;IF CONTFLAG IS FALSE THEN CALL MKCONTROL
TST CONTFLAG
BNE 1\$
CALL MKCONTROL
1\$: CALL MKTEST ;YES - DO MK11 TESTS
BR 3\$
2\$: CALL MJTEST ;DO MJ11 TESTS
3\$: CLR MUT ;NOW - NO MEMORY UNDER TEST
SET HEADER ;ALLOW HEADERS NORMAL
RETURN

3214 016664
 3215
 3216
 3217
 3218
 3219
 3220
 3221
 3222
 3223
 3224
 3225
 3226
 3227 016664
 3228 016674
 3229 016700 112737 000200 002416
 3230 016706 112737 000001 002417
 3231 016714
 3232 016722
 3233 016732 004737 017300
 3234 016736
 3235 016744 004737 017610
 3236 016750
 3237 016750 005037 002424
 3238 016754
 3239 016762 004737 042422
 3240 016766
 3241 017002
 3242 017016 104511
 3243 017020 000241
 3244 017022
 3245 017034
 3246 017036
 3247 017044 004737 017374
 3248 017050
 3249 017052 104424
 3250 017054
 3251 017060 005037 002100
 3252 017064
 3253 017072 005037 002362
 3254 017076
 3255 017102 004737 020114
 3256 017106
 3257 017122 104423
 3258 017124
 3259 017132
 3260 017134
 3261 017134
 3262 017152
 3263 017152
 3264 017152
 3265 017166
 3266 017166
 3267 017174 113737 002420 113625

```

MKCONTROL:SUBTST      <<SUBR TEST MK11 CONTROLLER LOGIC DISPATCH>>
:*****
:*SUBTEST      SUBR      TEST MK11 CONTROLLER LOGIC DISPATCH
:*****
:THIS MODULE AND THE NEXT FEW PAGES SOLVE THE PROBLEM OF
:HOW TO RUN THE CSR TESTS ON EACH MK11 MEMORY
:
:IN ORDER TO TEST EACH CONTROLLER (2 PER BOX) I MUST DO
:  EACH EXISTING BOX
:    IF ECC IS NOT DISABLED
:    EACH SIDE OF EACH BOX
:    FIND THE ADDRESS OF THIS SIDE (DEPENDS ON INTERLEAVE)
:    FIND A LOCATION THAT HAS NO SINGLE BIT ERRORS
:    RUN ALL CSR TESTS
:  NEXT SIDE
:  NEXT BOX
:  IF SELONLY IS TRUE THEN $RETURN
PUSH      BANK
MOVB     #200,CSRINDEX      ;SET BIT FOR CONTROLLER
MOVB     #1,CSROUDEX       ;SET BIT FOR CONTROLLER
FOR CSRLOOP := #'0 TO #'7
  IFB CSRINDEX SET IN MKCSRS
  CALL    ISECCON
  FOR SIDE := #'0 TO #'1
    CALL    INTKRAP
  BEGIN CSRSTUFF
  CLR     SUCCESS
  FOR BANK := CSRLBANK DOWNTO CSRFBANK
    CALL    EXBANK
    IF ACFLAG IS TRUE AND RRFLAG IS FALSE
      MAP    BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      INVALIDATE          ;INVALIDATE BACKGROUND PATTERN ON "BANK"
    CLC
    IF SPECIAL IS TRUE THEN $CALL SIDEOK
  ON,NOERROR
  FOR TESTADD := CSRFIRST TO #LAST BY CSRINC
    CALL    SBTEST
  ON,NOERROR
  CACHOFF          ;TURN CACHE OFF
  FOR I := #0 TO #37
    CLR     NOPAR          ;INDICATE PARITY ACTION
    SET     HEADER
    CLR     PASFLG
    LET RO := I
    CALL    CSRCASE
  END ;OF FOR I
  CACHON          ;TURN CACHE ON
  SET     SUCCESS
  LEAVE CSRSTUFF
  END ;OF ON,NOERROR
  END ;OF FOR TESTADD
  END ;OF ON,NOERROR
  END ;OF IF
  END ;OF FOR BANK
  END CSRSTUFF
  IF SUCCESS IS FALSE
    MOVB    CSRLOOP,MSG34      ;SETUP CSR NUMBER
  
```

```
3268 017202 113737 002422 113642          MOVB   SIDE,MSGB34          ;SETUP SIDE (0 OR 1)
3269 017210                                TYPE   MSG034
3270 017214                                END ;OF IF SUCCESS
3271 017214                                END ;OF FOR SIDE
3272 017230                                END ;OF IFB CSRINDEX
3273 017230 000241                          CLC
3274 017232 106037 002416                    RORB  CSRINDEX           ;CHANGE TO NEXT CONTROLLER
3275 017236 106337 002417                    ASLB  CSROUDEX           ;CHANGE TO NEXT CONTROLLER
3276 017242                                END ;OF FOR CSRLOOP
3277 017256 104472                          ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
3278 017260                                SET   CONTFLAG
3279 017266                                POP   BANK
3280 017272 004737 042422                    CALL  EXBANK
3281 017276 000207                          RETURN
```

3284 017300
 3285
 3286 017300 005000
 3287 017302 113702 002417
 3288 017306 106002
 3289 017310 103403
 3290 017312 062700 000004
 3291 017316 000773
 3292 017320 010037 002256
 3293 017324 104503
 3294 017326 104426
 3295 017330
 3296 017340
 3297 017344 113737 002420 112134
 3298 017352
 3299 017356
 3300 017364
 3301 017366 005037 002474
 3302 017372
 3303 017372 000207

```

ISECCON:SUBTST <<IS ECC ON?>>
:*****
:*SUBTEST      IS ECC ON?
:*****
:SETUP CSR NUMBER
CLR          R0                ;START A ZERO
MOVB        CSROUDX,R2        ;GET CSR INDEX #2
RORB        R2                ;GET CSR BIT IN CARRY (MAYBE)
BCS         2$                ;I GOT IT - SKIP
ADD         #4,R0             ;BUMP CSR ADDRESS LSB'S BY 4
BR          1$                ;LOOP TILL I FIND THE CSR
2$:MOV       R0,CSRNO         ;UPDATE CSR NUMBER
CLR1CSR
READCSR
IF #BIT1 SET,IN CSR
TYPE        MSG078           ;CLEAR 1 SELECTED MK11 CSR
MOVB        CSRL0OP,MSG015
TYPE        MSG015
SET         NOECCFLAG
ELSE
CLR         NOECCFLAG
END :OF IF #BIT1
RETURN
  
```

;WARNING! ECC IS DISABLED ON BOX

3306 017374

```
SBTEST:SUBTST <<CHECK FOR SBE FREE LOCATIONS>>
:*****
:*SUBTEST      CHECK FOR SBE FREE LOCATIONS
:*****
```

3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331 017374 013701 002506
3332 017400
3333 017406 104424
3334 017410 104471
3335 017412
3336 017420 005711
3337 017422 001061
3338 017424 005761 000002
3339 017430 001056
3340
3341 017432 104503
3342 017434
3343 017442 005711
3344 017444 001050
3345 017446 005761 000002
3346 017452 001045
3347
3348 017454 104510
3349 017456
3350
3351 017466 104471
3352 017470 005111
3353 017472 005161 000002
3354 017476 023711 002726
3355 017502 001031
3356 017504 023761 002726 000002
3357 017512 001025
3358
3359 017514 104503

```
: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
MOV TESTADD,R1
SUPERVISOR ;ENTER SUPERVISOR MODE
CACHOFF ;TURN CACHE OFF
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
CLEAR (R1),2(R1)
TST (R1)
BNE 1$
TST 2(R1)
BNE 1$

CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
CLEAR (R1),2(R1)
TST (R1)
BNE 1$
TST 2(R1)
BNE 1$

TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
IF #BIT15!BIT4 SET.IN CSR THEN GOTO 1$

ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
COM (R1)
COM 2(R1)
CMP ONES,(R1)
BNE 1$
CMP ONES,2(R1)
BNE 1$

CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
```

```
3360 017516 005011 CLR (R1)
3361 017520 012761 100000 000002 MOV #BIT15,2(R1)
3362 017526 005711 TST (R1)
3363 017530 001016 BNE 1$
3364 017532 022761 100000 000002 CMP #BIT15,2(R1)
3365 017540 001012 BNE 1$
3366
3367 017542 104510 TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
3368 017544 IF #BIT15!BIT4 SET.IN CSR THEN GOTO 1$
3369
3370 017554 104417 KERNEL ;ENTER KERNEL MODE
3371 017556 104473 ECC1INIT ;INITIALIZE 1 SELECTED MK11 CSR
3372 017560 104423 CACHON ;TURN CACHE ON
3373 017562 $RETURN NOERROR
3374
3375 017566 104503 1$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
3376 017570 CLEAR (R1),2(R1)
3377 017576 104417 KERNEL ;ENTER KERNEL MODE
3378 017600 104473 ECC1INIT ;INITIALIZE 1 SELECTED MK11 CSR
3379 017602 104423 CACHON ;TURN CACHE ON
3380 017604 $RETURN ERROR
3381 .DSABL LSB
```

3384 017610

INTKRAP:SUBTST <<SETUP INTERLEAVE INFO>>

3385
 3386
 3387
 3388
 3389
 3390 017610
 3391 017612 004737 017736
 3392 017616 013737 002260 002410
 3393 017624 013737 002262 002412
 3394 017632 005037 002432
 3395 017636 104426
 3396 017640 013700 002146
 3397 017644 072027 177765
 3398 017650 042700 177760
 3399 017654
 3400 017664 020234
 3401 017666 020260
 3402 017670 020324
 3403 017672 020350
 3404 017674 020414
 3405 017676 020440
 3406 017700 020504
 3407 017702 020530
 3408 017704 020574
 3409 017706 020574
 3410 017710 020616
 3411 017712 020642
 3412 017714 020706
 3413 017716 020732
 3414 017720 020776
 3415 017722 021022
 3416 017724
 3417 017732
 3418 017734 000207

```

*****
: *SUBTEST      SETUP INTERLEAVE INFO
*****
: BASED ON THE STARTING ADDRESS, BOX CAPACITY, INTERLEAVE, AND SIDE
: SETUP VARIABLES TO DETERMINE
: (1)  THE STARTING ADDRESS OF THIS CONTROLLER
: (2)  THE NUMBER TO ADD TO EACH ADDRESS TO STAY IN THIS CONTROLLER
: (3)  THE LAST ADDRESS OF THIS CONTROLLER
PUSH   R0
CALL   GETSIZE
MOV    FIRSTB,CSRFBANK
MOV    LASTB,CSRLBANK
CLR    SPECIAL
READCSR                ;READ CSR
MOV    CSR+2,R0
ASH   #-11,R0
BIC   #^C17,R0
;R0 = (0..17) = BITS 14,13,12,11 OF CSR+2
;IF R0 = 0 CALL NEXT; IF R0 = 1 CALL NEXT+2, ETC.
CASE R0
INT..0
INT..1
INT..2
INT..3
INT..4
INT..5
INT..6
INT..7
INT..10
INT..11
INT..12
INT..13
INT..14
INT..15
INT..16
INT..17
END
POP    R0
RETURN
;OF CASE
    
```

3421 017736
 3422
 3423 017736
 3424 017740 052737 000010 002144
 3425 017746 042737 000006 002144
 3426 017754 104425
 3427 017756 104426
 3428 017760 042737 000010 002144
 3429 017766 113700 002145
 3430 017772 006300
 3431 017774 110037 002262
 3432 020000 013700 002146
 3433
 3434 020004
 3435 020016
 3436 020030 042700 177000
 3437 020034 006300
 3438 020036 010037 002260
 3439 020042 063737 002260 002262
 3440 020050 005337 002262
 3441 020054
 3442 020072
 3443 020110
 3444 020112 000207

```

GETSIZE:SUBTST <<SUBR GET SIZE FROM CSR>>
:*****
:*SUBTEST      SUBR      GET SIZE FROM CSR
:*****
:SETUP FIRSTB AND LASTB
PUSH  R0
BIS   #BIT3,CSR           ;GET SET TO READ CSR
BIC   #BIT2!BIT1,CSR
LOADCSR
READCSR
BIC   #BIT3,CSR           ;RESTORE THIS FROM 'SIZE' MODE
MOVB  CSR+1,R0           ;GET SIZE
ASL   R0                 ;R0 = # OF BANKS
MOVB  R0, LASTB
MOV   CSR+2,R0
:CORRECT LAST BANK FOR EXTERNAL INTERLEAVE
IF #BIT12 SET.IN R0 THEN LET LASTB := LASTB L.SHIFT 1
IF #BIT13 SET.IN R0 THEN LET LASTB := LASTB L.SHIFT 1
BIC   #177000,R0
ASL   R0
MOV   R0,FIRSTB
ADD   FIRSTB,LASTB
DEC   LASTB
IF FIRSTB HI #167 THEN LET FIRSTB := #170
IF LASTB HI #167 THEN LET LASTB := #167
POP   R0
RETURN
    
```


3490 020234

3491
3492
3493
3494
3495
3496
3497
3498
3499 020234
3500 020242
3501 020250
3502 020256 000207
3503
3504
3505 020260
3506 020266
3507 020276
3508 020304
3509 020306
3510 020314
3511 020314
3512 020322 000207
3513
3514
3515 020324
3516 020332
3517 020340
3518 020346 000207
3519
3520
3521 020350
3522 020356
3523 020366
3524 020374
3525 020376
3526 020404
3527 020404
3528 020412 000207
3529
3530
3531 020414
3532 020422
3533 020430
3534 020436 000207
3535
3536
3537 020440
3538 020446
3539 020456
3540 020464
3541 020466
3542 020474
3543 020474

```
      SUBTST <<DETERMINE ADDRESSING ACCORDING TO INTERLEAVE>>
:*****
:*SUBTEST      DETERMINE ADDRESSING ACCORDING TO INTERLEAVE
:*****
      ;CSRFBANK := THE FIRST BANK OF THIS CONTROLLER OF THIS CSR
      ;CSRIBANK := THE LAST BANK OF THIS CONTROLLER OF THIS CSR
      ;CSRFIRST := THE FIRST ADDRESS OF THIS CONTROLLER OF THIS CSR
      ;CSRINC  := THE NUMBER TO ADD TO THE ADDRESS PAIR
      ;        TO RETURN TO THIS CONTROLLER OF THIS CSR
      ;SPECIAL := A FLAG SET IF THIS BOX IS NOT INTERNALLY INTERLEAVED

      ;NO INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE
INT..0: LET CSRFIRST := #FIRST
      SET      SPECIAL
      LET CSRINC := #4
      RETURN

      ;INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE
INT..1: LET CSRFBANK := FIRSTB
      IFB SIDE EQ #'0
        LET CSRFIRST := #FIRST
      ELSE
        LET CSRFIRST := #FIRST+4
      END ;OF IFB
      LET CSRINC := #10
      RETURN

      ;NO INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH EVEN DOUBLE WORDS SELECTED?)
INT..2: LET CSRFIRST := #FIRST
      SET      SPECIAL
      LET CSRINC := #10
      RETURN

      ;INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH EVEN DOUBLE WORDS SELECTED?)
INT..3: LET CSRFBANK := FIRSTB
      IFB SIDE EQ #'0
        LET CSRFIRST := #FIRST
      ELSE
        LET CSRFIRST := #FIRST+10
      END ;OF IFB
      LET CSRINC := #20
      RETURN

      ;NO INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=0,A02=0)
INT..4: LET CSRFIRST := #FIRST
      SET      SPECIAL
      LET CSRINC := #20
      RETURN

      ;INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=0,A02=0)
INT..5: LET CSRFBANK := FIRSTB
      IFB SIDE EQ #'0
        LET CSRFIRST := #FIRST
      ELSE
        LET CSRFIRST := #FIRST+20
      END ;OF IFB
      LET CSRINC := #40
```

```
3544 020502 000207          RETURN
3545
3546          ;NO INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=1,A02=0)
3547 020504          INT..6: LET CSRFIRST := #FIRST+10
3548 020512          SET          SPECIAL
3549 020520          LET CSRINC := #20
3550 020526 000207          RETURN
3551
3552          ;INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=1,A02=0)
3553 020530          INT..7: LET CSRFBANK := FIRSTB
3554 020536          IFB SIDE EQ #'0
3555 020546          LET CSRFIRST := #FIRST+10
3556 020554          ELSE
3557 020556          LET CSRFIRST := #FIRST+30
3558 020564          END ;OF IFB
3559 020564          LET CSRINC := #40
3560 020572 000207          RETURN
3561
3562          ;IMPOSSIBLE CONDITION
3563          ;NO INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3564 020574          INT.10: ;FALL THRU TO INT.11
3565
3566          ;IMPOSSIBLE CONDITION
3567          ;INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3568 020574          INT.11: IF FSINFLAG IS FALSE
3569 020602          SET          FSINFLAG
3570 020610          TYPE          MSG097
3571 020614          END ;OF IF FSINFLAG
3572 020614          GOTO          INT..1
3573
3574          ;NO INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3575 020616          INT.12: LET CSRFIRST := #FIRST+4
3576 020624          SET          SPECIAL
3577 020632          LET CSRINC := #10
3578 020640 000207          RETURN
3579
3580          ;INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3581 020642          INT.13: LET CSRFBANK := FIRSTB
3582 020650          IFB SIDE EQ #'0
3583 020660          LET CSRFIRST := #FIRST+4
3584 020666          ELSE
3585 020670          LET CSRFIRST := #FIRST+14
3586 020676          END ;OF IFB
3587 020676          LET CSRINC := #20
3588 020704 000207          RETURN
3589
3590          ;NO INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=0,A02=1)
3591 020706          INT.14: LET CSRFIRST := #FIRST+4
3592 020714          SET          SPECIAL
3593 020722          LET CSRINC := #20
3594 020730 000207          RETURN
3595
3596          ;INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=0,A02=1)
3597 020732          INT.15: LET CSRFBANK := FIRSTB
3598 020740          IFB SIDE EQ #'0
3599 020750          LET CSRFIRST := #FIRST+4
3600 020756          ELSE
```

```
3601 020760          LET CSRFIRST := #FIRST+24
3602 020766          END :OF IFB
3603 020766          LET CSRINC := #40
3604 020774 000207   RETURN
3605
3606                :NO INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=1,A02=1)
3607 020776 INT.16: LET CSRFIRST := #FIRST+14
3608 021004          SET SPECIAL
3609 021012          LET CSRINC := #20
3610 021020 000207   RETURN
3611
3612                :INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=1,A02=1)
3613 021022 INT.17: LET CSRFBANK := FIRSTB
3614 021030          IFB SIDE EQ #'0
3615 021040          LET CSRFIRST := #FIRST+14
3616 021046          ELSE
3617 021050          LET CSRFIRST := #FIRST+34
3618 021056          END :OF IFB
3619 021056          LET CSRINC := #40
3620 021064 000207   RETURN
3621
```

3624 021066

SIDEOK: SUBST <<DETERMINE IF BANK BELONGS TO CORRECT SIDE>>
:*****
:*SUBTEST DETERMINE IF BANK BELONGS TO CORRECT SIDE
:*****
:RETURN ERROR IF NOT!

3625
3626
3627 021066 005037 002404
3628 021072 013702 002336
3629 021076 104475
3630 021100
3631 021106 104424
3632 021110 012722 000000
3633 021114 012712 000000
3634 021120 104503
3635 021122 005712
3636 021124 104426
3637 021126 013701 002146
3638
3639
3640 021132 104471
3641 021134 013712 002430
3642 021140 013742 002430
3643 021144 104503
3644 021146 104417
3645 021150 104423
3646
3647 021152
3648 021174
3649 021216

CLR CHECK
MOV CSRFIRST,R2
CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
SUPERVISOR ;ENTER SUPERVISOR MODE
CACHOFF ;TURN CACHE OFF
MOV #0,(R2)+
MOV #0,(R2)
CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
TST (R2)
READCSR
MOV CSR+2,R1

:ELIMINATE DBE
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
MOV ZEROS,(R2)
MOV ZEROS,-(R2)
CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
KERNEL ;ENTER KERNEL MODE
CACHON ;TURN CACHE ON

IF #BIT9 OFF.IN R1 AND SIDE EQ #'0 THEN \$RETURN NOERROR
IF #BIT9 SET.IN R1 AND SIDE EQ #'1 THEN \$RETURN NOERROR
\$RETURN ERROR

```

3653 021222
3654 021222
3655 021232
3656 021250 104470
3657 021252
3658 021254 104502
3659 021256
3660 021256 012737 000002 002100
3661 021264 012737 000002 002406
3662 021272 013700 002120
3663 021276 006300
3664 021300
3665 021320 104511
3666 021322
3667 021322 004770 021402
3668 021326
3669 021344 104476
3670 021346
3671 021350
3672 021356 013737 002144 002150
3673 021364 013737 002146 002152
3674 021372
3675 021372
3676 021372 104472
3677 021374 005037 002100
3678 021400 000207
3679
3680
3681
3682
3683
3684 021402
3685 021402 025652
3686 021404 023134
3687 021406 022414
3688 021410 021740
3689 021412 022000
3690 021414 022216
3691 021416 022300
3692 021420 023612
3693 021422 023156
3694 021424 023734
3695 021426 024240
3696 021430 024032
3697 021432 025214
3698 021434 025340
3699 021436 025530
3700 021440 025652
3701
3702 021442 025772
3703 021444 025772
3704 021446 025772
3705 021450 025772
3706 021452 025772
    
```

```

MKTEST: SUBST <<SUBR MK11 TEST DISPATCH>>
*****
:SUBTEST SUBR MK11 TEST DISPATCH
*****
IF MULTIONLY IS TRUE THEN $RETURN
IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
    ECCDIS ;DISABLE ERROR CORRECTION
ELSE
    CLRCSR ;CLEAR MK11 CSRS
END ;OF IF
MOV #2,NOPAR ;INDICATE PARITY ACTION
MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
MOV PATTERN,R0 ;GET PATTERN NUMBER
ASL R0 ;MAKE IT A WORD ADDRESS
IF MKPAT(R0) NE #MT0035 AND MKPAT(R0) NE #MT0999
    INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
END ;OF IF MKPAT(R0)
CALL @MKPAT(R0) ;INDEX OFF TABLE
IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
    WASSBE ;WAS THERE ANY SINGLE BIT ERRORS
    ON.ERROR
        SET SBFLAG
        MOV CSR,SBECRSR
        MOV CSR+2,SBECRSR+2
    END ;OF ON.ERROR
END ;OF IF #SWO
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
CLR NOPAR ;INDICATE PARITY ACTION
RETURN
    
```

```

:WARNING IF YOU CHANGE THIS TABLE ALSO
:CHANGE '$DDWO' - '$DDW5' (THE PATTERN BIT MAP)
MKPAT:
:PAT TIME DISCIPTION
:NOTE MT0035 MUST BE FIRST & LAST
MT0035 :<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
MT0033 :<1 SEC ;WRITE RECOVERY TEST
MT0034 :35 SEC ;BRANCH GOBBLE TEST
MT0035 :<1 SEC ;SOFT ERROR - BACKGROUND PATTERN TEST
:NOTE MT0035 MUST BE FIRST & LAST
MT0999 : 0 SEC ;NULL TEST
    
```

3707	021454	025772	MT0999	:	0	SEC	:	NULL	TEST
3708	021456	025772	MT0999	:	0	SEC	:	NULL	TEST
3709	021460	025772	MT0999	:	0	SEC	:	NULL	TEST
3710	021462	025772	MT0999	:	0	SEC	:	NULL	TEST
3711	021464	025772	MT0999	:	0	SEC	:	NULL	TEST
3712	021466	025772	MT0999	:	0	SEC	:	NULL	TEST
3713	021470	025772	MT0999	:	0	SEC	:	NULL	TEST
3714	021472	025772	MT0999	:	0	SEC	:	NULL	TEST
3715	021474	025772	MT0999	:	0	SEC	:	NULL	TEST
3716	021476	025772	MT0999	:	0	SEC	:	NULL	TEST
3717	021500	025772	MT0999	:	0	SEC	:	NULL	TEST
3718	021502	025772	MT0999	:	0	SEC	:	NULL	TEST
3719	021504	025772	MT0999	:	0	SEC	:	NULL	TEST
3720	021506	025772	MT0999	:	0	SEC	:	NULL	TEST
3721	021510	025772	MT0999	:	0	SEC	:	NULL	TEST

3778
 3779
 3780
 3781 021704

3782 021704 005037 002366
 3783 021710 012700 060000
 3784 021714 012701 040000
 3785 021720 004737 035204
 3786 021724
 3787 021732 004737 026042
 3788 021736 000207
 3789 021740

3790 021740 012737 000001 002366
 3791 021746 012700 060000
 3792 021752 012701 040000
 3793 021756 005002
 3794 021760 004737 035204
 3795 021764
 3796 021772 004737 026042
 3797 021776 000207
 3798 022000

3799 022000 012737 000002 002366
 3800 022006 012700 160000
 3801 022012 012701 040000
 3802 022016 012702 000001
 3803 022022 010103
 3804 022024 012704 060000
 3805 022030 012705 100001
 3806 022034
 3807 022042 004737 026042
 3808 022046 000207

```

.SBTTL PATTERNS
.SBTTL MEMORY TEST SETUP ROUTINES
MT0000: SUBTST <<MT0000 SETUP DATA PATTERN TEST>>
:*****
:*SUBTEST MT0000 SETUP DATA PATTERN TEST
:*****
      CLR REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV #FIRST,R0
      MOV #SIZE,R1
      CALL REGCOPY
      BMOV MTP000
      CALL SUPDO1 ;DO IT IN SUPERVISOR MODE
      RETURN
MT0001: SUBTST <<MT0001 SETUP ADDRESS TEST>>
:*****
:*SUBTEST MT0001 SETUP ADDRESS TEST
:*****
      MOV #1,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV #FIRST,R0
      MOV #SIZE,R1
      CLR R2
      CALL REGCOPY
      BMOV MTP001
      CALL SUPDO1 ;DO IT IN SUPERVISOR MODE
      RETURN
MT0002: SUBTST <<MT0002 SETUP COMPLEMENT ADDRESS TEST>>
:*****
:*SUBTEST MT0002 SETUP COMPLEMENT ADDRESS TEST
:*****
      MOV #2,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV #LAST+2,R0
      MOV #SIZE,R1
      MOV #1,R2
      MOV R1,R3
      MOV #FIRST,R4
      MOV #100001,R5
      BMOV MTP002
      CALL SUPDO1
      RETURN
    
```

3811 022050

MT0003: SUBTST <<MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST>>

:SUBTEST MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST

3812 022050 012737 000003 002366
3813 022056 005037 002406
3814 022062 004737 035214
3815 022066 012701 060000
3816 022072 005000
3817 022074 160001
3818 022076 012703 020000
3819 022102 072327 177770
3820 022106 012702 000004
3821 022112 012705 000100
3822 022116
3823 022130 104415
3824 022132 012737 177636 002360
3825 022140 004737 026220
3826 022144
3827 022152
3828 022164
3829 022176 104416
3830 022200 004737 026056
3831 022204 022737 000003 002734
3832
3833 022212 001323
3834 022214 000207
3835
3836 022216

MOV #3,REALPAT ; SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR PCBUMP ; TRAPS DO NOT ADD TO PC
1\$: CALL FLIPWARN ; SETUP WARNING CONSTANTS & R2
2\$: MOV #FIRST,R1 ; R1 <-- STARTING ADDRESS
CLR R0 ; INTERLEAVE ADDRESS CONSTANT
SUB R0,R1 ; CORRECT FOR ADD BEFORE USE MODE
MOV #SIZE/2,R3 ; INTERLEAVE SIZE (8K,4K,2K,OR 1K (DOUBLE WORDS))
ASH #-8,R3 ; R3 <-- R3 / 256.
MOV #4,R2 ; SMALL LOOP SIZE
MOV #64,R5 ; MEDIUM LOOP SIZE
BMOV MTPA03,UDPDR7,17.
SAVREG
MOV #FASTCITY-2,SUPDOADD
CALL SUPD03
BMOV MTPB03
BMOV MTPC03,KDPAR0,8.
BMOV MTPD03,SDPAR0,8.
RESREG
CALL SUPD02
CMP #3,FLIPL0C ; DONE WITH 4 PATTERNS
; [(0,177777);(177777,0);(401,177777);(177777,401)]?
; NO - LOOP
BNE 1\$
RETURN

MT0004: SUBTST <<MT0004 SETUP ROTATING ZEROS TEST>>

:SUBTEST MT0004 SETUP ROTATING ZEROS TEST

3837 022216 012737 000004 002366
3838 022224 012737 000004 002406
3839 022232 013702 002726
3840 022236 004737 035344
3841 022242 012700 060000
3842 022246 012701 040000
3843 022252
3844 022260
3845 022272 004737 026056
3846 022276 000207
3847 022300

MOV #4,REALPAT ; SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ; TRAPS ADD 4 TO PC
MOV ONES,R2
CALL BACKGND ; WRITE BACKGROUND OF ONES
MOV #FIRST,R0
MOV #SIZE,R1
BMOV MTPA04
BMOV MTPB04,KDPAR0,8.
CALL SUPD02
RETURN

MT0005: SUBTST <<MT0005 SETUP ROTATING ONES TEST>>

:SUBTEST MT0005 SETUP ROTATING ONES TEST

3848 022300 012737 000005 002366
3849 022306 012737 000004 002406
3850 022314 005002
3851 022316 004737 035344
3852 022322 012700 060000
3853 022326 012701 040000
3854 022332
3855 022340
3856 022352 004737 026056
3857 022356 000207

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
BMOV MTP005
BMOV MTPB04,KDPAR0,8.
CALL SUPD02
RETURN

3860 022360

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

: *SUBTEST MT0006 SETUP INITIAL DATA TEST
: *****

3861 022360 012737 000006 002366
3862 022366 012737 000004 002406
3863 022374 013701 002506
3864 022400 012737 026760 002360
3865 022406 004737 026220
3866 022412 000207
3867 022414

MOV #6,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV TESTADD,R1
MOV #MTP006,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

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

: *SUBTEST MT0007 SETUP ADDRESS BIT TEST
: *****

3868 022414 012737 000007 002366
3869 022422 005002
3870 022424 004737 035344
3871 022430 012701 060000
3872 022434 012702 000001
3873 022440 050201
3874 022442 012737 027150 002360
3875 022450 004737 026220
3876 022454 000207
3877 022456

MOV #7,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR R2
CALL BACKGND ;OF ZEROS
MOV #FIRST,R1
MOV #1,R2
BIS R2,R1
MOV #MTP007,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

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

: *SUBTEST MT0010 SETUP BYTE ADDRESSING TEST
: *****

3878 022456 012737 000010 002366
3879 022464 012737 000004 002406
3880 022472 013704 002506
3881 022476 012737 027250 002360
3882 022504 004737 026220
3883 022510 000207

MOV #10,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV TESTADD,R4
MOV #MTP010,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

3886 022512

3887 022512
 3888 022522
 3889 022536
 3890 022546
 3891 022546 012737 000011 002366
 3892 022554 012737 027356 002360
 3893 022562 004737 026220
 3894 022566 000207
 3895 022570

3896 022570
 3897 022600
 3898 022614
 3899 022624
 3900 022624 012737 000012 002366
 3901 022632 012737 027766 002360
 3902 022640 004737 026220
 3903 022644 000207
 3904 022646

3905 022646
 3906 022662
 3907 022672
 3908 022672 012737 000013 002366
 3909 022700 012737 030344 002360
 3910 022706 012737 000003 002100
 3911 022714 004737 026220
 3912 022720 000207
 3913 022722

3914 022722
 3915 022732
 3916 022746
 3917 022756
 3918 022756 012737 000014 002366
 3919 022764 012737 030716 002360
 3920 022772 004737 026220
 3921 022776 000207

```

MT0011: SUBTST <<MT0011      SETUP CREATE SINGLE BIT ERROR TEST>>
:*****
:*SUBTEST      MT0011  SETUP CREATE SINGLE BIT ERROR TEST
:*****
      IF NOECCFLAG IS TRUE THEN $RETURN
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
      IF $PASS NE #0 THEN $RETURN
      END ;OF IF ACTFLAG
      MOV      #11,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #MTP011,SUPDOADD
      CALL     SUPDO3          ;DO IT IN SUPERVISOR MODE
      RETURN

MT0012: SUBTST <<MT0012      SETUP WRITE BYTE CLEARS SBE TEST>>
:*****
:*SUBTEST      MT0012  SETUP WRITE BYTE CLEARS SBE TEST
:*****
      IF NOECCFLAG IS TRUE THEN $RETURN
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
      IF $PASS NE #0 THEN $RETURN
      END ;OF IF ACTFLAG
      MOV      #12,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #MTP012,SUPDOADD
      CALL     SUPDO3          ;DO IT IN SUPERVISOR MODE
      RETURN

MT0013: SUBTST <<MT0013      SETUP CREATE DOUBLE BIT ERROR TEST>>
:*****
:*SUBTEST      MT0013  SETUP CREATE DOUBLE BIT ERROR TEST
:*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
      IF $PASS NE #0 THEN $RETURN
      END ;OF IF ACTFLAG
      MOV      #13,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #MTP013,SUPDOADD
      MOV      #3,NOPAR        ;INDICATE PARITY ACRION
      CALL     SUPDO3          ;DO IT IN SUPERVISOR MODE
      RETURN

MT0014: SUBTST <<MT0014      SETUP WRITE INHIBIT DURING DATIP WITH DBE>>
:*****
:*SUBTEST      MT0014  SETUP WRITE INHIBIT DURING DATIP WITH DBE
:*****
      IF NOECCFLAG IS TRUE THEN $RETURN
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
      IF $PASS NE #0 THEN $RETURN
      END ;OF IF ACTFLAG
      MOV      #14,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #MTP014,SUPDOADD
      CALL     SUPDO3          ;DO IT IN SUPERVISOR MODE
      RETURN

```

3924 023000

MT0015: SUBTST <<MT0015 SETUP WRITE INHIBIT OF BYTE WITH DBE>>

:*****
:*SUBTEST MT0015 SETUP WRITE INHIBIT OF BYTE WITH DBE
:*****

3925 023000
3926 023010
3927 023024
3928 023034
3929 023034 012737 000015 002366
3930 023042 012737 031312 002360
3931 023050 004737 026220
3932 023054 000207
3933 023056

IF NOECCFLAG IS TRUE THEN \$RETURN
IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
IF \$PASS NE #0 THEN \$RETURN
END ;OF IF ACTFLAG
MOV #15,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #MTP015,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

MT0016: SUBTST <<MT0016 SETUP WRITE INHIBIT OF WORD WITH DBE>>

:*****
:*SUBTEST MT0016 SETUP WRITE INHIBIT OF WORD WITH DBE
:*****

3934 023056
3935 023066
3936 023102
3937 023112
3938 023112 012737 000016 002366
3939 023120 012737 031700 002360
3940 023126 004737 026220
3941 023132 000207
3942 023134

IF NOECCFLAG IS TRUE THEN \$RETURN
IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
IF \$PASS NE #0 THEN \$RETURN
END ;OF IF ACTFLAG
MOV #16,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #MTP016,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

MT0017: SUBTST <<MT0017 SETUP HOLDING 1'S & 0'S>>

:*****
:*SUBTEST MT0017 SETUP HOLDING 1'S & 0'S
:*****

3943 023134 012737 000017 002366
3944 023142 012737 032302 002360
3945 023150 004737 026220
3946 023154 000207

MOV #17,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #MTP017,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

3949 023156

MT0020: SUBTST <<MT0020 SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST>>

:SUBTEST MT0020 SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST

3950 023156
3951 023172
3952 023202
3953 023202 012737 000020 002366
3954 023210 005037 002100
3955 023214 012737 032572 002360
3956 023222 0137 W 002334
3957 023226
3958 023232 013737 023320 002256
3959 023240 106300
3960 023242
3961 023244 112737 000060 002422
3962 023252 004737 017610
3963 023256
3964 023264 006237 002414
3965 023270
3966
3967 023270
3968 023272 004737 023322
3969 023276
3970 023300
3971 023300
3972 023316 000207
3973 023320 000000
3974
3975 023322 013700 002336
3976 023326 010001
3977 023330 013702 002414
3978 023334 005003
3979 023336 012704 100000
3980 023342 012705 160000
3981 023346 104503
3982 023350
3983 023356 004737 026042
3984 023362
3985 023400
3986 023402
3987 023402
3988 023410 004737 026056
3989 023414
3990 023432 104477
3991 023434
3992 023436
3993 023436
3994 023444 004737 026056
3995 023450
3996 023466 104477
3997 023470
3998 023472
3999 023472 013701 002336
4000 023476
4001 023504 004737 026056
4002 023510

IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
IF \$PASS NE #0 THEN \$RETURN
END ;OF IF ACTFLAG
MOV #20,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR NOPAR ;INDICATE PARITY ACTION
MOV #MTPA20,SUPDOADD
MOV MKCSRS,R0
FOR MTV020 := #0 TO #34 BY #4
MOV MTV020,CSRNO
ASLB R0
ON.ERROR
MOVB #'0,SIDE
CALL INTKRAP
IF SPECIAL IS FALSE
ASR CSRINC
END ;OF IF SPECIAL
;NOW CSRFIRST & CSRINC ARE SETUP
PUSH R0
CALL MT020Z
POP R0
END ;OF ON.ERROR
END ;OF FOR CSRNO
RETURN
MTV020: 0 ;VARIABLE FOR PAT 20
MT020Z: MOV CSRFIRST,R0
MOV R0,R1
MOV CSRINC,R2
CLR R3
MOV #BIT15,R4
MOV #LAST+2,R5
CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
BMOV MTP20A
CALL SUPD01
IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
GOTO MT020Y
END ;OF IF #SW11
BMOV MTP20B
CALL SUPD02
IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
WAS1SBE ;WAS THERE ANY SINGLE BIT ERRORS ON 1 SELECTED CSR
ON.ERROR THEN GOTO MT020X
END ;OF IF #SW0
BMOV MTP20C
CALL SUPD02
IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
WAS1SBE ;WAS THERE ANY SINGLE BIT ERRORS ON 1 SELECTED CSR
ON.ERROR THEN GOTO MT020X
END ;OF IF #SW0
MT020Y: MOV CSRFIRST,R1
BMOV MTP20D
CALL SUPD02
IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE

```

4003 023526 104477          WAS1SBE          ;WAS THERE ANY SINGLE BIT ERRORS ON 1 SELECTED CSR
4004 023530          ON.ERROR THEN GOTO MT020X
4005 023532          END ;OF IF #SW0
4006 023532          BMOV      MTP20E
4007 023540 004737 026056  CALL      SUPD02
4008 023544          IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
4009 023562 104477          WAS1SBE          ;WAS THERE ANY SINGLE BIT ERRORS ON 1 SELECTED CSR
4010 023564          ON.ERROR THEN GOTO MT020X
4011 023566          END ;OF IF #SW0
4012 023566          BMOV      MTP20F
4013 023574 004737 026056  CALL      SUPD02
4014 023600 104503          CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
4015 023602 000207          RETURN
4016
4017 023604 004737 026234  MT020X: CALL      SUPD04
4018 023610 000207          RETURN

```

4021 023612

MT0021: SUBTST <<MT0021 SETUP MARCHING 0'S & 1'S TEST>>

: *SUBTEST MT0021 SETUP MARCHING 0'S & 1'S TEST

4022 023612
4023 023620 012737 000021 002366
4024 023626 013702 002762
4025 023632 004737 035344
4026 023636 010203
4027 023640 000303
4028 023642 012701 160000
4029 023646 010105
4030 023650 012704 060000
4031 023654
4032 023662 004737 026042
4033
4034 023666
4035 023674 004737 026056
4036
4037 023700 010401
4038 023702
4039 023710 004737 026056
4040
4041 023714
4042 023722 004737 026056
4043 023726 005037 002540
4044 023732 000207

SET NOSCOPE
MOV #21,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV BAKPAT,R2
CALL BACKGND
MOV R2,R3
SWAB R3
MOV #LAST+2,R1
MOV R1,R5
MOV #FIRST,R4
BMOV MTPA21
CALL SUPD01

BMOV MTPB21
CALL SUPD02

MOV R4,R1
BMOV MTPC21
CALL SUPD02

BMOV MTPD21
CALL SUPD02
CLR NOSCOPE
RETURN

4045
4046 023734

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

: *SUBTEST MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST

4047 023734 004737 026006
4048 023740
4049 023744 012737 000022 002366
4050 023752 012737 033404 002360
4051 023760 004737 026220
4052 023764 000207
4053

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
MOV #22,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #MTP022,SUPDOADD
CALL SUPD03 ;DO IT IN SUPERVISOR MODE
RETURN

4054 023766

MT0023: SUBTST <<MT0023 SHIFTING DIAGONAL TEST>>

: *SUBTEST MT0023 SHIFTING DIAGONAL TEST

4055 023766 004737 026006
4056 023772
4057 023776 012737 000023 002366
4058 024004 012737 033404 002360
4059 024012
4060 024020 004737 026220
4061 024024 005037 002006
4062 024030 000207

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
MOV #23,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #MTP022,SUPDOADD
SET DIAGFLAG ;IDENTIFY DIAGONAL TEST TO MTP022
CALL SUPD03 ;DO IT IN SUPERVISOR MODE
CLR DIAGFLAG
RETURN

4064 024032

MT0024: SUBTST <<MT0024 SETUP FAST GALLOPING PATTERN TEST>>

 :*SUBTEST MT0024 SETUP FAST GALLOPING PATTERN TEST

4065 024032 004737 026006
 4066 024036
 4067 024042
 4068 024050 012737 000024 002366
 4069 024056 013702 002762
 4070 024062 004737 035344
 4071 024066 010203
 4072 024070 010304
 4073 024072 000304
 4074 024074 012701 060000
 4075 024100 012705 157776
 4076 024104 104415
 4077 024106
 4078 024114
 4079 024126
 4080 024140 012737 172260 002360
 4081 024146 004737 026234
 4082
 4083
 4084 024152 104416
 4085 024154 000302
 4086 024156 000303
 4087 024160 004737 026234
 4088 024164 005037 002540
 4089 024170 000207
 4090 024172

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
 SAVREG
 BMOV MTPA24
 BMOV MTPB24,SDPAR0,8.
 BMOV MTPC24,KDPAR0,8.
 MOV #SDPAR0,SUPDOADD
 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

4091 024172
 4092 024206
 4093 024216
 4094 024216 012737 000025 002366
 4095 024224 012737 034152 002360
 4096 024232 004737 026220
 4097 024236 000207

IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
 IF \$PASS NE #0 THEN \$RETURN
 END ;OF IF ACTFLAG
 MOV #25,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 MOV #MTP025,SUPDOADD
 CALL SUPD03 ;DO IT IN SUPERVISOR MODE
 RETURN

4100 024240

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

: *SUBTEST MT0026 SETUP RANDOM DATA TEST
: *****

4101 024240 012737 000026 002366
4102 024246 005037 002406
4103 024252 013703 002716
4104 024256 013702 002714
4105 024262 010305
4106 024264 010204
4107 024266 012701 060000
4108 024272 012700 020000
4109 024276 104415
4110 024300
4111 024306
4112 024320
4113 024332 004737 026042
4114 024336 005037 034626
4115 024342
4116 024350 104416
4117 024352 004737 026042
4118 024356 010337 002716
4119 024362 010237 002714
4120 024366 000207

MOV #26,REALPAT
CLR PCBUMP ; TRAPS DO NOT ADD TO THE PC
MOV SEEDLO,R3 ; INITIALIZE RANDOM NUMBERS
MOV SEEDHI,R2
MOV R3,R5
MOV R2,R4
MOV #FIRST,R1
MOV #SIZE/2,R0
SAVREG
BMOV MTPA26 ; WRITE ROUTINE TO FAST MEMORY
BMOV MTPC26,KDPAR0,8. ; RANDOM SUBPROGRAM TO FAST MEMORY
BMOV MTPD26,SDPAR0,8. ; RANDOM SUBSUBPROGRAM TO FAST MEMORY
CALL SUPD01 ; WRITE RANDOM DATA
CLR RANODD ; FOR ERROR REPORTING
BMOV MTPB26 ; READ ROUTINE TO FAST MEMORY
RESREG
CALL SUPD01 ; READ RANDOM DATA
MOV R3,SEEDLO ; UPDATE FOR NEW RANDOM NUMBERS
MOV R2,SEEDHI
RETURN

4123 024370

MTO027: SUBTST <<MTO027 UNIQUE BANK TEST>>

 :*SUBTEST MTO027 UNIQUE BANK TEST

4124
 4125
 4126 024370 012737 000027 002366
 4127 024376
 4128 024406 104502
 4129 024410
 4130 024416 012737 000207 177644 WARN7:
 4131 024424 012737 177646 002360
 4132 024432
 4133 024440
 4134 024446
 4135 024452 004737 042422
 4136 024456
 4137 024472 104511
 4138 024474
 4139 024500 012700 060000
 4140 024504 010004
 4141 024506 012701 040000
 4142 024512 010103
 4143 024514
 4144 024524 004737 026042
 4145 024530
 4146 024530
 4147 024540 004737 026220
 4148 024544
 4149 024544
 4150 024544
 4151 024560
 4152 024574
 4153 024602 005037 002526
 4154 024606 000207
 4155 024610
 4156 024610
 4157 024616
 4158 024624 004737 042422
 4159 024630
 4160 024644
 4161 024650 005102
 4162 024652 012700 060000
 4163 024656 010004
 4164 024660 012701 040000
 4165 024664 010103
 4166 024666
 4167 024676 004737 026042
 4168 024702
 4169 024702
 4170 024712 004737 026220
 4171 024716
 4172 024716
 4173 024716
 4174 024732
 4175 024746 005037 002526
 4176 024752 000207

```

: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
CLEAR MARGIN,MAINT ;CLEAR MARGINS
CLRCSR ;CLEAR MK11 CSRS
BMOV MTP000
MOV #207,UIPAR2 ;PUT 'RETURN' INSTRUCTION AFTER WRITE ROUTINE
MOV #UIPAR3,SUPDOADD
SET NOFSMODE
FOR I := #1 TO #2
FOR BANK := #0 TO #167
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
CALL SUPD01
END ;OF IF
IF I EQ #2
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 := #167 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
CALL SUPD01
END ;OF IF
IF I EQ #2
CALL SUPD03
END ;OF IF
END ;OF IF
END ;OF FOR BANK
END ;OF FOR I
CLR NOFSMODE
RETURN
    
```

4179 024754
 4180 024754 005037 002362
 4181 024760 012737 000030 002366
 4182 024766
 4183 024776 012737 000001 002100
 4184 025004
 4185 025012 104470
 4186 025014
 4187 025030
 4188 025034 004737 042422
 4189 025040
 4190 025046
 4191 025062 012701 040000
 4192 025066 012700 060000
 4193 025072 004737 026042
 4194 025076
 4195 025076
 4196 025076
 4197 025112
 4198 025120
 4199 025126 104502
 4200 025130 004737 041130
 4201 025134
 4202 025136 104472
 4203 025140
 4204 025150 000207
 4205 025152
 4206 025152 013737 002400 002106
 4207 025160 004737 042422
 4208 025164 004737 024760
 4209 025170 104472
 4210 025172 004737 041604
 4211 025176 000207
 4212 025200
 4213 025200 104472
 4214 025202
 4215 025212 000207

```

MT0030: SUBTST <<MT0030      SETUP FLUSH OUT DBE'S TEST>>
:*****
:*SUBTEST      MT0030  SETUP FLUSH OUT DBE'S TEST
:*****
      CLR      PASFLG
MTA030: MOV      #30,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      CLEAR    MARGIN,MAINT      ;CLEAR MARGINS
      MOV      #1,NOPAR          ;INDICATE COUNT PARITY ERRORS
      BMOV     MTP030
      ECCDIS                    ;DISABLE ERROR CORRECTION
      SET      NOFSMODE,NOSCOPE
      FOR BANK := #0 TO #167
      CALL     EXBANK
      IF MKFLAG IS TRUE
      IF ACFLAG IS TRUE AND RRFLAG IS FALSE
      MOV      #SIZE,R1
      MOV      #FIRST,R0
      CALL     SUPD01
      END ;OF IF ACFLAG
      END ;OF IF MKFLAG
      END ;OF FOR
      IF PASFLG IS FALSE
      SET PASFLG
      CLRCSR                    ;CLEAR MK11 CSRS
      CALL     RELOCATE
      ON.ERROR
      ECCINIT                    ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
      CLEAR    NOFSMODE,NOSCOPE
      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
      RETURN
    
```

4218 025214

MTO031: SUBTST <<MTO031 SETUP SOB-A-LONG TEST>>

 :SUBTEST MTO031 SETUP SOB-A-LONG TEST

4219 025214 004737 026006

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
 ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN

4220 025220

SET NOSCOPE

4221 025224

MOV #31,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

4222 025232 012737 000031 002366

CLR NOPAR ;SETUP PARITY ACTION

4223 025240 005037 002100

MAP BANK ;MAP FIRST SO BLOCK MOVE WORKS

4224 025244

SUPERVISOR ;ENTER SUPERVISOR MODE

4225 025260

BMOV MTP031,FIRST,SOBLENGTH/2

4226 025266

KERNEL ;ENTER KERNEL MODE

4227 025300 104417

MOV SOBK,R2

4228 025302 013702 002704

MOV R2,R0

4229 025306 010200

MOV #100776,R1 ;COMPLEMENT OF INSTRUCTION "SOB R0,DOT"

4230 025310 012701 100776

MOV #FIRST+SOBLENGTH,R5

4231 025314 012705 060056

MOV #FIRST+2,SUPDOADD

4232 025320 012737 060002 002360

CALL SUPD04

4233 025326 004737 026234

CLR NOSCOPE

4234 025332 005037 002540

RETURN

4235 025336 000207

4238 025340

MT0033: SUBST <<MT0033 SETUP WRITE RECOVERY TEST>>

 :SUBTEST MT0033 SETUP WRITE RECOVERY TEST

4239 025340 004737 026006

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
 ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN

4240 025344

SET NOSCOPE

4241 025350

MOV #33,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

4242 025356 012737 000033 002366

CLR NOPAR ;SETUP PARITY ACTION

4243 025364 005037 002100

MAP BANK ;MAP FIRST SO THAT THE BLOCK MOVE WORKS

4244 025370

MOV #10247,R0 ;OPP CODE OF INSTRUCTION "MOV R2,-(PC)"

4245 025404 012700 010247

MOV #177667,R1 ;OPP CODE OF COMPLEMENT OF INSTRUCTION "JMP (R0)"

4246 025410 012701 177667

MOV #SIZE/2,R2 ;USED FOR 1/2 BANK LOOP

4247 025414 012702 020000

MOV #FIRST,R3

4248 025420 012703 060000

MOV #LAST+2,R4

4249 025424 012704 160000

4250

SUPERVISOR ;ENTER SUPERVISOR MODE

4251 025430

;MOVE TEST TO MEMORY UNDER TEST

4252

1\$: MOV R0,(R3)+

4253 025436 010023

MOV R1,-(R4)

4254 025440 010144

SOB R2,1\$

4255 025442 077203

4256

;MOVE LAST PART OF TEST TO FASTCITY

4257

BMOV MTP033

4258 025444

KERNEL ;ENTER KERNEL MODE

4259 025452 104417

4260

MOV #5141,R2 ;OPP CODE OF INSTRUCTION "COM -(R1)"

4261 025454 012702 005141

MOV #2\$,R0 ;ADDRESS TO RETURN TO IN R0

4262 025460 012700 025526

MOV #LAST+2,R1 ;TOP OF BANK

4263 025464 012701 160000

MOV #FIRST,SUPDOADD

4264 025470 012737 060000 002360

CALL SUPD04

4265 025476 004737 026234

MOV #SIZE/2,R3

4266 025502 012703 020000

MOV #110,R5

4267 025506 012705 000110

MOV #FIRST,R4

4268 025512 012704 060000

CALL SUPD02

4269 025516 004737 026056

CLR NOSCOPE

4270 025522 005037 002540

2\$: RETURN ;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0033

4271 025526 000207

4272

4273

;ALSO A RETURN FROM THE "CALL SUPD04" ABOVE

4276 025530
 4277 025530 004737 026006
 4278 025534
 4279 025540
 4280 025546 012737 000034 002366
 4281 025554 005037 002100
 4282 025560
 4283
 4284 025574
 4285 025602
 4286 025614 104417
 4287
 4288 025616 012705 060076
 4289 025622 012737 060004 002360
 4290 025630 012701 060002
 4291 025634 012702 060003
 4292
 4293 025640 004737 026234
 4294 025644 005037 002540
 4295 025650 000207
 4296
 4297 025652

```

MT0034: SUBTST <<MT0034      SETUP BRANCH GOBBLE TEST>>
:*****
:*SUBTEST      MT0034  SETUP BRANCH GOBBLE TEST
:*****
CALL      KAMITEST          ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
SET       NOSCOPE
MOV       #34,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR       NOPAR            ;SETUP PARITY ACTION
MAP       BANK             ;MAP FIRST SO THAT BLOCK MOVE WORKS

SUPERVISOR                ;ENTER SUPERVISOR MODE
BMOV     MTP034,FIRST,GBLENGTH/2
KERNEL                  ;ENTER KERNEL MODE

MOV       #FIRST+GBLENGTH,R5
MOV       #FIRST+4,SUPDOADD
MOV       #FIRST+2,R1
MOV       #FIRST+3,R2

CALL     SUPD04
CLR     NOSCOPE
RETURN
  
```

4298 025652 012737 000035 002366
 4299 025660 012700 060000
 4300 025664 012701 040000
 4301 025670 013702 002744
 4302 025674 010103
 4303 025676 013705 002106
 4304 025702 070527 000006
 4305 025706 010004
 4306 025710
 4307 025716 012737 000207 177644
 4308 025724 012737 177646 002360
 4309 025732
 4310
 4311 025750 004737 026220
 4312 025754
 4313
 4314 025756 004737 026042
 4315 025762 052765 020000 003020
 4316 025770
 4317 025770 000207

```

MT0035: SUBTST <<MT0035      SOFT ERROR - BACKGROUND PATTERN TEST>>
:*****
:*SUBTEST      MT0035  SOFT ERROR - BACKGROUND PATTERN TEST
:*****
MOV       #35,REALPAT
MOV       #FIRST,R0
MOV       #SIZE,R1
MOV       SOFTPAT,R2
MOV       R1,R3
MOV       BANK,R5
MUL      #6,R5
MOV       R0,R4
BMOV     MTP000
MOV       #207,UIPAR2      ;PUT RETURN INSTRUCTION AFTER WRITE ROUTINE
MOV       #UIPAR3,SUPDOADD
IF #BIT13 SET.IN CONFIG+2(R5) AND WRITEONLY IS FALSE
  ;BACKGROUND PATTERN IS VALID
  CALL    SUPD03          ;READ IT
ELSE
  ;BACKGROUND PATTERN HAS BEEN INVALIDATED
  CALL    SUPD01          ;WRITE IT
  BIS    #BIT13,CONFIG+2(R5) ;VALIDATE IT
END ;OF IF #BIT13
RETURN
  
```

4320 025772
4321 025772 005037 002366
4322 025776
4323 026004 000207
4324
4325 026006

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

4326 026006
4327 026030
4328 026034
4329 026036
4330 026042

```
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
```

4334 026042

SUPD01: SUBTST <<SUBR EXECUTE PATTERN IN SUPERVISOR>>

:*****
:SUBTEST SUBR EXECUTE PATTERN IN SUPERVISOR
:*****

4335 026042

MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK

4336 026056 004737 066514

SUPD02: CALL GETDIS

4337 026062

PUSH \$LPERR,\$LPADR

4338 026072 010037 002266

MOV R0,SUPDR0

4339 026076 012700 002270

MOV #SUPDR1,R0

4340 026102 010120

MOV R1,(R0)+

4341 026104 010220

MOV R2,(R0)+

4342 026106 010320

MOV R3,(R0)+

4343 026110 010420

MOV R4,(R0)+

4344 026112 010520

MOV R5,(R0)+

4345 026114 010620

MOV SP,(R0)+

4346 026116 013700 002266

MOV SUPDR0,R0

4347 026122 012737 026136 002746

MOV #TAG4\$,\$LPADR

4348 026130 013737 002746 002750

MOV \$LPADR,\$LPERR

TAG4\$:

MOV #SUPDR6+2,R0

4349 026136 012700 002304

MOV -(R0),SP

4350 026142 014006

MOV -(R0),R5

4351 026144 014005

MOV -(R0),R4

4352 026146 014004

MOV -(R0),R3

4353 026150 014003

MOV -(R0),R2

4354 026152 014002

MOV -(R0),R1

4355 026154 014001

MOV -(R0),R0

4356 026156 014000

SUPERVISOR

;ENTER SUPERVISOR MODE

4357 026160

MOV #SUPSTK,\$SP

4358 026166 012706 000740

CACHOFF

;TURN CACHE OFF

4359 026172 104424

CALL FASTCITY

;CALL TO THE USER INSTRUCTION PAR'S

4360 026174 004737 177640

CACHON

;TURN CACHE ON

4361 026200 104423

KERNEL

;ENTER KERNEL MODE

4362 026202 104417

SCOPE

4363 026204 000004

POP

\$LPADR,\$LPERR

4364 026206

RETURN

4365 026216 000207

4369 026220
 4370 026234 004737 066514
 4371 026240
 4372 026250 010037 002266
 4373 026254 012700 002270
 4374 026260 010120
 4375 026262 010220
 4376 026264 010320
 4377 026266 010420
 4378 026270 010520
 4379 026272 010620
 4380 026274 013700 002266
 4381 026300 012737 026314 002746
 4382 026306 013737 002746 002750
 4383 026314 012700 002304
 4384 026320 014006
 4385 026322 014005
 4386 026324 014004
 4387 026326 014003
 4388 026330 014002
 4389 026332 014001
 4390 026334 014000
 4391 026336
 4392 026344 012706 000740
 4393 026350 104424
 4394 026352 004777 154002
 4395 026356 104423
 4396 026360 104417
 4397 026362 000004
 4398 026364
 4399 026374 000207

SUPDO3: MAP BANK
 SUPDC4: 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 #TBG4\$,\$LPADR
 MOV \$LPADR,\$LPERR
 TBG4\$: 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
 MOV #SUPSTK,SSP
 CACHOFF
 CALL @SUPDOADD
 CACHON
 KERNEL
 SCOPE
 POP \$LPADR,\$LPERR
 RETURN

;MAP SUPERVISOR SPACE (TEST AREA) TO BANK

;ENTER SUPERVISOR MODE
 ;TURN CACHE OFF
 ;TURN CACHE ON
 ;ENTER KERNEL MODE

4403
 4404
 4405
 4406
 4407
 4408
 4409
 4410
 4411
 4412
 4413 026376

 4414 026376 010220
 4415 026400 077102
 4416 026402 000240
 4417 026404 012401
 4418 026406 020102
 4419 026410 001402
 4420 026412 104430
 4421 026414 000240
 4422 026416 077306
 4423 026420 000207
 4424 026422

 4425 026422 010220
 4426 026424 062702 000002
 4427 026430 077104
 4428 026432 000240
 4429 026434 012400
 4430 026436 020005
 4431 026440 001401
 4432 026442 104427
 4433 026444 062705 000002
 4434 026450 077307
 4435 026452 000207
 4436 026454

 4437 026454 010540
 4438 026456 062705 000002
 4439 026462 077104
 4440 026464 000240
 4441 026466 162702 000002
 4442 026472 012401
 4443 026474 020102
 4444 026476 001401
 4445 026500 104430
 4446 026502 077307
 4447 026504 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
    
```

4451 026506

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

4452
 4453
 4454
 4455
 4456
 4457
 4458
 4459 026506 060001
 4460 026510 010421
 4461 026512 010421
 4462 026514 077204
 4463 026516 005104
 4464 026520 052704
 4465 026522 000401
 4466 026524 012702 000004
 4467 026530 077512
 4468 026532 005104
 4469 026534 052704
 4470 026536 000401
 4471 026540 012705 000100
 4472 026544 077320
 4473 026546 000207
 4474
 4475
 4476 026550

```

:R0 = INTERLEAVE FACTOR
:R1 = ADDRESS
:R2 = SMALL LOOP CONSTANT
:R3 = NUM OF ADD TO TEST (LARGE LOOP)
:R4 = GOOD DATA
:R5 = MEDIUM LOOP CONSTANT
.ENABL  LSB
1$:  ADD    R0,R1          ;V177636 ADD INTERLEAVE FACTOR
     MOV    R4,(R1)+      ;V177640
     MOV    R4,(R1)+      ;V177642
     SOB    R2,1$         ;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)
 :*****

4477
 4478 026550 000137 172360
 4479 026554 077203
 4480 026556 005104
 4481 026560 052704
 4482 026562 000401
 4483 026564 012702 000004
 4484 026570 077511
 4485 026572 005104
 4486 026574 052704
 4487 026576 000401
 4488 026600 012705 000100
 4489 026604 077317
 4490 026606 000207
 4491

```

.ENABL  LSB
1$:  JMP    KDPAR0         ;V177640 GO TO V172360
     SOB    R2,1$         ;V177644
     COM    R4            ;V177646
     BIS    (PC)+,R4      ;V177650
WARN4: 401                ;V177652 WARNING LOCATION IS MODIFIED BEFORE 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
  
```

4494 026610

MTPC03: SUBTST <<MTPC03 TEST DATA SUBPROGRAM>>

:*SUBTEST MTPC03 TEST DATA SUBPROGRAM

4495 026610 060001
4496 026612 020421
4497 026614 001401
4498 026616 104431
4499 026620 005141
4500 026622 005111
4501 026624 000137 172260
4502
4503 026630

ADD R0,R1 ;V172360
CMP R4,(R1)+ ;V172362
BEQ 1\$;V172364
PERR03 ;V172366
1\$: COM -(R1) ;V172370
COM (R1) ;V172372
JMP SDPAR0 ;V174372 GO TO V172260

MTPD03: SUBTST <<MTPD03 TEST DATA SUBSUBPROGRAM>>

:*SUBTEST MTPD03 TEST DATA SUBSUBPROGRAM

4504 026630 020421
4505 026632 001401
4506 026634 104431
4507 026636 005127
4508 026640 000000
4509 026642 001033
4510 026644 000137 177644

CMP R4,(R1)+ ;V172260
BEQ 1\$;V172262
PERR03 ;V172264
1\$: COM (PC)+ ;V172266
O ;V172270
BNE .+70 ;V172272 GO TO V172362
JMP UIPAR2 ;V172274 GO TO V177644

4513 026650
4514 026650 012705 000010
4515 026654 010504
4516 026656 000241
4517 026660 000137 172360
4518 026664 016004 177776
4519 026670 103402
4520 026672 020204
4521 026674 001401
4522 026676 104432
4523 026700 077115
4524 026702 000207
4525
4526 026704

```

MTPA04: SUBTST <<MTPA04 ROTATING ZEROS TEST>>
:*****
:*SUBTEST MTPA04 ROTATING ZEROS TEST
:*****
1$: MOV #8.,R5 ;V177640
MOV R5,R4 ;V177644
CLC ;V177646
JMP KDPAR0 ;V177650
MOV -2(R0),R4 ;V177654
BCS 2$ ;V177660
CMP R2,R4 ;V177662
BEQ 3$ ;V177664
2$: PERRO4 ;V177666
3$: SOB R1,1$ ;V177670
RETURN ;V177672
    
```

4527 026704 106110
4528 026706 077502
4529 026710 106120
4530 026712 106110
4531 026714 077402
4532 026716 106120
4533 026720 000137 177654
4534
4535 026724

```

MTPB04: SUBTST <<MTPB04 SUBR ROTATING BIT>>
:*****
:*SUBTEST MTPB04 SUBR ROTATING BIT
:*****
1$: ROLB (R0) ;V172360
SOB R5,1$ ;V172362
ROLB (R0)+ ;V172364
2$: ROLB (R0) ;V172366
SOB R4,2$ ;V172370
ROLB (R0)+ ;V172372
JMP UIPAR6 ;V172374
    
```

4536 026724 012705 000010
4537 026730 010504
4538 026732 000261
4539 026734 000137 172360
4540 026740 016004 177776
4541 026744 103002
4542 026746 020204
4543 026750 001401
4544 026752 104432
4545 026754 077115
4546 026756 000207

```

MTP005: SUBTST <<MTP005 ROTATION ONES TEST>>
:*****
:*SUBTEST MTP005 ROTATION ONES TEST
:*****
1$: MOV #8.,R5 ;V177640
MOV R5,R4 ;V177644
SEC ;V177646
JMP KDPAR0 ;V177650
MOV -2(R0),R4 ;V177654
BCC 2$ ;V177660
CMP R2,R4 ;V177662
BEQ 3$ ;V177664
2$: PERRO4 ;V177666
3$: SOB R1,1$ ;V177670
RETURN ;V177672
    
```

IF THIS HAPPENS THE GOOD & BAD MATCH

4550 026760

MTP006: SUBST <<MTP006 INITIAL DATA TEST>>

 :*SUBTEST MTP006 INITIAL DATA TEST
 :*****

```

4551 : THIS TEST CHECKS THE DI/DO LINES BY
4552 : SHIFTING A 1 THROUGH THE WORD.
4553 026760 012737 000001 002340 MOV #1,DATBUF ;SET THE FIRST TEST BIT
4554 026766 005037 002342 CLR DATBUF+2 ;CLEAR 2ND WORD
4555 026772 013711 002340 1$: MOV DATBUF,(R1) ;WRITE TEST WORD 1
4556 026776 013761 002347 000002 MOV DATBUF+2,2(R1) ;AND TEST WORD 2
4557 027004 011102 MOV (R1),R2
4558 027006 023702 002340 CMP DATBUF,R2 ;NOW READ THEM
4559 027012 001401 BEQ 2$ ;BR IF FIRST 16 OK
4560 027014 104433 PERR07 ;ERROR TRAP
4561
4562 027016 016102 000002 2$: MOV 2(R1),R2 ;NOW READ SECOND WORD
4563 027022 023702 002342 CMP DATBUF+2,R2 ;BR IF OK
4564 027026 001401 BEQ 3$ ;ERROR TRAP
4565 027030 104434 PERR10
4566
4567 027032 005737 002342 3$: TST DATBUF+2 ;HAS LAST BIT BEEN TESTED ?
4568 027036 100405 BMI 4$ ;MINUS MEANS BIT 31
4569 027040 DLEFT DATBUF ;NO, SHIFT TEST BIT LEFT
4570 027050 000750 BR 1$ ;GO WRITE NEW TEST DATA
4571 ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
4572 027052 012737 177776 002340 4$: MOV #177776,DATBUF ;PUT A 0 IN BIT 0
4573 027060 012737 177777 002342 MOV #-1,DATBUF+2 ;AND 1'S IN ALL OTHERS
4574 027066 013711 002340 5$: MOV DATBUF,(R1) ;WRITE THE DATA
4575 027072 013761 002342 000002 MOV DATBUF+2,2(R1) ;2 WORDS WORTH
4576 027100 011102 MOV (R1),R2
4577 027102 023702 002340 CMP DATBUF,R2 ;NOW READ FIRST WORD
4578 027106 001401 BEQ 6$ ;BR IF OK
4579 027110 104433 PERR07
4580
4581 027112 016102 000002 6$: MOV 2(R1),R2 ;NOW, READ SECOND WORD
4582 027116 023702 002342 CMP DATBUF+2,R2 ;BR IF OK
4583 027122 001401 BEQ 7$
4584 027124 104434 PERR10
4585
4586 027126 005737 002342 7$: TST DATBUF+2 ;TESTED BIT 31 YET?
4587 027132 100005 BPL 8$ ;BR IF YES, WE'RE DONE
4588 027134 DLEFT DATBUF
4589 027144 000750 BR 5$ ;KEEP GOING
4590 027146 000207 8$: RETURN
  
```


4634 027250

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

```

4635
4636
4637 027250 010402
4638 027252 010403
4639 027254 062702 000004
4640 027260 012713 177777
4641 027264 012763 177777 000002
4642 027272 105013
4643 027274 010401
4644 027276 020201
4645 027300 001420
4646 027302 020301
4647 027304 001007
4648 027306 111100
4649
4650 027310 022700 000000
4651 027314 001401
4652 027316 104435
4653
4654 027320 005201
4655 027322 000765
4656 027324 111100
4657 027326 122700 177777
4658 027332 001401
4659 027334 104436
4660
4661 027336 005201
4662 027340 000756
4663 027342 112713 177777
4664 027346 005203
4665 027350 020302
4666 027352 001347
4667 027354 000207
  
```

```

MOV R4,R2 ;R4 HAS LOWEST ADDRESS
MOV R4,R3 ;PUT IT IN R3 ALSO
ADD #4,R2 ;POINT R2 TO LAST BYTE +1
MOV #-1,(R3) ;WRITE ALL ONES IN
MOV #-1,2(R3) ;THE 4 TEST BYTES
1$: CLR B (R3) ;CLEAR A BYTE
MOV R4,R1 ;INITIALIZE R1 FOR EACH PASS
2$: CMP R2,R1 ;IF EQUAL, JUST READ LAST BYTE
BEQ 6$ ;BR IF EQUAL
CMP R3,R1 ;IS THIS THE BYTE OF ZEROS
BNE 4$ ;BR IF NOT
MOV B (R1),R0
;WARNING IF YOU OPTIMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
CMP #0,R0 ;IT IS, COMPARE FOR ZEROS
BEQ 3$
PERR11
3$: INC R1 ;NEXT BYTE
BR 2$ ;RETURN
4$: MOV B (R1),R0
CMPB #-1,R0 ;ITS NOT THE BYTE OF 0'S, READ 1'S
BEQ 5$
PERR12
5$: INC R1 ;MOVE TO NEXT BYTE
BR 2$
6$: MOV B #-1,(R3) ;RESTORE 1'S TO BYTE JUST TESTED
INC R3 ;INC TO NEXT BYTE
CMP R3,R2 ;WAS THAT JUST THE LAST ONE?
BNE 1$ ;BR IF NO
RETURN
  
```

4671 027356

```

MTP011: SUBST <<MPT011 SINGLE BIT ERROR TEST>>
*****
: *SUBTEST MPT011 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 FORM 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 MK11 CSR
:BIG LOOP
MOV #1,DATBUF ;INITIAL DATA
CLR DATBUF+2 ;32 BITS WORTH
:MEDIUM LOOP
MOV #1,SBEMSK ;INITIAL ERROR MASK
CLR SBEMSK+2 ;32 BITS WORTH
:LITTLE LOOP
MOV DATBUF,TSTDAT ;
MOV DATBUF+2,TSTDAT+2;TO SAVE ORIG DATA
TSTB PASFLG ;COMP DATA ON SECOND PASSONLY
BEQ 4$ ;BR IF FIRST PASS
COM TSTDAT ;SECOND PASS, COMP BOTH WORDS
COM TSTDAT+2
4$: MOV TSTDAT,R2
MOV TSTDAT+2,R3
MOV #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
CALL CHKGEN ;GEN CHECKBITS ON TSTDAT
*****
: ** CREATE A SINGLE BIT ERROR **
*****
5$: MOV SBEMSK,R1
XOR R1,TSTDAT
MOV SBEMSK+2,R1
XOR R1,TSTDAT+2
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 AND
;CHECK BITS. WE NOW HAVE CHECKBITS
;GENERATED ON DATBUF AND DATA WITH
;ONE BIT IN ERROR (AS PER SBEMSK).
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
MOV -(R1),R0
CMP R0,TSTDAT ;READ THE LOW WORD (UNCORRECTED)
BEQ 6$ ;BR IF OK

```

4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688 027356 104503
4689
4690 027360 012737 000001 002340 1\$:
4691 027366 005037 002342
4692
4693 027372 012737 000001 002350 2\$:
4694 027400 005037 002352
4695
4696 027404 013737 002340 002344 3\$:
4697 027412 013737 002342 002346
4698 027420 105737 002362
4699 027424 001404
4700 027426 005137 002344
4701 027432 005137 002346
4702 027436 013702 002344 4\$:
4703 027442 013703 002346
4704 027446 012737 002344 002402
4705 027454 004737 040540
4706
4707
4708
4709 027460 013701 002350
4710 027464 074137 002344
4711 027470 013701 002352
4712 027474 074137 002346
4713 027500 013701 002506 5\$:
4714 027504 104471
4715 027506 013721 002344
4716 027512 104475
4717 027514 013711 002346
4718
4719
4720
4721 027520 104471
4722 027522 014100
4723 027524 020037 002344
4724 027530 001401

```

4725 027532 104455          PERR31
4726
4727 027534 062701 000002    6$:  ADD    #2,R1          ;POINT R1 TO SECOND 16 BITS
4728 027540 011100          MOV    (R1),R0
4729 027542 020037 002346    CMP    R0,TSTDAT+2      ;READ THE HIGH WORD (UNCORRECTED)
4730 027546 001401          BEQ    7$                ;BR IF OK
4731 027550 104455          PERR31
4732
4733 027552 104503          7$:  CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
4734 027554 014100          MOV    -(R1),R0
4735 027556 020002          CMP    R0,R2            ;SEE IF ITS BEEN CORRECTED
4736 027560 001401          BEQ    8$                ;IT SHOULD HAVE BEEN
4737 027562 104456          PERR32
4738
4739 027564 104510          8$:  TSTREAD          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4740 027566 103411          BCS    9$                ;BR IF IT IS SET
4741 027570          SET    HEADER           ;ENABLE PRINTING OF ERROR HEADER INFO
4742 027576 010137 002040    MOV    R1,ADDRESS
4743 027602 104460          PERR34
4744 027604          SET    HEADER           ;ENABLE PRINTING OF ERROR HEADER INFO
4745
4746 027612 104503          9$:  CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
4747 027614 062701 000002    ADD    #2,R1            ;POINT TO HIGH WORD
4748 027620 011100          MOV    (R1),R0
4749 027622 020003          CMP    R0,R3            ;SEE IF ITS BEEN CORRECTED
4750 027624 001401          BEQ    10$              ;BR IF OK
4751 027626 104456          PERR32
4752
4753 027630 104510          10$: TSTREAD          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4754 027632 103411          BCS    11$              ;BR IF YES
4755 027634          SET    HEADER           ;ENABLE PRINTING OF ERROR HEADER INFO
4756 027642 010137 002040    MOV    R1,ADDRESS
4757 027646 104460          PERR34
4758 027650          SET    HEADER           ;ENABLE PRINTING OF ERROR HEADER INFO
4759 027656 005737 002352    11$: TST    SBEMSK+2      ;TEST FOR LAST MASK BIT
4760 027662 100405          BMI    12$              ;MINUS MEANS BIT 31
4761 027664          DLEFT SBEMSK
4762 027674 000643          BR    3$
4763 027676          12$: IF #SW11 SET.IN @SWR THEN GOTO 13$
4764 027706          IF QVFLAG IS TRUE THEN GOTO 13$
4765 027714 005737 002342    TST    DATBUF+2        ;LAST DATA BIT ?
4766 027720 100405          BMI    13$              ;WHICH IS BIT 31
4767 027722          DLEFT DATBUF
4768 027732 000617          BR    2$
4769 027734 105737 002362    13$: TSTB    PASFLG ;FIRST OR SECOND PASS ?
4770 027740 001003          BNE    14$              ;NON ZERO MEANS WE'RE DONE
4771 027742 105237 002362    INCB   PASFLG ;NOT DONE, GO DO SECOND PASS
4772 027746 000604          BR    1$
4773          ;CLEAR OUT ANY DBE'S OR SBE'S
4774 027750 104471          14$: ECC1DIS          ;DISABLE ECC ON 1 SELECTED CSR
4775 027752 013701 002506    MOV    TESTADD,R1
4776 027756          CLEAR (R1)+,(R1)
4777 027762 104503          CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
4778 027764 000207          RETURN

```

4782 027766

MTP012: SUBTST <<MTP012 WRITE BYTE CLEARS SBE TEST>>
:*****
:*SUBTEST MTP012 WRITE BYTE CLEARS SBE TEST
:*****

```

4783                                     :SINGLE BIT ERROR TEST TO INSURE THAT A WRITE
4784                                     :BYTE CLEARS SINGLE BIT ERRORS.
4785 027766 104503                       CLR1CSR                               :CLEAR 1 SELECTED MK11 CSR
4786 027770 012737 000001 002340         MOV #1,DATBUF                        :INITIAL DATA
4787 027776 005037 002342                 CLR DATBUF+2                        :32 BITS WORTH
4788 030002 012737 000001 002350 1$:     MOV #1,SBEMSK                        :INITIAL ERROR MASK
4789 030010 005037 002352                 CLR SBEMSK+2                        :32 BITS WORTH
4790 030014 013737 002340 002344 2$:     MOV DATBUF,TSTDAT                   :SAVE ORIGINAL DATA
4791 030022 013737 002342 002346         MOV DATBUF+2,TSTDAT+2               :BOTH WORDS
4792 030030 012737 002344 002402         MOV #TSTDAT,SOURCE                 :NEED ADDRESS FOR CHKGEN
4793 030036 004737 040540                 CALL CHKGEN                         :GENERATE CHECK BITS
4794 030042 013701 002350                 MOV SBEMSK,R1
4795 030046 074137 002344                 XOR R1,TSTDAT
4796 030052 013701 002352                 MOV SBEMSK+2,R1
4797 030056 074137 002346                 XOR R1,TSTDAT+2
4798 030062 013704 002506 3$:           MOV TESTADD,R4                       :FIRST TEST ADDRESS
4799 030066 010401                         MOV R4,R1                            :PUT IT IN R1 ALSO
4800 030070 104471                         ECC1DIS                               :DISABLE ECC ON 1 SELECTED CSR
4801 030072 013721 002344                 MOV TSTDAT,(R1)+                   :WRITE 16 BITS
4802 030076 104475                         CB1CSR                               :WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
4803 030100 013711 002346                 MOV TSTDAT+2,(R1)                 :WRITE HIGH WORD+CHECKBITS
4804 030104 104503                         CLR1CSR                               :CLEAR 1 SELECTED MK11 CSR
4805                                     :IT'S DANGEROUS IF WE DON'T
4806 030106 012702 002350                 MOV #SBEMSK,R2                     :ADDRESS OF ERROR MASK
4807 030112 162701 000002                 SUB #2,R1                           :ADJUST ADDRESS
4808 030116 112711 177777 4$:           MOVB #-1,(R1)                       :WRITE A BYTE OF 1'S
4809 030122 132712 177777                 BITB #-1,(R2)                       :DID THIS BYTE HAVE THE BAD BIT IN IT?
4810 030126 001420                         BEQ 6$                               :NO - BRANCH
4811 030130 104510                         TSTREAD                              :TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4812 030132 103011                         BCC 5$                               :NO - SKIP
4813 030134                                     SET HEADER                           :ENABLE PRINTING OF ERROR HEADER INFO
4814 030142 010137 002040                 MOV R1,ADDRESS
4815 030146 104017                         ERROR +17
4816 030150                                     SET HEADER                           :ENABLE PRINTING OF ERROR HEADER INFO
4817
4818 030156 111100 5$:                   MOVB (R1),R0
4819 030160 122700 177777                 CMPB #-1,R0                         :CHECK DATA
4820 030164 001414                         BEQ 7$                               :BR IF OK
4821 030166 104457                         PERR33
4822
4823 030170 104510 6$:                   TSTREAD                              :TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4824                                     :READ THE BYTE
4825                                     :SBE ERROR BIT ONLY SET ?
4826 030172 103771                         BCS 5$                               :SHOULD BE SET, BR IF OK
4827 030174                                     SET HEADER                           :ENABLE PRINTING OF ERROR HEADER INFO
4828 030202 010137 002040                 MOV R1,ADDRESS
4829 030206 104460                         PERR34
4830 030210                                     SET HEADER                           :ENABLE PRINTING OF ERROR HEADER INFO
4831
4832 030216 132712 177777 7$:           BITB #-1,(R2)                       :CHECK FOR LAST BYTE
4833 030222 001012                         BNE 8$                               :
4834 030224 005202                         INC R2
4835 030226 005201                         INC R1                               :MOVE TO NEXT BYTE

```

```

4836 030230 013704 002506      MOV     TESTADD,R4      ;FIRST TEST ADDRESS
4837 030234 032701 000002      BIT     #2,R1          ;TEST FOR LOWER WORD
4838 030240 001726              BEQ     4$             ;BR IF IT'S LOW 16 BITS
4839 030242 062704 000002      ADD     #2,R4          ;ADJUST POINTER FOR ERROR REPT.
4840 030246 000723              BR      4$
4841 030250 005737 002352      8$:    TST     SBEMSK+2   ;LAST ERROR BIT ?
4842 030254 100405              BMI     9$             ;MINUS MEANS BIT 31
4843 030256                      DLEFT  SBEMSK
4844 030266 000652              BR      2$
4845 030270                      9$:    IF #SW11 SET.IN @SWR THEN GOTO 10$
4846 030300                      IF QVFLAG IS TRUE THEN GOTO 10$
4847 030306 005737 002342      TST     DATBUF+2       ;LAST DATA BIT?
4848 030312 100405              BMI     10$            ;MINUS = BIT 31
4849 030314                      DLEFT  DATBUF
4850 030324 000626              BR      1$
4851                      ;CLEAR OUT ANY DBE'S OR SBE'S
4852 030326 104471 002506      10$:   ECC1DIS           ;DISABLE ECC ON 1 SELECTED CSR
4853 030330 013701 002506      MOV     TESTADD,R1
4854 030334                      CLEAR  (R1)+,(R1)
4855 030340 104503                      CLR1CSR           ;CLEAR 1 SELECTED MK11 CSR
4856 030342 000207                      RETURN

```

4860 030344

```

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 MK11 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
5$: BEQ      5$
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      9$         ;GO MAKE THEM NOT EQUAL
6$: MOV      #TSTDAT,SOURCE ;SOURCE ADDRESS FOR CHKGEN
CALL     CHKGEN     ;GO GENERATE CHECK BITS
MOV      SBEMSK,R2
XOR      R2,TSTDAT
MOV      SBEMSK+2,R2
XOR      R2,TSTDAT+2
MOV      DBEMSK,R2
XOR      R2,TSTDAT
MOV      DBEMSK+2,R2
XOR      R2,TSTDAT+2
ECC1DIS
MOV      TSTDAT,(R1)+ ;DISABLE ECC ON 1 SELECTED CSR
CB1CSR   ;WRITE 16 BITS
MOV      TSTDAT+2,(R1) ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
CLR1CSR  ;WRITE HIGH WORD
SUB      #2,R1      ;CLEAR 1 SELECTED MK11 CSR
TST      (R1)      ;ADJUST TEST ADDRESS
WAS1DBE ;READ THE LOCATION
BCS      9$        ;WAS THERE ANY DOUBLE BIT ERRORS ON 1 SELECTED CSR
SET      HEADER   ;IT SHOULD BE SET
MOV      R1,ADDRESS
ERROR   +30
SET      HEADER

```

4861
4862 030344 104503
4863 030346 013701 002506
4864 030352 005037 002340
4865 030356 005037 002342
4866 030362 012737 000001 002350
4867 030370 005037 002352
4868 030374 012737 000001 002354
4869 030402 005037 002356
4870 030406 013737 002340 002344
4871 030414 013737 002342 002346
4872 030422 105737 002362
4873 030426 001404
4874 030430 005137 002344
4875 030434 005137 002346
4876 030440 104503
4877 030442 023737 002350 002354
4878 030450 001004
4879 030452 023737 002352 002356
4880 030460 001452
4881 030462 012737 002344 002402
4882 030470 004737 040540
4883 030474 013702 002350
4884 030500 074237 002344
4885 030504 013702 002352
4886 030510 074237 002346
4887 030514 013702 002354
4888 030520 074237 002344
4889 030524 013702 002356
4890 030530 074237 002346
4891 030534 104471
4892 030536 013721 002344
4893 030542 104475
4894 030544 013711 002346
4895 030550 104503
4896 030552 162701 000002
4897 030556 005711
4898 030560 104501
4899 030562 103411
4900 030564
4901 030572 010137 002040
4902 030576 104030
4903 030600

```
4906 030606 005737 002356      9$:  TST    DBEMSK+2      ;CHECK MASK FOR LAST BIT
4907 030612 100405              BMI    10$           ;MINUS = BIT31
4908 030614              DLEFT  DBEMSK
4909 030624 000670              BR     4$
4910 030626      10$:  IF #SW11 SET.IN @SWR THEN GOTO 11$
4911 030636              IF QVFLAG IS TRUE THEN GOTO 11$
4912 030644 005737 002352      TST    SBEMSK+2      ;CHECK SINGLE ERROR MASK TOO
4913 030650 100405              BMI    11$           ;BR IF DONE
4914 030652              DLEFT  SBEMSK
4915 030662 000644              BR     3$
4916 030664 105737 002362      11$:  TSTB   PASFLG ;FIRST PASS
4917 030670 001003              BNE    12$           ;NON ZERO MEANS WE'RE DONE
4918 030672 105237 002362      INCB   PASFLG ;FIRST PASS, NOT DONE
4919              ;CLEAR OUT ANY DBE'S OR SBE'S
4920 030676 000625              BR     1$           ;KEEP GOING
4921 030700 104471      12$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
4922 030702 013701 002506      MOV    TESTADD,R1
4923 030706              CLEAR  (R1)+,(R1)
4924 030712 104503              CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
4925 030714 000207              RETURN
```

4929 030716

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

: *SUBTEST MTP014 WRITE INHIBIT DURING DATIP WITH DBE TEST

```

4930                                     :THIS TEST CHECKS THE WRITE INHIBIT ON DOUBLE
4931                                     :BIT ERRORS DURING A DATIP OPERATION BY USE
4932                                     :OF AN 'ASRB' INSTRUCTION.
4933                                     :
4934                                     :NOTE: THIS IS ONLY A USEFULL TEST ON PDP11/70 WITH THE MULTIPOST ECO
4935                                     :IE THOSE THAT DO A DATIP ON THE ASRB (LOCK) INSTRUCTION.
4936 030716 005037 002340 1$: CLR DATBUF ;INITIAL DATA
4937 030722 005037 002342 CLR DATBUF+2 ;2 WORDS WORTH
4938 030726 012737 000001 002350 2$: MOV #1,SBEMSK ;INITIAL ERROR MASK
4939 030734 005037 002352 CLR SBEMSK+2 ;
4940 030740 012737 000001 002354 3$: MOV #1,DBEMSK ;DOUBLE ERROR MASK
4941 030746 005037 002356 CLR DBEMSK+2 ;2 WORDS
4942 030752 013737 002340 002344 4$: MOV DATBUF,TSTDAT ;PRESERVE ORIG DATA
4943 030760 013737 002342 002346 MOV DATBUF+2,TSTDAT+2
4944 030766 105737 002362 TSTB PASFLG ;SECOND PASS YET ?
4945 030772 001404 BEQ 5$ ;BR IF NO
4946 030774 005137 002344 COM TSTDAT ;COMPL DATA ON SECOND PASS
4947 031000 005137 002346 COM TSTDAT+2
4948 031004 104503 5$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
4949 031006 023737 002354 002350 CMP DBEMSK,SBEMSK ;CHECK FOR SAME MASKS
4950 031014 001004 BNE 6$ ;BR IF OK
4951 031016 023737 002356 002352 CMP DBEMSK+2,SBEMSK+2
4952 031024 001466 BEQ 11$ ;BR IF THEY'RE EQUAL
4953 031026 012737 002344 002402 6$: MOV #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
4954 031034 004737 040540 CALL CHKGEN ;GENERATE CHECK BITS
4955 031040 013701 002350 MOV SBEMSK,R1
4956 031044 074137 002344 XOR R1,TSTDAT
4957 031050 013701 002352 MOV SBEMSK+2,R1
4958 031054 074137 002346 XOR R1,TSTDAT+2
4959 031060 013701 002354 MOV DBEMSK,R1
4960 031064 074137 002344 XOR R1,TSTDAT
4961 031070 013701 002356 MOV DBEMSK+2,R1
4962 031074 074137 002346 XOR R1,TSTDAT+2
4963 031100 013701 002506 7$: MOV TESTADD,R1 ;TEST ADDRESS
4964 031104 104471 ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
4965 031106 013721 002344 MOV TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
4966 031112 104475 CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
4967 031114 013711 002346 MOV TSTDAT+2,(R1) ;SECOND 16 BITS+CHECKBITS
4968 031120 105037 002363 CLRB UPPFLG ;INDICATE LOWER WORD
4969 031124 013703 002506 MOV TESTADD,R3 ;TEST ADDRESS
4970 031130 104503 8$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
4971 031132 106223 ASRB (R3)+ ;SPECIAL DATIP INSTRUCTION
4972 031134 014100 MOV -(R1),R0
4973 031136 023700 002344 CMP TSTDAT,R0 ;CHECK FOR UNCHANGED DATA
4974 031142 001401 BEQ 9$ ;SHOULD BE UNCHANGED
4975 031144 104455 PERR31
4976
4977 031146 062701 000002 9$: ADD #2,R1 ;POINT TO UPPER WORD
4978 031152 011100 MOV (R1),R0
4979 031154 023700 002346 CMP TSTDAT+2,R0 ;READ IT
4980 031160 001401 BEQ 10$ ;BR IF UNCHANGED
4981 031162 104455 PERR31
4982

```

```

4983 031164 122737 000003 002363 10$:  CMPB  #3,UPPFLG          ;LOWER WORD
4984 031172 001403          BEQ    11$              ;BR IF NO
4985 031174 105237 002363          INCB  UPPFLG
4986 031200 000753          BR    8$
4987 031202 005737 002356          11$:  TST   DBEMSK+2        ;LAST BIT IN MASK ?
4988 031206 100405          BMI   12$              ;BR IF BIT 31
4989 031210          DLEFT DBEMSK
4990 031220 000654          BR    4$
4991 031222          12$:  IF #SW11 SET.IN @SWR THEN GOTO 13$
4992 031232          IF QVFLAG IS TRUE THEN GOTO 13$
4993 031240 005737 002352          TST   SBEMSK+2        ;LAST BIT IN SINGLE ERROR MASK ?
4994 031244 100405          BMI   13$              ;BR IF YES
4995 031246          DLEFT SBEMSK
4996 031256 000630          BR    3$
4997 031260 105737 002362          13$:  TSTB  PASFLG ;WHICH PASS
4998 031264 001003          BNE   14$              ;BR IF WE'RE DONE
4999 031266 105237 002362          INCB  PASFLG ;INDICATE SECOND PASS COMING
5000          ;CLEAR OUT ANY DBE'S OR SBE'S
5001 031272 000611          BR    1$              ;GO DO IT!
5002 031274 104471          14$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5003 031276 013701 002506          MOV   TESTADD,R1
5004 031302          CLEAR (R1)+,(R1)
5005 031306 104503          CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
5006 031310 000207          RETURN
  
```

5010 031312

MTP015: SUBSTST <<MTP015 WRITE INHIBIT OF BYTE WITH DBE>>

: *SUBTEST MTP015 WRITE INHIBIT OF BYTE WITH DBE

```

5011                                     ;CHECK FOR WRITE INHIBIT DURING A WRITE BYTE.
5012                                     ;CHECKS FOR UNCORRECTED DATA.
5013 031312 005037 002340 1$: CLR DATBUF ;INITIAL DATA
5014 031316 005037 002342 CLR DATBUF+2 ;32 BITS WORTH
5015 031322 012737 000001 002350 2$: MOV #1,SBEMSK ;SINGLE ERROR MASK
5016 031330 005037 002352 CLR SBEMSK+2 ;
5017 031334 012737 000001 002354 3$: MOV #1,DBEMSK ;DOUBLE ERROR MASK
5018 031342 005037 002356 CLR DBEMSK+2 ;
5019 031346 013737 002340 002344 4$: MOV DATBUF,TSTDAT ;PRESERVE ORIG DATA
5020 031354 013737 002342 002346 MOV DATBUF+2,TSTDAT+2
5021 031362 105737 002362 TSTB PASFLG ;WHICH PASS ?
5022 031366 001404 BEQ 5$ ;FIRST PASS, NO COMPLEMENTING
5023 031370 005137 002344 COM TSTDAT
5024 031374 005137 002346 COM TSTDAT+2 ;SECOND PASS, COMPLEMENT TSTDAT
5025 031400 104503 CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
5026 031402 023737 002350 002354 CMP SBEMSK,DBEMSK ;CHECK FOR SAME MASKS
5027 031410 001004 BNE 6$ ;BR IF NOT EQUAL
5028 031412 023737 002352 002356 CMP SBEMSK+2,DBEMSK+2 ;SECOND WORD ALSO
5029 031420 001463 BEQ 11$ ;BR TO MAKE THEM NOT EQUAL
5030 031422 012737 002344 002402 6$: MOV #TSTDAT,SOURCE ;ADDRESS FOR CHKGEN
5031 031430 004737 040540 CALL CHKGEN ;GO GENERATE CHECK BITS
5032 031434 013701 002350 MOV SBEMSK,R1
5033 031440 074137 002344 XOR R1,TSTDAT
5034 031444 013701 002352 MOV SBEMSK+2,R1
5035 031450 074137 002346 XOR R1,TSTDAT+2
5036 031454 013701 002354 MOV DBEMSK,R1
5037 031460 074137 002344 XOR R1,TSTDAT
5038 031464 013701 002356 MOV DBEMSK+2,R1
5039 031470 074137 002346 XOR R1,TSTDAT+2
5040 031474 013701 002506 7$: MOV TESTADD,R1 ;TEST LOCATION
5041 031500 104471 ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5042 031502 013721 002344 MOV TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
5043 ;LOAD CSR WITH IMAGE FROM R2
5044 031506 104475 CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5045 031510 013711 002346 MOV TSTDAT+2,(R1) ;WRITE SECOND 16 BITS + CHECKBITS
5046 031514 104503 CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
5047 031516 013702 002506 MOV TESTADD,R2 ;GET ADDRESS OF TEST LOC
5048 031522 010203 MOV R2,R3 ;R2 DESIGNATES FIRST BYTE
5049 031524 062703 000003 ADD #3,R3 ;R3 DESIGNATES LAST BYTE
5050 031530 112722 000360 8$: MOVB #360,(R2)+ ;TRY WRITING A BYTE
5051 031534 013701 002506 MOV TESTADD,R1
5052 031540 012100 MOV (R1)+,R0
5053 031542 023700 002344 CMP TSTDAT,R0 ;CHECK FOR UNCHANGED DATA
5054 031546 001401 BEQ 9$ ;BR IF OK
5055 031550 104455 PERR31
5056
5057 031552 011100 9$: MOV (R1),R0
5058 031554 023700 002346 CMP TSTDAT+2,R0 ;READ SECOND WORD
5059 031560 001401 BEQ 10$ ;BR IF UNCHANGED
5060 031562 104455 PERR31
5061
5062 031564 020203 10$: CMP R2,R3 ;TESTED LAST BYTE ?
5063 031566 001360 BNE 8$ ;BR IF NO

```

```
5064 031570 005737 002356      11$:  TST      DBEMSK+2      ;CHECKING FOR LAST ERROR BIT
5065 031574 100405              BMI      12$           ;BR IF DONE HERE
5066 031576              DLEFT   DBEMSK
5067 031606 000657              BR      4$
5068 031610              12$:  IF #SW11 SET.IN @SWR THEN GOTO 13$
5069 031620              IF QVFLAG IS TRUE THEN GOTO 13$
5070 031626 005737 002352      TST      SBEMSK+2      ;LAST SBE MASK
5071 031632 100405              BMI      13$           ;BR IF DONE WITH THIS PASS
5072 031634              DLEFT   SBEMSK
5073 031644 000633              BR      3$
5074 031646 105737 002362      13$:  TSTB     PASFLG      ;TEST PASS FLAG
5075 031652 001003              BNE     14$           ;NON ZERO MEANS WE'RE DONE
5076 031654 105237 002362      INCB    PASFLG      ;NOT DONR
5077 031660 000614              BR      1$
5078 031662 104471              14$:  ECC1DIS           ;DISABLE ECC ON 1 SELECTED CSR
5079 031664 013701 002506      MOV     TESTADD,R1    ;TEST LOCATION
5080 031670              CLEAR  (R1)+,(R1)    ;TO ERASE ANY DBE'S FROM TESTING
5081              ;RESTORE CSR
5082 031674 104503              CLR1CSR           ;CLEAR 1 SELECTED MK11 CSR
5083 031676 000207              RETURN
```

5087 031700

MTP016: SUBST <<MTP016 WRITE INHIBIT OF WORD WITH DBE>>
:*****
:*SUBTEST MTP016 WRITE INHIBIT OF WORD WITH DBE
:*****

5088
5089
5090
5091

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

5092 031700 005037 002340
5093 031704 005037 002342
5094 031710 012737 000001 002350
5095 031716 005037 002352
5096 031722 012737 000001 002354
5097 031730 005037 002356
5098 031734 013737 002340 002344
5099 031742 013737 002342 002346
5100 031750 105737 002362
5101 031754 001404
5102 031756 005137 002344
5103 031762 005137 002346
5104 031766 023737 002350 002354
5105 031774 001004
5106 031776 023737 002352 002356
5107 032004 001470
5108 032006 012737 002344 002402
5109 032014 004737 040540
5110 032020 013701 002350
5111 032024 074137 002344
5112 032030 013701 002352
5113 032034 074137 002346
5114 032040 013701 002354
5115 032044 074137 002344
5116 032050 013701 002356
5117 032054 074137 002346
5118 032060 013701 002506
5119 032064 104471
5120 032066 013721 002344
5121 032072 104475
5122 032074 013711 002346
5123 032100 105037 002363
5124 032104 162701 000002
5125 032110 104503
5126 032112 012711 177400
5127 032116 013701 002506
5128 032122 011100
5129 032124 023700 002344
5130 032130 001401
5131 032132 104455
5132
5133 032134 062701 000002
5134 032140 011100
5135 032142 023700 002346
5136 032146 001401
5137 032150 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 MK11 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
PERR31
6\$: ADD #2,R1
MOV (R1),R0
CMP TSTDAT+2,R0 ;THIS SHOULD BE UNCHANGED ALSO
BEQ 7\$
PERR31

```

5140 032152 105737 002363      7$:  TSTB  UPPFLG      ;WHICH LOOP ?
5141 032156 001003              BNE   8$          ;SECOND, BR OUT
5142 032160 105237 002363      INCB  UPPFLG      ;FIRST, KEEP GOING
5143 032164 000751              BR    5$
5144 032166 005737 002356      8$:  TST   DBEMSK+2  ;LAST BIT ?
5145 032172 100405              BMI   9$          ;MINUS = BIT 31
5146 032174              DLEFT DBEMSK
5147 032204 000653              BR    1$
5148 032206      9$:  IF #SW11 SET.IN @SWR THEN GOTO 10$
5149 032216              IF QVFLAG IS TRUE THEN GOTO 10$
5150 032224 005737 002352      TST   SBEMSK+2    ;LAST BIT IN THIS MASK ?
5151 032230 100406              BMI   10$         ;BR IF LAST BIT
5152 032232              DLEFT SBEMSK
5153 032242 000137 031722      JMP   T12B
5154 032246 105737 002362      10$: TSTB  PASFLG ;FIRST PASS ?
5155 032252 001004              BNE   11$         ;BR IF SECOND
5156 032254 105237 002362      INCB  PASFLG ;INDICATE SECOND PASS COMING
5157 032260 000137 031700      JMP   T12A
5158 032264 104471      11$: ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5159 032266 013701 002506      MOV   TESTADD,R1 ;RESTORE TEST ADDRESS
5160 032272 005021              CLR   (R1)+       ;CLEAR ANY DBE'S FROM TEST
5161 032274 005011              CLR   (R1)
5162 032276 104503              CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
5163 032300 000207              RETURN
    
```

5167 032302

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

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

```

5168          :*(1) THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
5169          :*    OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
5170          :*    OF 000377 AND READING IT
5171          :*(2) MEMORY IS WRITTEN USING A BYTE AT A TIME
5172          :*(3) STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
5173          :NOTE: THIS TEST WRITES BYTES & READS WORDS
5174 032302 012701 060000      MOV    #FIRST,R1
5175 032306 010104            MOV    R1,R4
5176 032310 012705 160000      MOV    #LAST+2,R5
5177 032314 012700 000377      MOV    #377,R0          ;GET THE PATTERN INTO R0
5178 032320 010003            MOV    R0,R3
5179 032322 000303            SWAB   R3
5180 032324 110021 1$:      MOVB   R0,(R1)+          ;WRITE A BYTE
5181 032326 110321            MOVB   R3,(R1)+          ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
5182 032330 020105            CMP    R1,R5            ;COMPARE TEST LOC TO TOP + 2
5183 032332 103774            BLO   1$              ;BRANCH IF LOWER
5184
5185 032334 014102 2$:      MOV    -(R1),R2
5186 032336 020002            CMP    R0,R2            ;TEST THE MEMORY TO SEE IF IT CONTAINS
5187                                     ;THE WORD STORED IN BAKPAT
5188 032340 001401            BEQ   3$
5189 032342 104446            PERR22
5190
5191 032344 020104 3$:      CMP    R1,R4            ;KEEP ON TESTING THE MEMORY UNTIL
5192 032346 101372            BHI   2$              ;R1 EQUALS THE LOWEST ADDRESS
5193 032350 000303            SWAB   R3              ;CHANGE THE DATA PATTERN
5194 032352 000300            SWAB   R0
5195 032354 001763            BEQ   1$              ;IF THE DATA PATTERN DOES NOT HAVE LOW
5196                                     ;BYTE =0 THEN FALL THRU
5197 032356 000207            RETURN

```

5201 032360

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.

5202
5203
5204
5205
5206 032360 010311
5207 032362 010461 000002
5208 032366 060201
5209 032370 020105
5210 032372 103772
5211 032374 000207
5212
5213
5214 032376 160201
5215 032400 020461 000002
5216 032404 001002
5217 032406 005711
5218 032410 001401
5219 032412 104452
5220 032414 010361 000002
5221 032420 010311
5222 032422 020100
5223 032424 001364
5224 032426 000207
5225
5226
5227 032430 005711
5228 032432 001003
5229 032434 005761 000002
5230 032440 001401
5231 032442 104453
5232 032444 010311
5233 032446 010461 000002
5234 032452 060201
5235 032454 020105
5236 032456 103764
5237 032460 000207
5238
5239
5240 032462 005711
5241 032464 001003
5242 032466 020461 000002
5243 032472 001401
5244 032474 104452
5245 032476 010311
5246 032500 010361 000002
5247 032504 060201
5248 032506 020105
5249 032510 103764
5250 032512 000207

```

:WRITE UP ;100 --> CHECKBITS
MTP20A: MOV R3,(R1) ;V177640
MOV R4,2(R1) ;V177642
ADD R2,R1 ;V177646
CMP R1,R5 ;V177650 ;TOP + 2 ?
BLO MTP20A ;V177652 ;BR IF NOT
RETURN ;V177654

:100 --> 077 DOWN
MTP20B: SUB R2,R1 ;V177640
CMP R4,2(R1) ;V177642 ;2ND WORD OK?
BNE 21$ ;V177646 ;NO - SKIP
TST (R1) ;V177650 ;1ST WORD OK?
BEQ 3$ ;V177652 ;YES - SKIP
21$: PERR26 ;V177654 ;GOOD=000000,,100000,,100
3$: MOV R3,2(R1) ;V177656
MOV R3,(R1) ;V177662
CMP R1,R0 ;V177664
BNE MTP20B ;V177666
RETURN ;V177670

:077 --> 100 UP
MTP20C: TST (R1) ;V177640 ;1ST WORD OK?
BNE 41$ ;V177642 ;NO - SKIP
TST 2(R1) ;V177644 ;2ND WORD OK?
BEQ 5$ ;V177650 ;YES - SKIP
41$: PERR27 ;V177652 ;GOOD=000000,,000000,,077
5$: MOV R3,(R1) ;V177654 ;WRITE 1ST WORD
MOV R4,2(R1) ;V177656 ;WRITE 2ND WORD
ADD R2,R1 ;V177662
CMP R1,R5 ;V177664 ;TOP + 2 YET?
BLO MTP20C ;V177666 ;NO - LOOP
RETURN ;V177670

:100 --> 077 UP
MTP20D: TST (R1) ;V177640 ;1ST WORD OK?
BNE 61$ ;V177642 ;NO - SKIP
CMP R4,2(R1) ;V177644 ;2ND WORD OK?
BEQ 7$ ;V177650 ;YES - SKIP
61$: PERR26 ;V177652
7$: MOV R3,(R1) ;V177654
MOV R3,2(R1) ;V177656
ADD R2,R1 ;V177662
CMP R1,R5 ;V177664 ;TOP + 2 YET?
BLO MTP20D ;V177666 ;NO - LOOP
RETURN ;V177670

```

5253					:077 --> 100 DOWN			
5254	032514	160201		MTP20E:	SUB R2,R1	:V177640		
5255	032516	005761	000002		TST 2(R1)	:V177642	:2ND WORD OK?	
5256	032522	001002			BNE 81\$:V177646	:NO - SKIP	
5257	032524	005711			TST (R1)	:V177650	:1ST WORD OK?	
5258	032526	001401			BEQ 9\$:V177652	:YES - SKIP	
5259	032530	104453		81\$:	PERR27	:V177654		
5260	032532	010461	000002	9\$:	MOV R4,2(R1)	:V177656	:WRITE 2ND WORD	
5261	032536	010311			MOV R3,(R1)	:V177662	:WRITE 1ST WORD	
5262	032540	020100			CMP R1,R0	:V177664	:BOTTOM YET?	
5263	032542	001364			BNE MTP20E	:V177666	:NO - LOOP	
5264	032544	000207			RETURN	:V177670		
5265								
5266					:100 UP			
5267	032546	005711		MTP20F:	TST (R1)	:V177640	:1ST WORD OK?	
5268	032550	001003			BNE 101\$:V177642	:NO - SKIP	
5269	032552	020461	000002		CMP R4,2(R1)	:V177642	:2ND WORD OK?	
5270	032556	001401			BEQ 11\$:V177646	:YES - SKIP	
5271	032560	104452		101\$:	PERR26	:V177650		
5272	032562	060201		11\$:	ADD R2,R1	:V177652		
5273	032564	020105			CMP R1,R5	:V177654	:TOP + 2 YET?	
5274	032566	103767			BLO MTP20F	:V177656	:NO - LOOP	
5275	032570	000207			RETURN	:V177660		

5279 032572

MTPA20: SUBTST <<MTPA20 SLOW MARCHING 1'S & 0'S IN CHECK BITS TEST>>

:*SUBTEST MTPA20 SLOW MARCHING 1'S & 0'S IN CHECK BITS TEST

5280
5281
5282 032572
5283 032600 013737 002144 002150
5284 032606 013737 002146 002152
5285 032614 013701 002336
5286 032620 013702 002414
5287 032624 010103
5288 032626 005004
5289 032630 012705 160000
5290 032634 005037 002144
5291 032640 104505

:*THIS TEST IS CONCERNED ONLY WITH THE INTEGRITY
:*OF THE MOS RAMS THAT STORE THE CHECKBITS.
SET SBFLAG
MOV CSR,SBECRSR
MOV CSR+2,SBECRSR+2
MOV CSRFIRST,R1 ;FIRST TEST LOC
MOV CSRINC,R2
MOV R1,R3
CLR R4
MOV #LAST+2,R5
CLR CSR
CHK1DIS ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR

5292
5293
5294 032642 010411
5295 032644 060201
5296 032646 020105
5297 032650 103774
5298
5299

2\$: MOV R4,(R1) ;WRITE CHECKBITS BY WRITING LOC * WRITE UP
ADD R2,R1 ;POINT TO NEXT WORD * 0'S --> CHK BITS
CMP R1,R5 ;TOP + 2 *
BLO 2\$;BR IF NOT *

5300
5301 032652 160201
5302 032654 005711
5303 032656 104426
5304 032660 032737 077400 002144
5305 032666 001406
5306 032670 013700 002144
5307 032674 000300
5308 032676 042700 000200
5309 032702 104435

3\$: SUB R2,R1 ;ADJUST R1 *
TST (R1) ;READ *
READCSR ; *
BIT #77400,@#CSR ;SHOULD BE ALL ZEROS *
BEQ 4\$;BR IF OK *
MOV CSR,R0 ; *
SWAB R0 ; *
BIC #200,R0 ; *
PERR11 ; *

5310
5311 032704 012737 077400 002144 4\$:
5312 032712 104505
5313 032714 005104
5314 032716 010411
5315 032720 020103
5316 032722 001353
5317

MOV #77400,@#CSR ;MAKE CHECKBITS ALL 1'S *0'S --> 1'S
CHK1DIS ;DSABL ECC & WRITE CHKBITS FROM 1 CSR * DOWN
COM R4 ;COMP THE DATA *
MOV R4,(R1) ;WRITE THE CHECKBITS *
CMP R1,R3 ;BOTTOM YET? *
BNE 3\$;BR IF NO *

5318
5319 032724 005037 002144
5320 032730 104505

CLR CSR
CHK1DIS ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR

```

5323
5324 032732 005711          5$:  TST      (R1)          ;READ THE LOC AGAIN
5325 032734 104426          READCSR
5326 032736 013700 002144  MOV     @#CSR,R0      ;GET CSR CONTENTS INTO R0
5327 032742 042700 100377  BIC     #100377,R0    ;ONLY WANT BITS 11-5
5328 032746 022700 077400  CMP     #77400,R0    ;READ FOR ALL 1'S
5329 032752 001404          BEQ     6$            ;BR IF OK
5330 032754 000300          SWAB   R0
5331 032756 042700 000200  BIC     #200,R0
5332 032762 104445          PERR21
5333
5334 032764 005037 002144  6$:  CLR     CSR
5335 032770 104505          CHK1DIS ;DSABL ECC & WRITE CHKBTS FROM 1 CSR
5336 032772 005104          COM    R4
5337 032774 010411          MOV     R4,(R1)      ;ACCESS THE LOC
5338 032776 060201          ADD    R2,R1        ;POINT TO NEXT ADDRESS
5339 033000 020105          CMP    R1,R5        ;TOP+2 YET?
5340 033002 103753          BLO    5$            ;BR IF NOT
5341
5342
5343 033004 010301          MOV     R3,R1
5344
5345
5346 033006 005711          7$:  TST      (R1)          ;READ
5347 033010 104426          READCSR
5348 033012 032737 077400 002144  BIT     #77400,@#CSR  ;CHECKBITS SHOULD BE ZERO
5349 033020 001406          BEQ     8$            ;BR IF OK
5350 033022 013700 002144  MOV     CSR,R0
5351 033026 000300          SWAB   R0
5352 033030 042700 000200  BIC     #200,R0
5353 033034 104435          PERR11
5354
5355 033036 012737 077400 002144  8$:  MOV     #77400,@#CSR ;COMP CHECKBITS
5356 033044 104505          CHK1DIS ;DSABL ECC & WRITE CHKBTS FROM 1 CSR
5357 033046 005104          COM    R4
5358 033050 010411          MOV     R4,(R1)      ;WRITE THE LOC
5359 033052 060201          ADD    R2,R1        ;POINT TO NEXT ADDRESS
5360 033054 020105          CMP    R1,R5        ;TOP+2 YET?
5361 033056 103753          BLO    7$            ;BR IF NOT
5362
5363
5364 033060 005037 002144  CLR     CSR
5365 033064 104505          CHK1DIS ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR

```

*1'S --> 0'S
* UP

*0'S --> 1'S
* UP

```

5368
5369 033066 160201          9$:  SUB    R2,R1      ;ADJUST R1
5370 033070 005711          TST    (R1)       ;READ
5371 033072 104426          READCSR
5372 033074 013700 002144  MOV    @#CSR,R0   ;GET CSR CONTENTS
5373 033100 042700 100377  BIC    #100377,R0 ;ONLY WANT CHECKBITS
5374 033104 022700 077400  CMP    #77400,R0 ;ALL 1'S?
5375 033110 001404          BEQ    10$        ;BR IF OK
5376 033112 000300          SWAB   R0
5377 033114 042700 000200  BIC    #200,R0
5378 033120 104445          PERR21
5379
5380 033122 005037 002144  10$:  CLR    CSR
5381 033126 104505          CHK1DIS ;DSABL ECC & WRITE CHKBTS FROM 1 CSR
5382 033130 005104          COM    R4        ;COMP THE DATA
5383 033132 010411          MOV    R4,(R1)   ;WRITE THE CHECKBITS
5384 033134 020103          CMP    R1,R3    ;BOTTOM YET?
5385 033136 001353          BNE    9$        ;BR IF NO
5386
5387
5388
5389 033140 005711          11$:  TST    (R1)       ;READ
5390 033142 104426          READCSR
5391 033144 032737 077400 002144  BIT    #77400,@#CSR ;SHOULD BE ALL 0'S
5392 033152 001406          BEQ    12$        ;BR IF OK
5393 033154 013700 002144  MOV    CSR,R0
5394 033160 000300          SWAB   R0
5395 033162 042700 000200  BIC    #200,R0
5396 033166 104435          PERR11
5397
5398 033170 060201          12$:  ADD    R2,R1     ;POINT TO NEXT WORD
5399 033172 005037 002144  CLR    CSR
5400 033176 104505          CHK1DIS ;DSABL ECC & WRITE CHKBTS FROM 1 CSR
5401 033200 020105          CMP    R1,R5    ;TOP + 2 YET?
5402 033202 103756          BLO    11$      ;BR IF NOT
5403
5404
5405
5406 033204          ;FLUSH OUT ALL THESE DBE'S
5407 033212 012700 060000  BMOV   MTP030
5408 033216 012701 040000  MOV    #FIRST,R0
5409 033222 104471          MOV    #SIZE,R1
5410 033224 004737 177640  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5411 033230 104503          CALL   FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
5412 033232 000207          CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
          RETURN

```

*1'S --> 0'S
* DOWN

*0'S
* UP

5416 033234
 5417
 5418 033234 014100
 5419 033236 020200
 5420 033240 001401
 5421 033242 104443
 5422
 5423 033244 000311
 5424 033246 011100
 5425 033250 020300
 5426 033252 001401
 5427 033254 104444
 5428
 5429 033256 020401
 5430 033260 001365
 5431 033262 000207
 5432
 5433 033264
 5434 033264 011100
 5435 033266 020300
 5436 033270 001401
 5437 033272 104444
 5438
 5439 033274 000311
 5440 033276 011100
 5441 033300 020200
 5442 033302 001401
 5443 033304 104443
 5444
 5445 033306 062701 000002
 5446 033312 020501
 5447 033314 001363
 5448 033316 000207
 5449
 5450 033320
 5451 033320 011100
 5452 033322 020200
 5453 033324 001401
 5454 033326 104443
 5455
 5456 033330 000311
 5457 033332 011100
 5458 033334 020300
 5459 033336 001401
 5460 033340 104444
 5461
 5462 033342 062701 000002
 5463 033346 020501
 5464 033350 001363
 5465 033352 000207

```

MTPA21: SUBTST <<MTPA21      MARCHING 1'S & 0'S PATTERN TEST>>
:*****
:*SUBTEST      MTPA21  MARCHING 1'S & 0'S PATTERN TEST
:*****
:READ,BYTESWAP-MODIFY,READ,DOWN
1$:  MOV      -(R1),R0 :V177640
     CMP      R2,R0   :V177642
     BEQ      2$      :V177644
     PERR17          :V177646

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

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

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

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

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

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

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

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

```

5468 033354
5469 033354 014100
5470 033356 020300
5471 033360 001401
5472 033362 104444
5473
5474 033364 000311
5475 033366 011100
5476 033370 020200
5477 033372 001401
5478 033374 104443
5479
5480 033376 020401
5481 033400 001365
5482 033402 000207
5483

MTPD21: ;READ,BYTESWAP-MODIFY,READ,DOWN
1$:      MOV      -(R1),R0 :V177640
          CMP      R3,R0   :V177642
          BEQ      2$      :V177644
          PERR20          :V177646

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

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

```

5487 033404

```
MTP022: SUBST <<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
```

5488
5489
5490
5491
5492 000010
5493 033404
5494 033412
5495 033422
5496 033426
5497 033432
5498 033434
5499 033440
5500 033444
5501 033444
5502
5503
5504 033450 104423
5505 033452
5506 033460
5507 033464
5508 033472
5509 033506
5510 033520
5511 033526
5512 033530
5513 033534
5514 033536
5515 033540
5516 033544
5517 033544
5518 033550
5519 033554
5520
5521
5522 033556

5525
 5526 033570
 5527 033576
 5528 033602 104424
 5529 033604
 5530 033612
 5531 033626
 5532 033640
 5533 033646
 5534 033650
 5535 033654 104443
 5536 033656
 5537 033656
 5538 033662
 5539 033666 104443
 5540 033670
 5541 033670
 5542 033672
 5543 033674
 5544 033700 104444
 5545 033702
 5546 033702
 5547 033706
 5548 033712 104444
 5549 033714
 5550 033714
 5551 033714
 5552 033720
 5553 033724
 5554
 5555
 5556 033726
 5557 033742
 5558 033756 000207
 5559 033760

 5560
 5561 033760
 5562 033764 004737 034030
 5563 033770
 5564 034002
 5565 034006
 5566 034014 004737 034030
 5567 034020
 5568 034024
 5569 034026 000207
 5570 034030 012704 000640
 5571 034034 062700 000002
 5572 034040 005140
 5573 034042 005120
 5574 034044 005110
 5575 034046 005110
 5576 034050 077405
 5577 034052 162700 000002
 5578 034056 000207

```

;READ LOOP
LET COUNT := STRIPES
LET R1 := #FIRST
CACHOFF ;TURN CACHE OFF
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
  
```

5582 034060

MTPA24: SUBTST <<MTPA24 FAST GALLOPING PATTERN TEST>>

 :SUBTEST MTPA24 FAST GALLOPING PATTERN TEST

5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603
5604
5605
5606
5607

```

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

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

5608 034060 011100
 5609 034062 020004
 5610 034064 001401
 5611 034066 104447
 5612
 5613 034070 011200
 5614 034072 020003
 5615 034074 001401
 5616 034076 104450
 5617
 5618 034100 062702 000400
 5619 034104 020205
 5620 034106 101764
 5621
 5622 034110 062701 000002
 5623 034114 000137 172260

```

:UIPAR'S
1$: MOV (R1),R0 :V177640 :READ 'A'
   CMP R0,R4 :V177642 :CHECK 'A'
   BEQ 2$ :V177644 :BR IF OK
   PERR23 :V177646 :REPORT ERROR

2$: MOV (R2),R0 :V177650 :READ 'B'
   CMP R0,R3 :V177652 :CHECK 'B'
   BEQ 3$ :V177654 :BR IF OK
   PERR24 :V177656 :REPORT ERROR

3$: ADD #400,R2 :V177660 :BUMP 'B'
   CMP R2,R5 :V177664 :AT END YET?
   BLOS 1$ :V177666 :BR IF NO

   ADD #2,R1 :V177670 :BUMP 'A'
   JMP @WSDPAR0 :V177674 :GOTO V177260
  
```

5626 034120

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

5627
5628 034120 010411
5629 034122 020105
5630 034124 001001
5631 034126 000207
5632 034130 000137 172360
5633
5634 034134

: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 @#KDPAR0 ;V172270 ;GOTO V172360

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

5635
5636 034134 010102
5637 034136 011100
5638 034140 020004
5639 034142 001401
5640 034144 104447
5641 034146 000137 177660

: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 @#UDPAR0 ;V172372 ;GOTO V177660

5645 034152

MTP025: SUBTST <<MTP025 INTERRUPT ENABLE TEST>>
:*****
:*SUBTST MTP025 INTERRUPT ENABLE TEST
:*****

5646	034152	005037	002344		CLR	TSTDAT	:GENERATE CHECKBITS ON 0,,0
5647	034156	005037	002346		CLR	TSTDAT+2	
5648	034162	012737	002344	002402	MOV	#TSTDAT,SOURCE	
5649	034170	004737	040540		CALL	CHKGEN	
5650	034174	012737	000003	002100	MOV	#3,NOPAR	:SETUP PARITY ACTION
5651	034202	013701	002506		MOV	TESTADD,R1	:FIRST TEST ADDRESS
5652	034206	012737	034242	002372	MOV	#1\$,PARTHERE	:SETUP TRAP DESTINATION
5653	034214	004737	034464		CALL	MTPA25	:WRITE DATA & CHECKBITS
5654	034220	104473			ECC1INIT		:INITIALIZE 1 SELECTED MK11 CSR
5655	034222	005711			TST	(R1)	:ACCESS LOCATIONS FOR DBE TRAPS
5656	034224	005761	000002		TST	2(R1)	
5657						:NONE - GOOD - ACCESS FOR SBE TRAPS	
5658	034230	104507			ENA1SBE		:DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
5659	034232	005711			TST	(R1)	
5660	034234	005761	000002		TST	2(R1)	
5661	034240	000404			BR	2\$:NONE - GOOD - SKIP
5662	034242	104426			1\$: READCSR		
5663	034244				FATAL	27	
5664	034252	005237	002344		2\$: INC	TSTDAT	:CHECK FOR CORRECT ACTION ON SBE'S
5665	034256	004737	034412		CALL	MTPD25	:IN ALL 4 BYTES
5666	034262	012737	000400	002344	MOV	#400,TSTDAT	
5667	034270	004737	034412		CALL	MTPD25	
5668	034274	005037	002344		CLR	TSTDAT	
5669	034300	005237	002346		INC	TSTDAT+2	
5670	034304	004737	034412		CALL	MTPD25	
5671	034310	012737	000400	002346	MOV	#400,TSTDAT+2	
5672	034316	004737	034412		CALL	MTPD25	
5673							
5674	034322	005037	002346		CLR	TSTDAT+2	:CHECK FOR CORRECT ACTION ON DBE'S
5675	034326	012737	000003	002344	MOV	#3,TSTDAT	:IN ALL 4 BYTES
5676	034334	004737	034434		CALL	MTPE25	
5677	034340	012737	001400	002344	MOV	#1400,TSTDAT	
5678	034346	004737	034434		CALL	MTPE25	
5679	034352	005037	002344		CLR	TSTDAT	
5680	034356	012737	000003	002346	MOV	#3,TSTDAT+2	
5681	034364	004737	034434		CALL	MTPE25	
5682	034370	012737	001400	002346	MOV	#1400,TSTDAT+2	
5683	034376	004737	034434		CALL	MTPE25	
5684	034402	104503			CLR1CSR		:CLEAR 1 SELECTED MK11 CSR
5685	034404	005037	002100		CLR	NOPAR	:INDICATE PARITY ACTION
5686	034410	000207			RETURN		
5687							
5688	034412	004737	034464		MTPD25: CALL	MTPA25	:WRITE DATA & CHECKBITS
5689	034416	104471			ECC1DIS		:DISABLE ECC ON 1 SELECTED CSR
5690	034420	004737	034504		CALL	MTPB25	:CHECK FOR NO TRAPS
5691	034424	104507			ENA1SBE		:DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
5692	034426	004737	034536		CALL	MTPC25	:CHECK FOR EXPECTED TRAP
5693	034432	000207			RETURN		

```

5696 034434 004737 034464      MTP25: CALL      MTPA25      ;WRITE DATA & CHECKBITS
5697 034440 104471              ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
5698 034442 004737 034504      CALL      MTPB25      ;CHECK FOR NO TRAPS
5699              ;ENABLE DBE TRAPS
5700 034446 104473              ECC1INIT      ;INITIALIZE 1 SELECTED MK11 CSR
5701 034450 004737 034536      CALL      MTPC25      ;CHECK FOR EXPECTED TRAP
5702 034454 104507              ENA1SBE      ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
5703 034456 004737 034536      CALL      MTPC25      ;CHECK FOR EXPECTED TRAP
5704 034462 000207              RETURN
5705
5706              ;WRITE TSTDAT & TSTDAT+2 & CHECKBITS
5707 034464 104471      MTPA25: ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
5708 034466 013711 002344      MOV      TSTDAT,(R1) ;WRITE FIRST 16 BITS
5709 034472 104475              CB1CSR      ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5710 034474 013761 002346 000002      MOV      TSTDAT+2,2(R1) ;WRITE 2ND 16 BITS & CHECKBITS
5711 034502 000207              RETURN
5712
5713              ;CHECK FOR NO TRAP OCCURING CONDITION
5714 034504 012737 034522 002372      MTPB25: MOV      #1$,PARTHERE ;SETUP TRAP DESTINATION
5715 034512 005711              TST      (R1)      ;ACCESS LOCATIONS
5716 034514 005761 000002              TST      2(R1)
5717 034520 000207              RETURN      ;NO TRAP - GOOD - RETURN
5718
5719 034522 104426      1$:      READCSR
5720 034524 104024              ERROR      +24
5721 034526              SET      HEADER
5722 034534 000207              RETURN
5723
5724              ;TRAP SHOULD OCCURE TEST
5725 034536 012737 034550 002372      MTPC25: MOV      #1$,PARTHERE ;SETUP TRAP DESTINATION
5726 034544 005711              TST      (R1)      ;ACCESS 1ST LOCATION
5727 034546 000405              BR      2$      ;NO TRAP - BAD NEWS - SKIP
5728 034550 012737 034574 002372      1$:      MOV      #3$,PARTHERE ;SETUP TRAP DESTINATION
5729 034556 005761 000002              TST      2(R1)      ;ACCESS 2ND LOCATION
5730 034562 104426      2$:      READCSR
5731 034564 104025              ERROR      +25
5732 034566              SET      HEADER
5733 034574 000207      3$:      RETURN
5734
  
```

5738 034576

```
MTPA26: SUBTST <<MTPA26 RANDOM DATA (WRITE)>>
:*****
:*SUBTEST MTPA26 RANDOM DATA (WRITE)
:*****
1$: JMP KDPAR0 ;V177640 GOTO V172360
MOV R2,(R1)+ ;V177644
MOV R3,(R1)+ ;V177646
SOB R0,1$ ;V177650
RETURN ;V177652
```

5739 034576 000137 172360
 5740 034602 010221
 5741 034604 010321
 5742 034606 077005
 5743 034610 000207
 5744
 5745 034612

```
MTPB26: SUBTST <<MTPB26 RANDOM DATA (READ)>>
:*****
:*SUBTEST MTPB26 RANDOM DATA (READ)
:*****
1$: .DSABL AMA
.ENABL LSB
JMP @#KDPAR0 ;V177640 GOTO V172360
CMP R2,(R1)+ ;V177644
BEQ 2$ ;V177646
PERR25 ;V177650
2$: COM (PC)+ ;V177652
RANODD: 0 ;V177654 FOR ERROR REPORTING
CMP R3,(R1)+ ;V177656
BEQ 3$ ;V177660
PERR25 ;V177662
3$: COM RANODD ;V177664
SOB R0,1$ ;V177670
RETURN ;V177672
.DSABL LSB
.ENABL AMA
```

5746
 5747
 5748 034612 000137 172360
 5749 034616 020221
 5750 034620 001401
 5751 034622 104451
 5752 034624 005127
 5753 034626 000000
 5754 034630 020321
 5755 034632 001401
 5756 034634 104451
 5757 034636 005167 177764
 5758 034642 077015
 5759 034644 000207
 5760
 5761
 5762
 5763 034646

```
MTPC26: SUBTST <<RANDOM NUMBER SUBPROGRAM>>
:*****
:*SUBTEST RANDOM NUMBER SUBPROGRAM
:*****
: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
BR -.116 ;V172376 GOTO V172260
```

5764
 5765
 5766
 5767
 5768
 5769 034646 073427 000007
 5770 034652 060305
 5771 034654 005504
 5772 034656 060204
 5773 034660 062705 001057
 5774 034664 000730
 5775
 5776 034666

```
MTPD26: SUBTST <<RANDOM NUMBER SUBSUBPROGRAM>>
:*****
:*SUBTEST RANDOM NUMBER SUBSUBPROGRAM
:*****
ADC R4 ;V172260
ADD #47401,R4 ;V172262
MOV R5,R3 ;V172266
MOV R4,R2 ;V172270
JMP UIPAR2 ;V172272 GOTO V177644
```

5777 034666 005504
 5778 034670 062704 047401
 5779 034674 010503
 5780 034676 010402
 5781 034700 000137 177644

5784 034704

MTP030: SUBTST <<MT0030 FLUSH OUT DBE'S>>
 :*****

:*SUBTEST MT0030 FLUSH OUT DBE'S
 :*****

5785 034704 011002
 5786 034706 010220
 5787 034710 077103
 5788 034712 000207
 5789
 5790 034714

1\$: MOV (R0),R2 :V177640
 MOV R2,(R0)+ :V177642
 SOB R1,1\$:V177644
 RETURN :V177646

MTP031: SUBTST <<MTP031 SOB-A-LONG TEST>>
 :*****

:*SUBTEST MTP031 SOB-A-LONG TEST
 :*****

5791
 5792 034714 000000
 5793 034716 077001
 5794 034720 005167 177772
 5795 034724 020167 177766
 5796 034730 001403
 5797 034732 104454
 5798 034734 010167 177756
 5799 034740 005167 177752
 5800 034744 010200
 5801
 5802 034746 010503
 5803 034750 005725
 5804 034752 010504
 5805 034754 020527 160000
 5806 034760 001001
 5807 034762 000207
 5808
 5809 034764 014344
 5810 034766 001376
 5811 034770 000752
 5812 000056
 5813

.DSABL AMA
 0 :MOVE TERMINATOR
 1\$: SOB R0,1\$:SOB TILL R0 UNDERFLOWS
 COM 1\$:WRITE COMPLEMENT OF SOB
 CMP R1,1\$:READ & CHECK FOR NOT 'SOB R0,DOT'
 BEQ 2\$:OK - SKIP
 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,#LAST+2 :DONE?
 BNE 3\$:NO - SKIP
 RETURN :YES
 3\$: MOV -(R3),-(R4)
 BNE 3\$
 BR 1\$
 SOBLENGTH=-MTP031
 .ENABL AMA

5816 034772

MTPA32: SUBTST <<MTPA32 SKEWED UP ADDRESS TEST (WRITE)>>

 :*SUBTEST MTPA32 SKEWED UP ADDRESS TEST (WRITE)

5817 034772 062704 000006
 5818 034776 042704 177771
 5819 035002 010301
 5820 035004 060401
 5821 035006 010011
 5822 035010 062703 000010
 5823 035014 077512
 5824 035016 000207
 5825
 5826 035020

```

1$:  ADD    #6,R4           ;V177640      ;ADD OFFSET TO OFFSET COUNTER
      BIC    #^C6,R4        ;V177644      ;MASK TO (0,2,4,6)
      MOV    R3,R1         ;V177650      ;PUT IN LSB'S OF ADDRESS
      ADD    R4,R1         ;V177652      ;PUT IN MSB'S OF ADDRESS
      MOV    R0,(R1)       ;V177654      ;WRITE DATA IN R0 INTO MEMORY
      ADD    #10,R3        ;V177656      ;BUMP ADDRESS MSB TO NEXT BLOCK OF 4 WORDS
      SOB   R5,1$         ;V177662      ;LOOP TILL DONE
      RETURN                ;V177664
    
```

MTPB32: SUBTST <<MTPB32 SKEWED UP ADDRESS TEST (READ)>>

 :*SUBTEST MTPB32 SKEWED UP ADDRESS TEST (READ)

5827 035020 062704 000006
 5828 035024 042704 177771
 5829 035030 010301
 5830 035032 060401
 5831 035034 011102
 5832 035036 020200
 5833 035040 001401
 5834 035042 104446
 5835 035044 062703 000010
 5836 035050 077515
 5837 035052 000207
 5838
 5839 035054

```

1$:  ADD    #6,R4           ;V177640      ;ADD OFFSET TO OFFSET COUNTER
      BIC    #^C6,R4        ;V177644      ;MASK TO (0,2,4,6)
      MOV    R3,R1         ;V177650      ;PUT IN LSB'S OF ADDRESS
      ADD    R4,R1         ;V177652      ;PUT IN MSB'S OF ADDRESS
      MOV    (R1),R2       ;V177654      ;READ DATA INTO R0
      CMP    R2,R0         ;V177656      ;IS IT CORRECT?
      BEQ   2$            ;V177660      ;YES - SKIP
      PERR22                ;V177662      ;NO - PRINT ERROR
2$:  ADD    #10,R3         ;V177664      ;BUMP ADDRESS MSB TO NEXT BLOCK OF 4 WORDS
      SOB   R5,1$         ;V177666      ;LOOP TILL DONE
      RETURN                ;V177670
    
```

MTP033: SUBTST <<MTP033 WRITE RECOVERY TEST>>

 :*SUBTEST MTP033 WRITE RECOVERY TEST

5840
 5841
 5842
 5843
 5844
 5845 035054 012401
 5846 035056 020102
 5847 035060 001401
 5848 035062 104430
 5849 035064 077305
 5850 035066 012703 020000
 5851 035072 012400
 5852 035074 020005
 5853 035076 001401
 5854 035100 104427
 5855 035102 077305
 5856 035104 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
      PERR02                ;V177646      ;NO - TAKE ERROR TRAP
2$:  SOB   R3,1$         ;V177650      ;LOOP FOR 1/2 BANK
      MOV    #SIZE/2,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
      PERR01                ;V177664      ;NO- TAKE ERROR TRAP
4$:  SOB   R3,3$         ;V177666      ;LOOP FOR 1/2 BANK
      RETURN
    
```

5859 035106

MTP034: SUBTST <<MTP034 BRANCH GOBBLE TEST>>
 :*****
 :*SUBTEST MTP034 BRANCH GOBBLE TEST
 :*****

5860
 5861 035106 000000
 5862 035110 000000
 5863 035112 000261
 5864 035114 105511
 5865 035116 100402
 5866 035120 105212
 5867 035122 000773

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

5868
 5869
 5870 035124 102401
 5871 035126 104461
 5872
 5873 035130 000242
 5874 035132 105212
 5875 035134 103402
 5876 035136 102001
 5877 035140 100401
 5878 035142 104461

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

5879
 5880
 5881
 5882 035144 010701
 5883 035146 162701 000036
 5884 035152 010102
 5885 035154 005202

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

5886
 5887
 5888 035156 010503
 5889 035160 005725
 5890 035162 010504
 5891

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

5892
 5893 035164 020527 160000
 5894 035170 001001
 5895 035172 000207

7\$: ;DONE?
 CMP R5,#LAST+2 ;DONE?
 BNE 6\$;NO - SKIP
 RETURN ;YES - RETURN

5896
 5897
 5898 035174 014344
 5899 035176 001376
 5900 035200 005011
 5901 035202 000743
 5902 000076
 5903

8\$: ;MOVE CODE 1 LOCATION
 MOV -(R3),-(R4)
 BNE 6\$
 CLR (R1) ;CLEAR TEST WORD 'BGTEST'
 BR BRGOBB ;RUN MOVED CODE AGAIN
 GBLENGTH=-MTP034
 .ENABL AMA

5905
5906
5907 035204

.SBIT 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
```

5908 035204 010004
5909 035206 010103
5910 035210 010205
5911 035212 000207
5912
5913 035214

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

5914 035214
5915 035216 005237 002734
5916 035222 042737 177774 002734
5917 035230 022737 000001 002734
5918 035236 001414
5919 035240 022737 000002 002734
5920 035246 001413
5921 035250 022737 000003 002734
5922 035256 001414
5923 035260 005000
5924 035262 013704 002726
5925 035266 000414
5926 035270
5927 035274 000411
5928 035276 012700 000401
5929 035302 013704 002726
5930 035306 000404
5931 035310 012700 000401
5932 035314 012704 000401
5933 035320 010037 026522
5934 035324 010037 026536
5935 035330 010037 026562
5936 035334 010037 026576
5937 035340
5938 035342 000207
5939
5940 035344

```
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
```

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

5941
5942 035344 104415
5943 035346 012700 060000
5944 035352 012701 040000
5945 035356
5946 035364 012737 000207 177644 WARN6:
5947 035372 004737 026042
5948 035376 104416
5949 035400 000207

```
:WRITES DATA FROM R2
SAVREG
MOV #FIRST,R0
MOV #SIZE,R1
BMOV MTP000
MOV #207,UIPAR2
CALL SUPD01
RESREG
RETURN
```

:WARNING PUTTING 'RETURN' INSTRUCTION AFTER WRITE

```

5953 035402          PCONFIG:SUBTST <<SUBR PRINT CONFIGURATION MAP>>
:*****
:*SUBTEST          SUBR PRINT CONFIGURATION MAP
:*****
5954 035402          PUSH TKVEC,TKVEC+2,R0
5955 035414 010637 035662 MOV SP,PCONFS ;SAVE LAST GOOD SP
5956 035420 012737 035630 000060 MOV #PCONF2,TKVEC
5957 035426 012737 000340 000062 MOV #340,TKVEC+2
5958 035434 017700 145334 MOV @STKB,R0 ;KILL ANY OLD INTERRUPT
5959 035440 042737 000200 177776 BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
5960 035446 052777 000100 145316 BIS #BIT6,@STKS ;ENABLE KEYBOARD INTERRUPTS
5961
5962 035454          TYPE MSG001
5963 035460          TYPE MSG002
5964 035464          TYPE MSG003
5965          ;IF FAT PAPER ON TERMINAL GOTO 1$
5966 035470          IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1
5967 035504 012700 000074 MOV #60.,R0
5968 035510 010004 MOV R0,R4
5969 035512          CLEAR R1,R3
5970 035516          TYPE MSG004
5971 035522 004737 035664 CALL TCONFIG ;GO TYPE CONFIGURATION (1ST HALF)
5972 035526          TYPE $CRLF
5973 035532          TYPE MSG017 ;PRINT SPACE(S)
5974 035536          TYPE MSG011
5975 035542          TYPE $CRLF
5976 035546          TYPE MSG017 ;PRINT SPACE(S)
5977 035552          TYPE MSG012
5978 035556 012701 000550 MOV #60.*2*3,R1
5979 035562 010103 MOV R1,R3
5980 035564 004737 035664 CALL TCONFIG
5981 035570 000417 BR PCONF2
5982
5983 035572 012700 000170 PCONF1: MOV #120.,R0
5984 035576 010004 MOV R0,R4
5985 035600          CLEAR R1,R3
5986 035604          TYPE MSG014 ;SPACE
5987 035610          TYPE MSG011
5988 035614          TYPE MSG004
5989 035620          TYPE MSG012
5990 035624 004737 035664 CALL TCONFIG
5991
5992 035630 013706 035662 PCONF2: MOV PCONFS,SP ;RESTORE STACK
5993 035634 042777 000100 145130 BIC #BIT6,@STKS
5994 035642 117700 145126 MOV @STKB,R0 ;READ CHAR TO KILL FLAG
5995 035646          POP RO,TKVEC+2,TKVEC
5996 035660 000207          RETURN
5997
5998 035662 000000          PCONFS: 0 ;STACK SAVED HERE!
    
```

6002 035664

SUBSTST <<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
*****
    
```

6003
6004
6005
6006
6007
6008
6009
6010
6011
6012
6013
6014
6015 035664
6016 035670 032761 004000 003016
6017 035676 001403
6018 035700
6019 035704 000402
6020 035706
6021 035712 062701 000006
6022 035716 077014
6023 035720 010400
6024 035722 010301
6025
6026
6027
6028
6029 035724
6030 035730 016105 003016
6031 035734 072527 177774
6032 035740 042705 177760
6033 035744 022705 000012
6034 035750 100003
6035 035752 062705 000067
6036 035756 000402
6037 035760 062705 000060
6038 035764 110537 112134
6039 035770
6040 035774 062701 000006
6041 036000 077025
6042 036002 010400
6043 036004 010301

```

*****
** ERROR **
*****
TCONFIG:TYPE      MSG005
1$:  BIT      #BIT11,CONFIG(R1)      ;ERROR ON THIS BANK?
      BEQ      2$                    ;NO - SKIP
      TYPE     MSG013                ;PRINT 'X'
      BR       3$
2$:  TYPE     MSG014                ;PRINT SPACE
3$:  ADD      #6,R1                  ;BUMP POINTER
      SOB     R0,1$                  ;LOOP TILL DONE
      MOV     R4,R0
      MOV     R3,R1

*****
** CPU'S **
*****
4$:  TYPE     MSG008
      MOV     CONFIG(R1),R5
      ASH    #-4,R5                  ;GET CPU BITS
      BIC    #^C17,R5                ;CLEAR NON INTERESTING BITS
      CMP    #10.,R5
      BPL    8$
      ADD    #67,R5                  ;MAKE NUMBERS OVER 9 HEX
      BR     9$
8$:  ADD     #60,R5                  ;MAKE ASCII
9$:  MOVB    R5,MSG015                ;PLUG INTO MEMORY
      TYPE     MSG015
      ADD    #6,R1                  ;BUMP POINTER
      SOB    R0,4$                  ;LOOP TILL DONE
      MOV    R4,R0
      MOV    R3,R1
    
```

```

6046
6047
6048
6049 036006
6050
6051 036012 016105 003016
6052 036016 072527 177763
6053 036022 042705 177774
6054 036026 005705
6055 036030 001004
6056 036032 112737 000077 112134
6057 036040 000435
6058 036042 016105 003016
6059 036046 042705 177770
6060 036052 022705 000001
6061 036056 001002
6062 036060 006305
6063 036062 000420
6064 036064 022705 000002
6065 036070 001002
6066 036072 006305
6067 036074 000413
6068 036076 022705 000004
6069 036102 001003
6070 036104 012705 000010
6071 036110 000405
6072 036112 005705
6073 036114 001403
6074 036116 012705 000077
6075 036122 000402
6076 036124 062705 000060
6077 036130 110537 112134
6078 036134
6079 036140
6080 036150 062701 000006
6081 036154 077062
6082 036156 010400
6083 036160 010301
6084
6085
6086
6087
6088
6089 036162
6090 036166 016105 003016
6091 036172 072527 177763
6092 036176 042705 177774
6093 036202
6094 036210 112737 000113 112134
6095 036216
6096 036220
6097 036230 112737 000112 112134
6098 036236
6099 036240 112737 000060 112134
6100 036246
6101 036246
6102 036246

:*****
:** INTERLEAVE **
:*****
TYPE MSG007
:THIS IS AN ENTRY POINT FROM ERROR REPORTS
TCFIG1: MOV CONFIG(R1),R5
ASH #-13.,R5 ;GET MEMORY TYPE
BIC #^C3,R5 ;CLEAR NON INTERESTING BITS
TST R5
BNE 1$
MOVB #'?,MSG015
BR 7$
1$: MOV CONFIG(R1),R5
BIC #^C7,R5 ;GET INTERLEAVE BITS
CMP #1,R5
BNE 2$
ASL R5 ;2 TO 1
BR 5$
2$: CMP #2,R5
BNE 3$
ASL R5 ;4 TO 1
BR 5$
3$: CMP #4,R5
BNE 4$
MOV #8.,R5 ;8 TO 1
BR 5$
4$: TST R5
BEQ 5$
MOV #'?,R5
BR 6$
5$: ADD #60,R5 ;MAKE ASCII
6$: MOVB R5,MSG015 ;PLUG INTO MEMORY
7$: TYPE MSG015
IF NOTAB NE #0 THEN $RETURN
ADD #6,R1 ;BUMP POINTER
SOB R0,TCFIG1 ;LOOP TILL DONE
MOV R4,R0
MOV R3,R1

:*****
:** MEMORY TYPE **
:*****
.ENABL LSB
TYPE MSG009
8$: MOV CONFIG(R1),R5
ASH #-13.,R5 ;GET MEMORY TYPE
BIC #^C3,R5 ;CLEAR NON INTERESTING BITS
IF R5 EQ #1
MOVB #'K,MSG015
ELSE
IF #SLAVE3!SLAVE2!SLAVE1!MASTER SET.IN CONFIG(R1)
MOVB #'J,MSG015
ELSE
MOVB #'0,MSG015
END ;OF IF #SLAVE3!SLAVE2!SLAVE1!MASTER
END ;OF IF R5
TYPE MSG015

```

```

6103 036252 062701 000006      ADD    #6,R1      ;BUMP POINTER
6104 036256 077035              SOB    R0,8$     ;LOOP TILL DONE
6105 036260 010400              MOV    R4,R0
6106 036262 010301              MOV    R3,R1
6107                          .DSABL LSB
6108
6109                          :*****
6110                          **: BOX **
6111                          :*****
6112 036264              TYPE    MSG016
6113 036270 016105 003016      9$:    MOV    CONFIG(R1),R5
6114 036274 032705 000010      BIT    #BIT3,R5   ;EXTERNAL INTERLEAVE?
6115 036300 001404              BEQ    10$        ;NO - SKIP
6116 036302 112737 000115 112134  MOVB   #'M,MSG015 ;INDICATE MULTIPLE BOXES
6117 036310 000417              BR     12$
6118 036312 032705 020014      10$:   BIT    #BIT13:14,R5 ;MJ11?
6119 036316 001004              BNE   11$        ;NO - SKIP
6120 036320 112737 000077 112134  MOVB   #'?,MSG015 ;I DON'T KNOW HOW MJ11'S INTERLEAVE
6121 036326 000410              BR     12$
6122 036330 072527 177770      11$:   ASH   #-8,,R5   ;GET BOX NUMBER
6123 036334 042705 177774      BIC   #^C3,R5    ;CLEAR NON INTERESTING BITS
6124 036340 062705 000060      ADD   #60,R5     ;MAKE ASCII
6125 036344 110537 112134      MOVB  R5,MSG015  ;PLUG INTO MEMORY
6126 036350              12$:   TYPE    MSG015
6127 036354 062701 000006      ADD    #6,R1      ;BUMP POINTER
6128 036360 077035              SOB    R0,9$     ;LOOP TILL DONE
6129 036362 010400              MOV    R4,R0
6130 036364 010301              MOV    R3,R1
6131
6132                          :*****
6133                          **: PROTECTED **
6134                          :*****
6135 036366              TYPE    MSG010
6136 036372 005761 003016      13$:   TST   CONFIG(R1) ;BANK PROTECTED?
6137 036376 100004              BPL   14$        ;NO - SKIP
6138 036400 112737 000120 112134  MOVB   #'P,MSG015
6139 036406 000407              BR     15$
6140 036410 032761 010000 003016  14$:   BIT    #BIT12,CONFIG(R1) ;PROTECTED REGION OF MK11 BOX?
6141 036416 001406              BEQ   16$        ;NO - SKIP
6142 036420 112737 000113 112134  MOVB   #'K,MSG015
6143 036426              15$:   TYPE    MSG015
6144 036432 000402              BR     17$
6145 036434              16$:   TYPE    MSG014
6146 036440 062701 000006      17$:   ADD    #6,R1      ;PRINT SPACE
6147 036444 077026              SOB    R0,13$    ;BUMP POINTER
6148 036446 010400              MOV    R4,R0     ;LOOP TILL DONE
6149 036450 010301              MOV    R3,R1
6150 036452 000207              RETURN

```

.SBTTL TRAP PARITY ERROR HANDLER

.....
:VECTOR TO HERE FROM TRAPS TO 114
:IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
:.....

CODE ACTION

0 PRINT UNEXPECTED PARITY TRAP
1 COUNT ERROR & SAVE 1ST IN 'PADDRESS'
2 SET 'ABORT' / SETUP 'BADPC' / RETURN VIA PCBUMP
3 RETURN VIA 'PARTHERE'
4 COUNT ERROR & SAVE 1ST IN 'PADDRESS' REGARDLESS OF ERROR REGISTER

6168	036454	022737	000001	002100	PARITY:	CMP	#1,NOPAR	:COUNTING PARITY ERRORS?
6169	036462	001023				BNE	3\$:NO - SKIP
6170	036464	032737	000001	177744		BIT	#BIT0, MEMERR	:MEMORY TIMEOUT?
6171	036472	001402				BEQ	1\$:NO - SKIP
6172	036474	000137	036646			JMP	NONEXIST	:YES - SIMULATE A TRAP TO 4
6173	036500	005237	002074		1\$:	INC	PARCNT	:PARITY ERROR COUNTER + 1
6174	036504	022737	000001	002074		CMP	#1,PARCNT	:FIRST PARITY ERROR?
6175	036512	001003				BNE	2\$:NO - SKIP
6176	036514	013737	177740	002042		MOV	LOADRS,PADDRESS	:SAVE 1ST BAD PARITY ADDRESS
6177	036522	013737	177744	177744	2\$:	MOV	MEMERR, MEMERR	:CLEAR ERROR BITS
6178	036530	000002				RTI		
6179	036532	022737	000002	002100	3\$:	CMP	#2,NOPAR	:ACTION CODE = 2 ?
6180	036540	001016				BNE	6\$:NO - SKIP
6181	036542					SET	ABORTFLAG	:YES
6182	036550	004737	036770			CALL	BADSTACK	:FIND BAD SP,PC,PSW OFF STACK
6183	036554	063716	002406			ADD	PCBUMP,(SP)	:UPDATE RETURN PC
6184	036560	013737	177744	177744		MOV	MEMERR, MEMERR	:CLEAR ERROR BITS
6185	036566	042766	000004	000002		BIC	#BIT2,2(SP)	:SHOW FAILURE BY .NE.
6186	036574	000002				RTI		
6187	036576	022737	000003	002100	6\$:	CMP	#3,NOPAR	:ACTION CODE = 3 ?
6188	036604	001006				BNE	7\$:NO - SKIP
6189	036606	013716	002372			MOV	PARTHERE,(SP)	
6190	036612	013737	177744	177744		MOV	MEMERR, MEMERR	:CLEAR ERROR BITS
6191	036620	000002				RTI		
6192	036622	022737	000004	002100	7\$:	CMP	#4,NOPAR	
6193	036630	001001				BNE	8\$	
6194	036632	000722				BR	1\$	
6195								
6196	036634	004737	036770		8\$:	CALL	BADSTACK	:FIND BAD SP,PC,PSW OFF STACK
6197	036640					FATAL	32	

```

6200          .SBTTL TRAP NON-EXISTANT MEMORY (HOLES) HANDLER
6201          :*****
6202          :VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
6203          :CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
6204          : 1) IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
6205          : 2) TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
6206          :*****
6207
6208 036646 022737 000001 002102 NONEXIST: CMP #1,NONEM ;COUNTING NON-EXISTANT MEMORY ERRORS?
6209 036654 001014          BNE 2$ ;NO - SKIP
6210 036656 005237 002072          INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
6211 036662 022737 000001 002072          CMP #1,NEMCNT ;FIRST ERROR?
6212 036670 001005          BNE 1$ ;NO - SKIP
6213 036672 010037 002040          MOV R0,ADDRESS ;ASSUME R0 CONTAINS THE ADDRESS ACCESSED
6214 036676 013737 177744 177744          MOV MEMERR, MEMERR ;CLEAR OUT HIADRS & LOADRS REG'S
6215 036704 000002          1$: RTI
6216 036706 005237 002072          2$: INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
6217 036712 012701 000001          MOV #1,R1 ;DUMMY UP R1 FOR A FORCED SOB EXIT
6218 036716 013737 177744 177744          MOV MEMERR, MEMERR ;CLEAR OUT HIADRS & LOADRS REG'S
6219 036724 000002          RTI
6220
6221          :*****
6222          .SBTTL TRAP TIMEOUT (TRAP TO 4) HANDLER
6223 036726 004737 036770 TIMEOUT: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6224 036732          FATAL 6
6225          :*****
6226          .SBTTL TRAP MEMORY MANAGEMENT (TRAP TO 250) HANDLER
6227 036740 004737 036770 MMTRAP: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6228 036744          FATAL 7
6229          .SBTTL TRAP RESERVED INSTRUCTION HANDLER
6230 036752 004737 036770 PDP1105: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6231 036756          FATAL 5
6232
6233 036764 IITRAP: SUBTST <<UNEXPECTED IIST TRAP HANDLER>>
6234          :*****
6235          :*SUBTEST UNEXPECTED IIST TRAP HANDLER
6236          :*****
6237          ERROR +57
6238          RTI
6239
6240          BADSTACK: SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
6241          :*****
6242          :*SUBTEST FIND BAD SP, PC, & PSW FROM STACK
6243          :*****
6244          MOV SP,BADSP
6245          ADD #2,BADSP
6246          MOV 2(SP),BADPC
6247          MOV 4(SP),BADPSW
6248          RETURN
  
```

```

6245          .SBTTL TRAP   KERNEL TRAP HANDLER
6246          :*****
6247          :KERNEL IS A TRAP THAT COMES HERE
6248          :*****
6249
6250 037020 042766 140000 000002 $KERNEL:   BIC   #140000,2(SP)
6251 037026 000002          RTI
6252          :*****
6253          .SBTTL TRAP   ENERGIZE TRAP HANDLER
6254 037030 052737 001001 177572 $ENERGIZE: BIS   #BIT9!BIT0,MMR0
6255 037036 000002          RTI
6256          :*****
6257          .SBTTL TRAP   DEENERGIZE TRAP HANDLER
6258 037040 042737 001001 177572 $DEENERGIZE: BIC #BIT9!BIT0,MMR0
6259 037046 000002          RTI
6260          :*****
6261          .SBTTL TRAP   CACHON TRAP HANDLER
6262 037050 013737 002700 177746 $CACHN:  MOV   CACHK,CONTRL ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
6263 037056 052737 000001 177746   BIS   #BIT0,CONTRL ;DISABLE TRAPS (BUT NOT ABORTS)
6264 037064 000002          RTI
6265          :*****
6266          .SBTTL TRAP   CACHOFF TRAP HANDLER
6267          :DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
6268 037066 052737 001415 177746 $CACHF:  BIS   #BIT0!BIT2!BIT3!BIT8!BIT9,CONTRL
6269 037074 000002          RTI

```

TRAP LOAD CSR TRAP HANDLER

6273
6274
6275 037076
6276 037100 012700 172100
6277 037104 063700 002256
6278 037110 005737 002254
6279 037114 001014
6280 037116 013720 002144
6281 037122
6282 037130 013710 002146
6283 037134
6284 037136 042710 100000
6285 037142
6286 037142
6287 037144 000002
6288 037146 042700 020000
6289 037152 013737 177746 002370
6290 037160 104424
6291 037162 013720 002144
6292 037166
6293 037174 013710 002146
6294 037200
6295 037202 042710 100000
6296 037206
6297 037206 013737 002370 177746
6298 037214
6299 037216 000002

```

.SBTTL TRAP LOAD CSR TRAP HANDLER
;LOAD CORRECT CSR WITH DATA IN CSR & CSR+2
$LOADC: PUSH RO
MOV #MK11ADD,RO ;CREATE CSR ADDRESS
ADD CSRNO,RO
TST CSRREL ;CSR'S RELOCATED?
BNE $LOAD1 ;YES - SKIP
MOV CSR,(RO)+ ;LOAD IT
IF TWOCSRS IS TRUE
MOV CSR+2,(RO)
ELSE
BIC #BIT15,(RO)
END ;OF IF TWOCSRS
POP RO
RTI
$LOAD1: BIC #20000,RO ;SAVE CACHE STATUS
MOV CONTRL,OLDCACHE ;TURN CACHE OFF
CACHOFF ;LOAD IT
MOV CSR,(RO)+
IF TWOCSRS IS TRUE
MOV CSR+2,(RO)
ELSE
BIC #BIT15,(RO)
END ;OF IF TWOCSRS
MOV OLDCACHE,CONTRL ;RESTORE CACHE
POP RO
RTI

```

```

6302          .SBTTL TRAP READ CSR TRAP HANDLER
6303          ;READ THE CORRECT CSR INTO LOCATIONS CSR & CSR+2
6304 037220   $READC: PUSH RO
6305 037222   012700 172100 MOV #MK11ADD,RO ;CREATE CSR ADDRESS
6306 037226   063700 002256 ADD CSRNO,RO
6307 037232   005737 002254 TST CSRREL ;CSR'S RELOCATED?
6308 037236   001005 BNE 1$ ;YES - SKIP
6309 037240   012037 002144 MOV (RO)+,CSR ;READ IT
6310 037244   011037 002146 MOV (RO),CSR+2 ;READ IT
6311 037250   000415 BR 2$
6312 037252   042700 020000 1$: BIC #20000,RO
6313 037256   013737 177746 002370 MOV CONTRL,OLDCACHE ;SAVE CACHE STATUS
6314 037264   104424 CACHOFF ;TURN CACHE OFF
6315 037266   012037 002144 MOV (RO)+,CSR ;READ IT
6316 037272   011037 002146 MOV (RO),CSR+2 ;READ IT
6317 037276   013737 002370 177746 MOV OLDCACHE,CONTRL ;RESTORE CACHE
6318 037304   037304 2$: POP RO
6319 037306   000002 RTI

6320
6321          .SBTTL TRAP TEST (R1) & READ CSR CAREFULLY
6322 037310   $TSTRD: PUSH RO
6323 037312   BMOV TSTRD1
6324 037320   012700 172100 MOV #MK11ADD,RO ;CREATE CSR ADDRESS
6325 037324   063700 002256 ADD CSRNO,RO
6326 037330   005737 002254 TST CSRREL ;CSR'S RELOCATED?
6327 037334   001003 BNE 1$ ;YES - SKIP
6328 037336   004737 177640 CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
6329 037342   000413 BR 2$
6330 037344   042700 020000 1$: BIC #20000,RO
6331 037350   013737 177746 002370 MOV CONTRL,OLDCACHE ;SAVE CACHE STATUS
6332 037356   104424 CACHOFF ;TURN CACHE OFF
6333 037360   004737 177640 CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
6334 037364   013737 002370 177746 MOV OLDCACHE,CONTRL ;RESTORE CACHE
6335          ;IF SINGLE BIT ERROR ONLY - SET CARRY BIT
6336 037372   2$: POP RO
6337 037374   IF #BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
6338 037414   052766 000001 000002 BIS #BIT0,2(SP)
6339 037422   ELSE
6340 037424   042766 000001 000002 BIS #BIT0,2(SP)
6341 037432   END ;OF IF #BIT4
6342 037432   000002 RTI
6343
6344 037434   005010 TSTRD1: CLR (RO) ;V177640
6345 037436   SUPERVISOR ;V177642 ;ENTER SUPERVISOR MODE
6346 037444   105711 TSTB (R1) ;V177646
6347 037446   042737 140000 177776 BIC #BIT15!BIT14,PSW ;V177650
6348 037454   012037 002144 MCV (RO)+,CSR ;V177656
6349 037460   011037 002146 MOV (RO),CSR+2 ;V177662
6350 037464   000207 RETURN ;V177666

```

```

6354 .SBTTL TRAP ECC DISABLE ALL CSR'S TRAP HANDLER
6355 037466 012737 000002 002144 $ECCDIS:MOV #BIT1,CSR
6356 037474 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6357 037510 005037 002146 CLR CSR+2
6358 037514 004737 040442 CALL CSROUT
6359 037520 000002 RTI
6360 .SBTTL TRAP ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
6361 037522 012737 000002 002144 $ECC1DIS:MOV #BIT1,CSR
6362 037530 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6363 037544 005037 002146 CLR CSR+2
6364 037550 104425 LOADCSR
6365 037552 000002 RTI
6366 .SBTTL TRAP MK11 INITIALIZE ALL CSR'S TRAP HANDLER
6367 037554 012737 000001 002144 $ECCINIT:MOV #BIT0,CSR
6368 037562 005037 002146 CLR CSR+2
6369 037566 004737 040442 CALL CSROUT
6370 037572 000002 RTI
6371 .SBTTL TRAP MK11 INITIALIZE 1 SELECTED CSR TRAP HANDLER
6372 037574 012737 000001 002144 $ECC1INIT:MOV #BIT0,CSR
6373 037602 005037 002146 CLR CSR+2
6374 037606 104425 LOADCSR
6375 037610 000002 RTI
6376 .SBTTL TRAP ENABLE SBE PARITY TRAPS ON ALL CSR'S
6377 037612 012737 000003 002144 $ENASBE:MOV #BIT0!BIT1,CSR
6378 037620 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6379 037634 005037 002146 CLR CSR+2
6380 037640 004737 040442 CALL CSROUT
6381 037644 000002 RTI
6382 .SBTTL TRAP ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
6383 037646 012737 000003 002144 $ENA1SBE:MOV #BIT0!BIT1,CSR
6384 037654 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6385 037670 005037 002146 CLR CSR+2
6386 037674 104425 LOADCSR
6387 037676 000002 RTI
6388 .SBTTL TRAP WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
6389 037700 013737 002404 002144 $CBCSR:MOV CHECK,CSR ;BITS 14-8
6390 037706 052737 000006 002144 BIS #BIT1!BIT2,CSR ;CHECK MODE
6391 037714 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6392 037730 005037 002146 CLR CSR+2
6393 037734 004737 040442 CALL CSROUT
6394 037740 000002 RTI
6395 .SBTTL TRAP WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
6396 037742 013737 002404 002144 $CB1CSR:MOV CHECK,CSR ;BITS 14-8
6397 037750 052737 000006 002144 BIS #BIT1!BIT2,CSR ;CHECK MODE
6398 037756 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6399 037772 005037 002146 CLR CSR+2
6400 037776 104425 LOADCSR
6401 040000 000002 RTI

```

6405					.SBTTL TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER
6406	040002				\$WASSBE: PUSH R1,R4
6407	040006	013701	002334		MOV MKCSRS,R1 ;GET CSR'S BYTE
6408	040012	005004			CLR R4
6409	040014				BEGIN LWSBE
6410	040014				FOR CSRNO := #0 TO #34 BY #4
6411	040020	106301			ASLB R1
6412	040022				ON.ERROR
6413	040024	104426			READCSR
6414	040026				IF #BIT4 SET.IN CSR
6415	040036				SET R4
6416	040042				LEAVE LWSBE
6417	040044				END ;OF IF #BIT4
6418	040044				END ;OF ON.ERROR
6419	040044				IFB R1 EQ #0 THEN LEAVE LWSBE
6420	040050				END ;OF FOR CSRNO
6421	040066				END LWSBE
6422	040066	006004			ROR R4 ;SET C BIT FOR ERROR
6423	040070				POP R4,R1
6424	040074				ON.ERROR
6425	040076	052766	000001	000002	BIS #BIT0,2(SP)
6426	040104				ELSE
6427	040106	042766	000001	000002	BIC #BIT0,2(SP)
6428	040114				END ;OF ON.ERROR
6429	040114	000002			RTI
6430					.SBTTL TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
6431					;ON RETURN IF CARRY IS SET THERE WAS A SBE
6432	040116	104426			\$WAS1SBE: READCSR
6433	040120	042766	000001	000002	BIC #BIT0,2(SP) ;CLR C BIT ON STACK
6434	040126	032737	000020	002144	BIT #BIT4,CSR
6435	040134	001403			BEQ 1\$
6436	040136	052766	000001	000002	BIS #BIT0,2(SP) ;SET C BIT ON STACK
6437	040144	000002			1\$: RTI

```

6440
6441 040146
6442 040152 013701 002334
6443 040156 005004
6444 040160
6445 040160
6446 040164 106301
6447 040166
6448 040170 104426
6449 040172
6450 040202
6451 040206
6452 040210
6453 040210
6454 040210
6455 040214
6456 040232
6457 040232 006004
6458 040234
6459 040240
6460 040242 052766 000001 000002
6461 040250
6462 040252 042766 000001 000002
6463 040260
6464 040260 000002
6465
6466
6467 040262 104426
6468 040264 005737 002144
6469 040270 100010
6470 040272 005737 002146
6471 040276 100401
6472 040300 104021
6473 040302 052766 000001 000002 2$:
6474 040310 000002
6475 040312 042766 000001 000002 3$:
6476 040320 000002

.SBTTL TRAP WAS THERE A DBE ON ANY CSR TRAP HANDLER
$WASDBE: PUSH R1,R4
MOV MKCSRS,R1 ;GET CSR'S BYTE
CLR R4
BEGIN LWDBE
FOR CSRNO := #0 TO #34 BY #4
ASLB R1
ON.ERROR
READCSR
IF #BIT15 SET.IN CSR
SET R4
LEAVE LWDBE
END ;OF IF #BIT15
END ;OF ON.ERROR
IFB R1 EQ #0 THEN LEAVE LWDBE
END ;OF FOR CSRNO
END LWDBE
ROR R4 ;SET C BIT FOR ERROR
POP R4,R1
ON.ERROR
BIS #BIT0,2(SP)
ELSE
BIC #BIT0,2(SP)
END ;OF ON.ERROR
RTI
.SBTTL TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
;ON RETURN IF CARRY IS SET THERE WAS A DBE
$WAS1DBE: READCSR
TST CSR ;DBE?
BPL 3$ ;NO - SKIP
TST CSR+2 ;DBE IN OTHER WORD?
BMI 2$ ;YES - SKIP
ERROR +21 ;NO - ONLY ONE FLAG POPPED UP!
BIS #BIT0,2(SP) ;SET C BIT ON STACK
RTI
BIC #BIT0,2(SP) ;CLR C BIT ON STACK
RTI
  
```

TRAP CLEAR ALL MK11 CSR'S TRAP HANDLER

```

6479          .SBTTL TRAP CLEAR ALL MK11 CSR'S TRAP HANDLER
6480 040322 $CLRCSR: CLEAR CSR,CSR+2
6481 040332 004737 040442 CALL CSROUT
6482 040336 000002 RTI
6483          .SBTTL TRAP CLEAR 1 SELECTED MK11 CSR TRAP HANDLER
6484 040340 $CLR1CSR: CLEAR CSR,CSR+2
6485 040350 104425 LOADCSR
6486 040352 000002 RTI
6487          .SBTTL TRAP ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
6488          :CHECKBITS ALREADY IN LOC "CSR"
6489 040354 052737 000006 002144 $CHKDIS: BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
6490 040362 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6491 040376 005037 002146 CLR CSR+2
6492 040402 004737 040442 CALL CSROUT
6493 040406 000002 RTI
6494          .SBTTL TRAP ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
6495          :CHECKBITS ALREADY IN LOC "CSR"
6496 040410 052737 000006 002144 $CHK1DIS: BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
6497 040416 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
6498 040432 005037 002146 CLR CSR+2
6499 040436 104425 LOADCSR
6500 040440 000002 RTI

```

6503 040442

```
CSROUT: SUBTST <<SUBR WRITE IN ALL CSR'S>>
:*****
:*SUBTEST SUBR WRITE IN ALL CSR'S
:*****
```

6504 040442
 6505 040444 013701 002334
 6506 040450
 6507 040450
 6508 040454 106301
 6509 040456
 6510 040460 104425
 6511 040462
 6512 040462
 6513 040466
 6514 040504
 6515 040504
 6516 040506 000207
 6517
 6518
 6519 040510
 6520 040514 013701 002106
 6521 040520 070127 000006
 6522 040524 042761 020000 003020
 6523 040532
 6524 040536 000002

```

PUSH R1
MOV MKCSRS,R1 ;GET CSR'S BYTE
BEGIN LCSROUT
FOR CSRNO := #0 TO #34 BY #4
ASLB R1
ON.ERROR
LOADCSR
END ;OF ON.ERROR
IFB R1 EQ #0 THEN LEAVE LCSROUT
END ;OF FOR CSRNO
END LCSROUT
POP R1
RETURN

.SBTTL TRAP INVALIDATE BACKGROUND PATTERN
$INVALID:PUSH R0,R1
MOV BANK,R1
MUL #6,R1
BIC #BIT13,CONFIG+2(R1)
POP R1,R0
RTI
```

6528 040540

CHKGEN: SUBTST<<SUBR GENERATE CHECK BITS>>

 :*SUBTEST SUBR GENERATE CHECK BITS

6529
 6530
 6531
 6532
 6533
 6534
 6535
 6536 040540
 6537 040554 012702 000077
 6538 040560 012703 040640
 6539 040564 013705 002402
 6540 040570 012501
 6541 040572 011500
 6542
 6543 040574 006704
 6544 040576 142304
 6545 040600 074402
 6546 040602 073027 000001
 6547 040606 001372
 6548
 6549 040610 042702 177600
 6550 040614 000302
 6551 040616 010237 002404
 6552 040622
 6553 040636 000207

```

:CHECK BIT GENERATOR ROUTINE
:CALLING SEQUENCE IS:
:      MOV      #WORD1, SOURCE ;SOURCE = ADDRESS OF DATA
:      CALL    CHKGEN
:
:CHECK BITS RETURNED IN BITS 14-8 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 14-8
MOV   R2,CHECK
POP   R5,R4,R3,R2,R1,R0
RETURN
  
```

Address	Offset	Value	CHKTAB	Bit
6556	040640		:BYTE #3	
6557	040640	200	.BYTE ^C177	:BIT 31
6558	040641	301	.BYTE ^C076	:BIT 30
6559	040642	302	.BYTE ^C075	:BIT 29
6560	040643	203	.BYTE ^C174	:BIT 28
6561	040644	304	.BYTE ^C073	:BIT 27
6562	040645	205	.BYTE ^C172	:BIT 26
6563	040646	206	.BYTE ^C171	:BIT 25
6564	040647	307	.BYTE ^C070	:BIT 24
6565			:BYTE #2	
6566	040650	310	.BYTE ^C067	:BIT 23
6567	040651	211	.BYTE ^C166	:BIT 22
6568	040652	212	.BYTE ^C165	:BIT 21
6569	040653	313	.BYTE ^C064	:BIT 20
6570	040654	214	.BYTE ^C163	:BIT 19
6571	040655	315	.BYTE ^C062	:BIT 18
6572	040656	316	.BYTE ^C061	:BIT 17
6573	040657	217	.BYTE ^C160	:BIT 16
6574			:BYTE #1	
6575	040660	320	.BYTE ^C057	:BIT 15
6576	040661	221	.BYTE ^C156	:BIT 14
6577	040662	222	.BYTE ^C155	:BIT 13
6578	040663	323	.BYTE ^C054	:BIT 12
6579	040664	224	.BYTE ^C153	:BIT 11
6580	040665	325	.BYTE ^C052	:BIT 10
6581	040666	326	.BYTE ^C051	:BIT 9
6582	040667	227	.BYTE ^C150	:BIT 8
6583			:BYTE #0	
6584	040670	340	.BYTE ^C037	:BIT 7
6585	040671	241	.BYTE ^C136	:BIT 6
6586	040672	242	.BYTE ^C135	:BIT 5
6587	040673	343	.BYTE ^C034	:BIT 4
6588	040674	244	.BYTE ^C133	:BIT 3
6589	040675	345	.BYTE ^C032	:BIT 2
6590	040676	346	.BYTE ^C031	:BIT 1
6591	040677	247	.BYTE ^C130	:BIT 0

6595 040700

SUBTST<<SUBR MAPPER>>

6596
 6597
 6598
 6599
 6600
 6601
 6602
 6603
 6604 040700
 6605 040712 012700 172340
 6606 040716 012701 172240
 6607 040722 012702 077406
 6608 040726 012704 172200
 6609 040732 012705 000010
 6610 040736 012021
 6611 040740 010224
 6612 040742 077503
 6613 040744 012741 177600
 6614
 6615
 6616 040750 022703 000170
 6617 040754 001412
 6618 040756 072327 000011
 6619
 6620 040762 012701 172246
 6621 040766 012702 000004
 6622 040772 010321
 6623 040774 062703 000200
 6624 041000 077204
 6625 041002
 6626 041014 000207
 6627
 6628 041016
 6629 041030 005000
 6630 041032 012701 172340
 6631 041036 012702 077406
 6632 041042 012703 172300
 6633 041046 012704 000010
 6634 041052 010021
 6635 041054 010223
 6636 041056 062700 000200
 6637 041062 077405
 6638 041064 012741 177600
 6639 041070 012741 177400
 6640 041074
 6641 041104 013700 002710
 6642 041110 004737 042370
 6643 041114
 6644 041114
 6645 041126 000002

```

*****
*SUBTEST      SUBR      MAPPER
*****
:THIS SUBROUTINE MAPS THE MEMORY BANK (16K WORDS = 1 BANK)
:IN R3 TO THE TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000 - 157777)).
:CALL  MOV      BANKNO,R3      ;SET UP BANK ARGUMENT
:      CALL     MAPPER         ;ACTUAL CALL
:      RETURN                    ;ONLY RETURN

:SET SUPERVISOR 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     #77406,R2       ;CONSTANT FOR 4K PAGE, UP, R/W
        MOV     #SIPDR0,R4      ;FIRST DISCRIPTOR REGISTER
        MOV     #8.,R5          ;COUNTER
1$:     MOV     (R0)+,(R1)+      ;PUT IN SUPERVISOR ADDRESS
        MOV     R2,(R4)+        ;PUT IN SUPERVISOR DISCRIPTOR
        SOB     R5,1$          ;LOOP TILL DONE
        MOV     #177600,-(R1)   ;CORRECT LAST FIELD FOR PERIPHERALS PAGE

:SET UP SUPERVISOR 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
        MOV     #4,R2           ;COUNTER
2$:     MOV     R3,(R1)+        ;PLUG IN PAR INFO
        ADD     #200,R3         ;BUMP ADDRESS 4K
        SOB     R2,2$          ;LOOP TILL DONE
3$:     POP     R5,R4,R2,R1,R0
        RETURN

$KMAP: .SBTTL TRAP MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
        PUSH   R0,R1,R2,R3,R4
        CLR    R0              ;1ST AREA TO MAP TO
        MOV    #KIPAR0,R1      ;FIRST ADDRESS
        MOV    #77406,R2       ;CONSTANT FOR 4K PAGE,UP,R/W
        MOV    #KIPDR0,R3      ;1ST PAGE DISCRIPTOR REGISTER
        MOV    #8.,R4          ;COUNTER
1$:     MOV    R0,(R1)+         ;PUT IN KERNEL ADDRESS
        MOV    R2,(R3)+         ;PUT IN KERNEL DISCRIPTOR
        ADD    #200,R0         ;ADD ADDRESS CONSTANT FOR 4K CHANGE
        SOB    R4,1$          ;LOOP TILL DONE
        MOV    #177600,-(R1)   ;THE PERIPHERALS PAGE TO KIPAR7
        MOV    #177400,-(R1)   ;AND NEXT LOWER PAGE TO KIPAR6
        IF CPUBIT NE #MASTER ;IF I'M NOT THE MASTER
        MOV    LOADHOME,R0     ;MAP TO LOADERS LIKE THE MASTER DID!
        CALL  NEWLOAD
        END ;OF IF CPUBIT
        POP   R4,R3,R2,R1,R0
        RTI
  
```

6649 041130

RELOCATE:SUBTST <<RELOCATE PROGRAM>>

 :SUBTEST RELOCATE PROGRAM

6650 041130
 6651 041144
 6652 041144
 6653 041152 004737 042422
 6654 041156
 6655 041200 013700 002106
 6656 041204 010037 002532
 6657 041210 013701 002710
 6658 041214 004737 042072
 6659 041220 004737 042370
 6660 041224 013701 002110
 6661 041230 052761 100000 003020
 6662 041236 042761 020000 003020
 6663 041244
 6664 041246
 6665 041246
 6666 041262
 6667 041272
 6668 041276
 6669 041276
 6670 041302
 6671 041302
 6672 041302
 6673 041306
 6674 041316
 6675 041326
 6676 041336
 6677 041346
 6678 041366
 6679
 6680 041406
 6681 041410
 6682 041410
 6683 041410
 6684 041410
 6685 041410
 6686 041410
 6687 041422
 6688 041432
 6689 041436
 6690 041436
 6691 041442
 6692 041442 042761 020000 003020
 6693 041450 005000
 6694 041452 071027 000006

```

IF #SW12 SET.IN @SWR THEN $RETURN ERROR
BEGIN LOADERBANK
FOR BANK := #1 TO #167
CALL EXBANK
IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE
MOV BANK,R0
MOV RO,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
FOR R1 := #3*2 TO #167*3*2 BY #6
IF #BIT11!BIT15 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 #BIT13!BIT14 OFF.IN CONFIG(R1) THEN LEAVE FINDBANK ;IF MJ11
IF #BIT13 SET.IN CONFIG(R1) AND #BIT14 OFF.IN CONFIG(R1) ;IF MK11
IF #BIT12 SET.IN CONFIG-6(R1) AND #BIT12 OFF.IN CONFIG(R1)
;IF 1ST UNPROTECTED MK11 BANK
LEAVE FINDBANK
END :OF IF #BIT12
END :OF IF #BIT13
END :OF IF CPUBIT
END :OF IF #BIT15
END :OF IF #BIT11
END :OF FOR
IF #SW13 OFF.IN @SWR
TYPE MSG075 ;RELOCATION NOT POSSIBLE
END :OF IF #SW13
$RETURN ERROR
END FINDBANK
BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
CLR RO
DIV #6,R0
  
```

```
6697 041456
6698 041462 004737 042240
6699 041466
6700 041474
6701 041506 104417
6702 041510 042737 000040 172516
6703 041516 013700 002400
6704 041522 006200
6705 041524
6706 041526 012737 100000 170200
6707 041534
6708 041534 010037 170202
6709 041540 004737 042026
6710 041544 052737 000040 172516
6711 041552 042737 001001 177572
6712 041560 004737 042322
6713 041564 052737 001001 177572
6714 041572
6715 041600
```

```
RELOC1: LET NEWBANK := R0
CALL USERMAP ;MAP NEWBANK TO USER PAR
USER ;ENTER USER MODE
BMOV 0,100000,SIZE ;MOVE PROGRAM
KERNEL ;ENTER KERNEL MODE
BIC #BIT5,MMR3 ;TURN OFF UNIBUS MAP
MOV NEWBANK,R0
ASR R0
ON.ERROR
MOV #BIT15,MAPLO
END ;OF ON.ERROR
MOV R0,MAPH0
CALL LOWMAP ;SETUP LOWER 16K IN UNIBUS MAP
BIS #BIT5,MMR3 ;ENERGIZE UNIBUS MAP
BIC #BIT9:BIT0,MMR0 ;DEENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
CALL NEWKERNEL
BIS #BIT9:BIT0,MMR0 ;ENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
SET RLFLAG
$RETURN NOERROR
```

6718 041604

UNRELOCATE:SUBTST <<UNRELOCATE PROGRAM>>

:SUBTEST UNRELOCATE PROGRAM

6719
6720 041604
6721 041606 013701 002532
6722 041612 013700 002710
6723 041616 004737 042072
6724 041622 004737 042370
6725 041626
6726 041632 013737 002532 002106
6727 041640 004737 042422
6728 041644 013701 002110
6729 041650 042761 100000 003020
6730 041656 013737 002710 002106
6731 041664 004737 042422
6732 041670 013701 002110
6733 041674 042761 020000 003020
6734 041702
6735
6736
6737 041706 042737 020000 003020
6738 041714
6739 041720 004737 042240
6740 041724
6741 041732
6742 041744 104417
6743 041746 042737 001001 177572
6744 041754 004737 042322
6745 041760 052737 001001 177572
6746 041766 005037 002134
6747 041772 042737 000040 172516
6748 042000
6749 042010 004737 042026
6750 042014 052737 000040 172516
6751 042022
6752 042024 000207
6753
6754 042026

:RESTORE LOADERS
PUSH R0
MOV LOADBANK,R1
MOV LOADHOME,R0
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

: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 #BIT9!BIT0,MMR0 ;DEENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
CALL NEWKERNEL
BIS #BIT9!BIT0,MMR0 ;ENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
CLR RLFLAG
BIC #BIT5,MMR3 ;TURN OFF UNIBUS MAP
CLEAR MAPLO,MAPHO
CALL LOWMAP ;SETUP LOWER 16K OF UNIBUS MAP
BIS #BIT5,MMR3 ;ENERGIZE UNIBUS MAP
POP R0
RETURN

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

:SUBTEST SETUP LOWER 16K OF UNIBUS MAP

6755 042026
6756 042034 012700 170200
6757 042040 012701 170204
6758 042044 012702 000003
6759 042050 012011
6760 042052 062721 020000
6761 042056 012021
6762 042060 077205
6763 042062
6764 042070 000207

PUSH R0,R1,R2
MOV #MAPLO,R0
MOV #MAPL1,R1
MOV #3,R2
1\$: MOV (R0)+,(R1)
ADD #BIT13,(R1)+
MOV (R0)+,(R1)+
SOB R2,1\$
POP R2,R1,R0
RETURN

6767 042072

BANKMOV:SUBTST <<MOVE BANKS>>

 :*SUBTEST MOVE BANKS

6768				:MOVE 3/4 OF A BANK	
6769				:CALLING SEQUENCE	
6770				:R0 = DESTINATION BANK	
6771				:R1 = SOURCE BANK	
6772	042072	104415		SAVREG	
6773	042074	004737	042240	CALL USERMAP	
6774	042100	104416		RESREG	
6775	042102	104415		SAVREG	
6776	042104	072027	000011	ASH #9,R0	
6777	042110	072127	000011	ASH #9,R1	
6778	042114	012702	177650	MOV #UIPAR4,R2	
6779	042120	012703	000200	MOV #200,R3	
6780					
6781	042124	010122		MOV R1,(R2)+	:MAP 1ST HALF BANK
6782	042126	060301		ADD R3,R1	:BUMP BY 4K
6783	042130	010122		MOV R1,(R2)+	
6784	042132	060301		ADD R3,R1	
6785					
6786	042134	010022		MOV R0,(R2)+	
6787	042136	060300		ADD R3,R0	
6788	042140	010022		MOV R0,(R2)+	
6789	042142	060300		ADD R3,R0	
6790					
6791	042144			USER	
6792	042152			BMOV 100000,140000,SIZE/2	:MOV 1ST HALF BANK
6793	042164	104417		KERNEL	:ENTER KERNEL MODE
6794					
6795	042166	012702	177650	MOV #UIPAR4,R2	
6796					
6797	042172	010122		MOV R1,(R2)+	:MAP 2ND HALF BANK
6798	042174	060301		ADD R3,R1	:BUMP BY 4K
6799	042176	010122		MOV R1,(R2)+	
6800	042200	060301		ADD R3,R1	
6801					
6802	042202	010022		MOV R0,(R2)+	
6803	042204	060300		ADD R3,R0	
6804	042206	010022		MOV R0,(R2)+	
6805	042210	060300		ADD R3,R0	
6806					
6807	042212			USER	
6808	042220			BMOV 100000,140000,SIZE/4	:MOV 3RD FOURTH OF BANK
6809	042232	104417		KERNEL	:ENTER KERNEL MODE
6810					
6811	042234	104416		RESREG	
6812	042236	000207		RETURN	

6815 042240

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

6816 042240 012701 177640
 6817 042244 012702 172340
 6818 042250 012703 177600
 6819 042254 012704 172300
 6820 042260 012705 000004
 6821 042264 012221
 6822 042266 011423
 6823 042270 077503
 6824
 6825 042272 013700 002400
 6826 042275 072027 000011
 6827
 6828 042302 012705 000004
 6829 042306 010021
 6830 042310 062700 000200
 6831 042314 011423
 6832 042316 077505
 6833 042320 000207
 6834
 6835 042322

```
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
:*****
```

6836 042322
 6837 042330 012700 172340
 6838 042334 013701 002400
 6839 042340 072127 000011
 6840
 6841 042344 012705 000004
 6842 042350 010120
 6843 042352 062701 000200
 6844 042356 077504
 6845 042360
 6846 042366 000207
 6847
 6848 042370

```
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
:*****
```

6849
 6850 042370
 6851 042374 012701 172350
 6852 042400 072027 000011
 6853 042404 010021
 6854 042406 062700 000200
 6855 042412 010021
 6856 042414
 6857 042420 000207

```
;R0 CONTAINS THE DESTINATION BANK
PUSH R0,R1
MOV #KIPAR4,R1
ASH #9.,R0 ;BANK 1 STARTS AT 1000000 LESS 6 LSB'S (1000)
MOV R0,(R1)+ ;SETUP KIPAR4
ADD #200,R0
MOV R0,(R1)+ ;SETUP KIPAR5
POP R1,R0
RETURN
```

6861 042422

EXBANK: SUBTST <<SUBR EXAMINE BANK>>

*SUBTEST SUBR EXAMINE BANK

6862
6863
6864
6865
6866
6867
6868
6869
6870
6871
6872
6873
6874
6875

:DOES THE FOLLOWING:
:(1) SETS UP 'BANKINDEX' AND R0 BASED ON VALUE OF 'BANK'
:(2) SETS THE 'MKFLAG' IF THE BANK IS AN MK11
:(3) SETS THE 'KPFLAG' IF THE BANK IS THE PROTECTED REGION OF MK11 MEMORY
:(4) SETS THE 'ACFLAG' IF THE BANK CAN BE ACCESSED BY THIS CPU
:(5) SETS THE 'PFLAG' IF THE BANK IS IN PROGRAM SPACE
:(6) 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)
: BUT! THE 'RRFLAG' IS ALWAYS SET TO DISABLE TESTING IF FIELD SERVICE MODE
: 'SELECTED BANKS' ARE BEING TESTED AND THIS BANK IS NOT SELECTED.
:(7) 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)

6876 042422
6877 042430
6878 042440
6879 042446
6880 042462 013701 002106
6881 042466 070127 000006
6882 042472 010137 002110
6883 042476 032761 010000 003016
6884 042504 001406
6885 042506
6886 042522 012700 000020
6887 042526
6888 042530
6889 042544 005037 002124
6890 042550
6891 042550 006300
6892 042552
6893 042562 016100 003016
6894 042566 042700 117777
6895 042572 022700 020000
6896 042576 001003
6897 042600
6898 042606 032761 100000 003016
6899 042614 001406
6900 042616
6901 042632 005737 002134
6902 042636 001402
6903 042640 005137 002132
6904 042644 032761 004000 003016
6905 042652 001403
6906 042654
6907 042662 005737 002712
6908 042666 001002
6909 042670 005137 002136
6910 042674
6911 042712
6912 042720
6913 042720
6914 042726 000207

PUSH R0,R1,R2
CLEAR MKFLAG,KPFLAG
SET ACFLAG
CLEAR PFLAG,RRFLAG,BMFLAG
MOV BANK,R1
MUL #6,R1 ;R0 = BANK * 6
MOV R1,BANKINDEX
BIT #BIT12,CONFIG(R1) ;PROTECTED REGION OF MK11 MEMORY?
BEQ 1\$;NO - SKIP
SET KPFLAG,RRFLAG
1\$: MOV #MASTER,R0
FOR R2 := #0 TO SLAVES
IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)
CLR ACFLAG
END ;OF IF R0
ASL R0
END ;OF FOR R2
2\$: MOV CONFIG(R1),R0
BIC #^C60000,R0 ;GET MEMORY TYPE BITS
CMP #20000,R0 ;IS IT AN MK11?
BNE 3\$;NO - SKIP
SET MKFLAG
3\$: BIT #BIT15,CONFIG(R1) ;BANK = PROGRAM SPACE?
BEQ 5\$;NO - SKIP
SET PFLAG,RRFLAG
5\$: TST RLFLAG ;IS PROGRAM RELOCATED?
BEQ 6\$;NO - SKIP
COM RRFLAG ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
6\$: BIT #BIT11,CONFIG(R1) ;ERRORS PRESENT IN THIS BANK?
BEQ 8\$;NO - SKIP
SET BMFLAG
8\$: TST WORST ;IS THIS A WORST FIRST PASS?
BNE 9\$;YES - SKIP
COM BMFLAG ;NO - COMPLEMENT BAD MEMORY FLAG
9\$: IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
SET RRFLAG
END ;OF IF SELONLY
POP R2,R1,R0
RETURN

6918 042730

6919
6920
6921
6922 042730 013700 002140
6923 042734 005100
6924 042736 013701 002126
6925 042742 074001
6926 042744 000207
6927
6928 042746
6929 042746

6930
6931
6932 042746 005237 002120
6933 042752 022737 000040 002120
6934 042760 000207
6935
6936 042762
6937 042762

6938 042762 012737 000037 002120
6939 042770 000207
6940
6941 042772

6942
6943 042772 005237 002106
6944 042776 022737 000170 002106
6945 043004 000207

```
BANKOK: SUBTST <<SUBR BANK OK?>>
:*****
:*SUBTEST SUBR BANK OK?
:*****
:TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
:IS OF THE TYPE WE ARE TESTING 'TMFLAG'.
:RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).
MOV TMFLAG,R0
COM R0
MOV MKFLAG,R1
XOR R0,R1
RETURN ;OK = (=OK)

INCRPT:
INCPAT: SUBTST <<SUBR INCREMENT PATTERN TESTING MK11'S>>
:*****
:*SUBTEST SUBR INCREMENT PATTERN TESTING MK11'S
:*****
:INCREMENT THE PATTERN & SET UP THE CONDITION CODES
:RESULT - Z BIT SET INDICATES OVERFLOW
INC PATTERN
CMP #40,PATTERN ;SET UP CONDITION CODES
RETURN ;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)

SETPAT:
HIPAT: SUBTST <<SUBR SET HIGHEST PATTERN TESTING TYPE>>
:*****
:*SUBTEST SUBR SET HIGHEST PATTERN TESTING TYPE
:*****
MOV #37,PATTERN ;SET HIGHEST PATTERN
RETURN

INCBNK: SUBTST <<SUBR INCREMENT BANK & TEST>>
:*****
:*SUBTEST SUBR INCREMENT BANK & TEST
:*****
:RESULTS RETURNED IN CONDITION CODES
INC BANK
CMP #120.,BANK ;TOO FAR?
RETURN
```

```

6948 043006      INCMAR: SUBTST <<SUBR INCREMENT MARGINS & TEST>>
:*****
:*SUBTEST      SUBR      INCREMENT MARGINS & TEST
:*****
6949              ;SKIPS THE SPECIAL CASE (MARGIN CANNOT = 1)
6950              ;RETURNS THE RESULT OF MARGIN OVERFLOW IN THE CONDITION CODES
6951              .ENABL  LSB
6952 043006      IF #SW11 OFF.IN @SWR AND QVFLAG IS FALSE
6953 043024      GOTO 1$
6954 043026      END ;OF IF #SW11
6955 043026      MOV      MARGIN,OLDMARGIN
6956 043034      CLR      MARGIN              ;YES - SET MARGIN TO ZERO
6957 043040      CALL    COREHISTORY
6958 043044      BR      3$
6959 043046      MOV      MARGIN,OLDMARGIN
6960 043054      INC      MARGIN
6961 043060      CMP      #1,MARGIN          ;SPECIAL CASE?
6962 043066      BNE     2$                  ;NO - SKIP
6963 043070      INC      MARGIN            ;YES - SET MARGIN TO '2'
6964 043074      2$:   IF MKFLAG IS FALSE AND MARGIN NE #6
6965 043112      GOTO 3$
6966 043114      END ;OF IF MKFLAG
6967 043114      IF MKFLAG IS TRUE AND MARGIN NE #4
6968 043132      GOTO 3$
6969 043134      END ;OF IF MKFLAG
6970 043134      CLR      MARGIN              ;YES - SET MARGIN TO ZERO
6971 043140      CALL    COREHISTORY
6972 043144      TST     MARGIN              ;SET CONDITION CODE - DONE IF Z BIT SET
6973 043150      RETURN
6974              .DSABL  LSB
6975
6976 043152      DECMAR: SUBTST <<DECREMENT MARGINS & TEST>>
:*****
:*SUBTEST      DECREMENT MARGINS & TEST
:*****
6977              ;SKIPS THE SPECIAL CASE (MARGIN CANNOT = 1)
6978              ;RETURNS THE RESULT OF MARGIN UNDERFLOW IN THE CONDITION CODES
6979 043152      MOV      MARGIN,OLDMARGIN
6980 043160      DEC      MARGIN
6981 043164      CMP      #1,MARGIN          ;SPECIAL CASE?
6982 043172      BNE     1$                  ;NO - SKIP
6983 043174      DEC      MARGIN            ;YES - SET MARGIN TO '0'
6984 043200      1$:   CALL    COREHISTORY
6985 043204      TST     MARGIN              ;SETUP CONDITION CODES - NEGATIVE IS UNDERFLOW
6986 043210      RETURN
  
```

6989 043212

```
SETMAR: SUBTST <<SUBR SET HIGHEST MARGIN>>
:*****
:*SUBTEST      SUBR      SET HIGHEST MARGIN
:*****
```

6990
 6991
 6992 043212
 6993 043230
 6994 043232
 6995 043232 013737 002112 002376
 6996 043240
 6997 043246 012737 000005 002112
 6998 043254
 6999 043256 012737 000003 002112
 7000 043264
 7001 043264 004737 043300
 7002 043270 000402
 7003 043272 005037 002112
 7004 043276 000207
 7005
 7006 043300

```
:RECOGNIZES THE 'INHIBIT MARGINS SWITCH'
.ENABL LSB
IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
GOTO 1$
END ;OF IF #SW11
MOV MARGIN,OLDMARGIN
IF MKFLAG IS FALSE
MOV #5,MARGIN ;HIGHEST MJ11 MARGIN
ELSE
MOV #3,MARGIN ;HIGHEST MK11 MARGIN
END ;OF IF MKFLAG
CALL COREHISTORY
BR 2$ ;SKIP
1$: CLR MARGIN ;SET MARGIN TO '0'
2$: RETURN
.DSABL LSB
```

```
COREHISTORY:SUBTST <<SUBR REWRITE CORE HISTORY>>
:*****
:*SUBTEST      SUBR      REWRITE CORE HISTORY
:*****
```

7007 043300
 7008 043314
 7009 043324
 7010 043364 013700 002112
 7011 043370 006300
 7012 043372 010037 177750
 7013
 7014 043376
 7015 043402 004737 035344
 7016 043406 104511
 7017 043410
 7018 043410
 7019 043410
 7020 043410 005037 002436
 7021 043414 000207

```
IF MKFLAG IS FALSE AND NULLFLAG IS FALSE
IF OLDMARGIN NE MARGIN
IF OLDMARGIN EQ #4 OR OLDMARGIN EQ #5 OR MARGIN EQ #4 OR MARGIN EQ #5
MOV MARGIN,RO
ASL RO
MOV RO,MAINT ;SET MARGINS

LET R2 := ONES
CALL BACKGND
INVALIDATE
END ;OF IF
END ;OF IF
END ;OF IF
CLR NULLFLAG
RETURN
```

7024 043416

BOOT: SUBTST <<BOOTSTRAP ROUTINE>>

: *SUBTEST BOOTSTRAP ROUTINE

7025
7026
7027
7028
7029
7030
7031
7032 043416 104472
7033 043420
7034 043426 005037 177750
7035 043432
7036 043444 004737 024754
7037 043450 104421
7038 043452 042737 000040 172516
7039 043460 005001
7040 043462 000005
7041 043464 012700 177406
7042 043470 010160 000004
7043 043474 012710 177400
7044 043500 012740 000005
7045 043504 105710
7046 043506 100376
7047 043510 062701 020000
7048 043514 005710
7049 043516 100761
7050 043520 005007

: INITIALIZE ALL CSR'S
: CLEAR ANY MARGINS
: UNRELOCATE IF NECESSARY
: FLUSH OUT ANY DBE'S
: TURN OFF MEMORY MANAGEMENT
: TURN OFF THE UNIBUS MAP
: BOOT RK0 OR RK1
ECCINIT ; TRAP ON DOUBLE BIT ERRORS (NORMAL)
SET4 #BOOT1 ; TRAPS TO 4 GOTO BOOT1
CLR MAINT
IF RLFLAG IS TRUE THEN \$CALL UNRELOCATE
CALL MT0030 ; FLUSH OUT DBE'S
DEENERGIZE ; TURN OFF MEMORY MANAGEMENT
BIC #BIT5,MMR3 ; TURN OFF THE UNIBUS MAP
BOOT1: CLR R1
1\$: RESET
MOV #177406,R0
MOV R1,4(R0)
MOV #177400,(R0)
MOV #5,-(R0)
2\$: TSTB (R0)
BPL 2\$
ADD #BIT13,R1
TST (R0)
BMI 1\$
CLR PC

7053 043522
 7054 043522 004737 043554
 7055 043526
 7056 043542 000777
 7057 043544
 7058 043546 000000
 7059 043550 000137 004352
 7060 043554
 7061
 7062 043554

```
EXIT:  SUBTST  <<HALT PROGRAM>>
:*****
:*SUBTEST    HALT PROGRAM
:*****
      CALL    SHUTUP
EXIT2:  IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
      BR      .
      ELSE
$EXHALT: HALT
      JMP     START
      END ;OF IF APTFLAG
```

7063
 7064
 7065
 7066
 7067
 7068
 7069
 7070
 7071 043554 104472
 7072 043556 005037 177750
 7073 043562
 7074 043574
 7075 043602 004737 024754
 7076 043606
 7077 043606 012700 000001
 7078 043612 013701 002710
 7079 043616 004737 042072
 7080 043622 104421
 7081 043624 042737 000040 172516
 7082 043632 005037 002670
 7083 043636 000207
 7084
 7085 043640

```
SHUTUP: SUBTST  <<SHUTDOWN DIAGNOSTIC>>
:*****
:*SUBTEST    SHUTDOWN DIAGNOSTIC
:*****
      :INITIALIZE ALL CSR'S
      :CLEAR MARGINS
      :UNRELOCATE
      :FLUSH OUT DBE'S
      :RESTORE LOADERS
      :TURN OFF MEMORY MANAGEMENT
      :UNMAP THE UNIBUS MAP
      :CLEAR MULTIPROCESSING LOCK
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
CLR     MAINT
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
BIC     #BIT5,MMR3 ;TURN OFF UNIBUS MAP
CLR     LOCK
RETURN
```

7086 043640
 7087 043654
 7088 043662 012737 043640 060024
 7089 043670 012737 000340 060026
 7090 043676 012737 000000 123640
 7091 043704 104417
 7092 043706 000000

```
APTDOWN: SUBTST  <<APT SHUTDOWN SEQUENCE>>
:*****
:*SUBTEST    APT SHUTDOWN SEQUENCE
:*****
      MAP     #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
      SUPERVISOR ;ENTER SUPERVISOR MODE
      MOV     #APTDOWN,FIRST+24
      MOV     #340,FIRST+26
      MOV     #0,FIRST+APTDOWN
      KERNEL ;ENTER KERNEL MODE
APTHLT: HALT
```

7095 043710

SUBSTST <<BLOCK MOVE SUBROUTINE>>

: *SUBTEST BLOCK MOVE SUBROUTINE

7096
7097
7098
7099
7100
7101
7102
7103
7104
7105
7106
7107
7108
7109
7110
7111
7112
7113
7114
7115
7116
7117
7118
7119
7120

043710
043716 012702 177640
043722 012701 000020
043726 000413
043730
043736 012701 000020
043742 000404
043744
043752 012501
043754 012502
043756 012500
043760 012022
043762 077102
043764
043772 000205

: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

7122
7123
7124 043774

7125 043774 104415
7126 043776
7127
7128 044002
7129 044016
7130 044022 104416
7131 044024 000207
7132 044026
7133 044026
7134 044042 104424
7135 044044
7136 044052
7137 044056 104414
7138 044060
7139 044062 020027 000024
7140 044066 101403
7141 044070
7142 044074 000766
7143 044076
7144 044106 044170
7145 044110 044256
7146 044112 044346
7147 044114 044520
7148 044116 044764
7149 044120 045274
7150 044122 046152
7151 044124 046160
7152 044126 046510
7153 044130 047104
7154 044132 047550
7155 044134 047734
7156 044136 050330
7157 044140 050356
7158 044142 050400
7159 044144 050420
7160 044146 050436
7161 044150 050454
7162 044152 050470
7163 044154 050502
7164 044156 050566
7165 044160
7166 044166 000731

```
.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
PUSH CONTRL,CSRNO,KAMIKAZE ;SAVE CACHE STATUS
CACHOFF ;TURN CACHE OFF
SET KAMIKAZE
FS1: TYPE MSG026 ;COMMAND:
RDDEC ;READ A DECIMAL NUMBER
POP RO ;COMMAND --> RO
CMP RO,#20.
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,MARGIN, & PATTERN
FSCMD6 ;TYPE CONFIGURATION MAP
FSCMD7 ;BATTERY BACKUP - P/F TEST
FSCMD8 ;SOB-A-LONG TEST
FSCMD9 ;SUPER TIGHT SCOPE LOOP
FCMD10 ;ERROR SUMMARY
FCMD11 ;REFRESH TEST
FCMD12 ;SET FILL COUNT
FCMD13 ;ENTER KAMIKAZE MODE
FCMD14 ;EXIT KAMIKAZE MODE
FCMD15 ;TURN CACHE OFF
FCMD16 ;TURN CACHE ON
FCMD17 ;RUN ONLY MULTIPOINT TESTS
FCMD18 ;RESUME RUNNING BOTH SINGLE & MULTIPOINT TESTS
FCMD19 ;TEST ONLY SELECTED BANKS
FCMD20 ;RESUME TESTING ALL BANKS
END ;OF CASE
BR FS1
```

7169 044170

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

7170 044170
 7171 044174 062706 000002
 7172 044200
 7173 044206 062706 000002
 7174 044212 005037 002012
 7175 044216
 7176 044220
 7177 044224
 7178 044224
 7179 044230
 7180 044240 062706 000002
 7181 044244
 7182 044246
 7183 044252
 7184 044252 104416
 7185 044254 000207
 7186
 7187 044256

```
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
IF CACHK EQ #BIT0!BIT2!BIT3!BIT8!BIT9 ;IF CACHE IS OFF
  ADD #2,SP ;THROW AWAY CACHE STATUS
ELSE
  POP CONTRL ;RESTORE CACHE STATUS
END :OF IF CACHK
RESREG
RETURN
```

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

7188 044256 004737 050630
 7189 044262 010637 002374
 7190 044266
 7191 044274 104426
 7192 044276
 7193 044304 104026
 7194 044306
 7195 044320 000207
 7196 044322
 7197 044326 013706 002374
 7198 044332
 7199 044344 000207

```
CALL WHICHCSR
MOV SP,FSSTACK
SET4 #1$ ;TRAPS TO 4 GOTO 1$
READCSR
SET NOERROR
ERROR +26 ;USE ERROR ROUTINE FOR PRINTOUT
RES4 ;RESET TRAPS TO 4 TO DEFAULT
RETURN
1$: TYPE MSG025 ;THIS CSR DOES NOT EXIST
MOV FSSTACK,SP
RES4 ;RESET TRAPS TO 4 TO DEFAULT
RETURN
```

```

7202 044346
7203 044346 004737 050630
7204 044352 010637 002374
7205 044356
7206 044364 104426
7207 044366
7208 044372
7209 044400 104026
7210 044402
7211 044414
7212 044420 104413
7213 044422
7214 044426
7215 044432 104413
7216 044434
7217 044440
7218 044446 104425
7219 044450 005037 002600
7220 044454 104426
7221 044456
7222 044462
7223 044470 104026
7224 044472 000207
7225 044474
7226 044500 013706 002374
7227 044504
7228 044516 000207
  
```

```

FSCMD2: SUBTST <<FS COMMAND 2 LOAD CSR>>
:*****
:*SUBTEST FS COMMAND 2 LOAD CSR
:*****
CALL WHICHCSR
MOV SP,FSSTACK
SET4 #1$ ;TRAPS TO 4 GOTO 1$
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"
TYPE MSG024 ;2ND CSR WORD
RDOCT ;READ AN OCTAL NUMBER
POP CSR+2 ;PUT IN IN LOC "CSR+2"
SET TWOCSRS
LOADCSR
CLR TWOCSRS
READCSR
TYPE MSG028
SET NOERROR
ERROR +26 ;USE FOR PRINTOUT - NOT AN ERROR
RETURN
1$: TYPE MSG025 ;THIS CSR DOES NOT EXIST
MOV FSSTACK,SP ;RESET TRAPS TO 4 TO DEFAULT
RES4
RETURN
  
```

7231 044520

FSCMD3: SUBTST <<FS COMMAND 3 EXAMINE MEMORY>>
 :*****
 :*SUBTEST FS COMMAND 3 EXAMINE MEMORY
 :*****

7232 044520
 7233 044540 012737 000002 002100
 7234 044546
 7235 044552
 7236 044556 104413
 7237 044560 013737 074434 002106
 7238 044566
 7239 044570 000241
 7240 044572 006100
 7241 044574 006137 002106
 7242 044600 000241
 7243 044602 006000
 7244 044604
 7245 044614 062700 060000
 7246 044620
 7247 044626
 7248 044634 012737 044706 002372
 7249 044642
 7250 044650
 7251 044664
 7252 044672 011001
 7253 044674 104417
 7254 044676
 7255 044704 000410
 7256
 7257 044706
 7258 044712 000405
 7259
 7260 044714 062706 000004
 7261 044720
 7262 044724 000400
 7263
 7264 044726 104417
 7265 044730
 7266 044750
 7267 044762 000207

PUSH BANK,NOPAR,PARHERE,4
 MOV #2,NOPAR ;INDICATE PARITY ACTION
 TYPE MSG029 ;EXAMINE MEMORY
 1\$: TYPE MSG031 ;PHYSICAL ADDRESS (0-1677776)??
 RDOCT ;READ OCTAL NUMBER ONTO STACK & \$HIOCT
 MOV \$HIOCT,BANK ;PUT MSB'S IN BANK
 POP R0 ;PUT LSB'S IN R0
 CLC
 ROL R0
 ROL BANK
 CLC
 ROR R0
 IF BANK GT #167 THEN GOTO 1\$;CHECK FOR BANK TOO HIGH
 ADD #FIRST,R0
 IF #BIT0 SET.IN R0 THEN GOTO 1\$;CHECK FOR ODD ADDRESS
 IF R0 HI #LAST THEN GOTO 1\$;CHECK FOR ADDRESS OVER 16K
 MOV #3\$,PARHERE ;INCASE OF ABORTS
 SET4 #4\$;TRAPS TO 4 GOTO 4\$
 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 SUPERVISOR ;ENTER SUPERVISOR MODE
 MOV (R0),R1
 KERNEL ;ENTER KERNEL MODE
 TYPOCS R1
 BR EXCMD3
 3\$: TYPE MSG032 ;PARITY ABORT
 BR EXCMD3
 4\$: ADD #4,SP ;FIX STACK
 TYPE MSG033 ;TIMEOUT TRAP
 BR EXCMD3
 EXCMD3: KERNEL ;ENTER KERNEL MODE
 POP 4,PARHERE,NOPAR,BANK
 RES4 ;RESET TRAPS TO 4 TO DEFAULT
 RETURN

```

7270 044764      FSCMD4: SUBTST <<FS  COMMAND 4  MODIFY MEMORY>>
:*****
:*SUBTEST      FS  COMMAND 4  MODIFY MEMORY
:*****
7271 044764      PUSH  BANK,NOPAR,PARHERE,4
7272 045004 012737 000003 002100      MOV   #3,NOPAR ;INDICATE PARITY ACTION
7273 045012      TYPE  MSG036 ;MODIFY MEMORY
7274 045016 1$:      TYPE  MSG031 ;PHYSICAL ADDRESS (0-1677776)??
7275 045022 104413      RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
7276 045024 013737 074434 002106      MOV   $HIOCT,BANK ;PUT MSB'S IN BANK
7277 045032      POP   RO ;PUT LSB'S IN RO
7278 045034 000241      CLC
7279 045036 006100      ROL   RO
7280 045040 006137 002106      ROL   BANK
7281 045044 000241      CLC
7282 045046 006000      ROR   RO
7283 045050      IF BANK GT #167 THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
7284 045060 062700 060000      ADD   #FIRST,RO
7285 045064      IF #BIT0 SET.IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
7286 045072      IF RO HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER 16K
7287 045100 012737 045146 002372      MOV   #3$,PARHERE ;INCASE OF ABORTS
7288 045106      SET4  #4$ ;TRAPS TO 4 GOTO 4$
7289 045114      MAP   BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7290 045130 104511      INVALIDATE
7291 045132      SUPERVISOR ;ENTER SUPERVISOR MODE
7292 045140 011001      MOV   (RO),R1
7293      ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
7294 045142 104417      KERNEL ;ENTER KERNEL MODE
7295 045144 000410      BR    5$
7296
7297 045146 3$:      TYPE  MSG032 ;PARITY ABORT
7298 045152 000431      BR    EXCMD4 ;EXIT
7299
7300 045154 062706 000004      4$:      ADD   #4,SP ;FIX STACK
7301 045160      TYPE  MSG033 ;TIMEOUT TRAP
7302 045164 000424      BR    EXCMD4 ;EXIT
7303
7304 045166 5$:      TYPE  MSG037 ;OLD DATA WAS
7305 045172      TYPOCS R1 ;PRINT IT
7306 045200      TYPE  MSG039 ;INPUT NEW DATA
7307 045204 104413      RDOCT ;READ ON OCTAL NUMBER ONTO THE STACK
7308 045206      POP   R1 ;GET NEW NUMBER
7309 045210      SUPERVISOR ;ENTER SUPERVISOR MODE
7310 045216 010110      MOV   R1,(RO) ;PUT IT IN MEMORY
7311 045220 011001      MOV   (RO),R1 ;READ IT AGAIN
7312 045222 104417      KERNEL ;ENTER KERNEL MODE
7313 045224      TYPE  MSG038 ;DATA IS NOW
7314 045230      TYPOCS R1 ;PRINT IT
7315
7316 045236 104417      EXCMD4: KERNEL ;ENTER KERNEL MODE
7317 045240      POP   4,PARHERE,NOPAR,BANK
7318 045260      RES4 ;RESET TRAPS TO 4 TO DEFAULT
7319 045272 000207      RETURN
  
```

```

7322 045274 FSCMD5: SUBTST <<FS COMMAND 5 SELECT BANK,MARGIN, & PATTERN>>
:*****
:*SUBTEST FS COMMAND 5 SELECT BANK,MARGIN, & PATTERN
:*****
7323 045274 PUSH BANK,PATTERN,MARGIN,CSRFIRST,PCBUMP,TKVEC,TKVEC+2
7324 045330 013737 002112 002376 MOV MARGIN,OLDMARGIN
7325 045336 010637 002374 MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
7326 045342 TYPE MSG040 ;SELECT BANK, MARGIN, & PATTERN TEST
7327 045346 1$: TYPE MSG030 ;BANK(0-167)?
7328 045352 104413 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7329 045354 POP BANK ;PUT IT IN BANK
7330 045360 IF BANK GT #167 THEN GOTO 1$ ;CHECK FOR BANK TO HIGH
7331
7332 045370 013701 002106 MOV BANK,R1
7333 045374 070127 000006 MUL #6,R1
7334 045400 IF CPUBIT OFF.IN CONFIG(R1)
7335 045410 TYPE MSG041 ;BANK NOT ACCESSABLE
7336 045414 GOTO 1$
7337 045416 END ;OF IF
7338
7339 045416 2$: TYPE MSG042 ;PATTERN(0-37)?
7340 045422 104413 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7341 045424 POP PATTERN ;PUT IT IN PATTERN
7342 045430 IF PATTERN GT #37 THEN GOTO 2$ ;CHECK FOR PATTERN TO HIGH
7343 045440 IF PATTERN EQ #0
7344 045446 TYPE MSG043 ;PATTERN 0 DATA IS?
7345 045452 104413 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7346 045454 POP R2 ;PUT IT IN R2
7347 045456 END ;OF IF
7348
7349 045456 CMD5A: TYPE MSG044 ;MARGIN(0,2,3,4,5)?
7350 045462 104413 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7351 045464 POP MARGIN ;PUT IN IN MARGIN
7352 045470 IF MARGIN GT #5 OR MARGIN EQ #1
7353 045510 GOTO CMD5A
7354 045512 END ;OF IF
7355 045512 IF MARGIN NE #0 AND #SW11 SET.IN @SWR
7356 045530 TYPE MSG045 ;MARGINS INHIBITED BY SW11
7357 045534 GOTO CMD5A
7358 045536 END ;OF IF MARGIN
7359
7360 045536 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7361 045552 104511 INVALIDATE
7362 045554 004737 042422 CALL EXBANK ;SET NEW MARGINS
7363 045560 IF RRFLAG IS TRUE
7364 045566 TYPE MSG049 ;BANK REQUIRES RELOCATION
7365 045572 GOTO CMD5C
7366 045574 END ;OF IF RRFLAG
7367 045574 004737 043300 CALL COREHISTORY
7368 045600 TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
7369 045604 012737 046052 000060 MOV #CMD5C,TKVEC
7370 045612 012737 000340 000062 MOV #340,TKVEC+2
7371 045620 017700 135150 MOV @TKB,R0 ;KILL ANY OLD INTERRUPT
7372 045624 042737 000200 177776 BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
7373 045632 052777 000100 135132 BIS #BIT6,@TKS ;ENABLE KEYBOARD INTERRUPTS
7374
7375 045640 013700 002112 MOV MARGIN,R0 ;SET MARGINS
  
```

```

7376 045644 006300 ASL R0 ;SET MARGINS
7377
7378 045646 010037 177750 MOV R0,MAINT
7379
7380 045652 SET HEADER,MUT
7381 045666 005037 002362 CMD5B: CLR PASFLG
7382 045672 012737 060000 002336 MOV #FIRST,CSRFIRST
7383 045700 IF #SWO SET.IN @SWR
7384 045710 104470 ECCDIS ;DISABLE ERROR CORRECTION
7385 045712 ELSE
7386 045714 104502 CLRCSR ;CLEAR MK11 CSRS
7387 045716 END ;OF IF
7388 045716 012737 000002 002100 MOV #2,NOPAR ;PARITY ACTION
7389 045724 012737 000002 002406 MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
7390 045732 013700 002120 MOV PATTERN,R0
7391 045736 006300 ASL R0
7392 045740 004770 045752 CALL @FSPAT(R0)
7393 045744 005037 002100 CLR NOPAR
7394 045750 000746 BR CMD5B ;LOOP TILL KEYBOARD INTERRUPT
7395
7396 045752 021704 FSPAT: MT0000 ;<1 SEC DATA PATTERN TEST
7397 045754 021740 MT0001 ;<1 SEC ADDRESS TEST
7398 045756 022000 MT0002 ;<1 SEC COMPLEMENT ADDRESS TEST
7399 045760 022050 MT0003 ; 1 SEC 3 XOR 9 WORST CASE NOISE TEST
7400 045762 022216 MT0004 ; 1 SEC ROTATING ZEROS TEST
7401 045764 022300 MT0005 ; 1 SEC ROTATING ONES TEST
7402 045766 022360 MT0006 ;<1 SEC INITIAL DATA TEST
7403 045770 022414 MT0007 ;<1 SEC ADDRESS BIT TEST
7404 045772 022456 MT0010 ;<1 SEC BYTE ADDRESSING TEST
7405 045774 022512 MT0011 ;<2 SEC CREATE SINGLE BIT ERROR TEST
7406 045776 022570 MT0012 ;<1 SEC WRITE BYTE CLEARS SBE TEST
7407 046000 022646 MT0013 ; 1 SEC CREATE DOUBLE BIT ERROR TEST
7408 046002 022722 MT0014 ; 1 SEC WRITE INHIBIT DURING DATIP WITH DBE
7409 046004 023000 MT0015 ; 1 SEC WRITE INHIBIT OF BYTE WITH DBE
7410 046006 023056 MT0016 ;<1 SEC WRITE INHIBIT OF WORD WITH DBE
7411 046010 023134 MT0017 ;<1 SEC HOLDING 1'S & 0'S TEST
7412 046012 023156 MT0020 ;<1 SEC MARCHING 1'S & 0'S IN CHECK BITS
7413 046014 023612 MT0021 ; 1 SEC MARCHING 0'S & 1'S TEST
7414 046016 023734 MT0022 ;10 SEC REFRESH & SHIFTING DIAGONAL TEST
7415 046020 023766 MT0023 ;10 SEC SHIFTING DIAGONAL TEST
7416 046022 024032 MT0024 ;20 SEC FAST GALLOPING PATTERN TEST
7417 046024 024172 MT0025 ;<1 SEC INTERRUPT ENABLE TEST
7418 046026 024240 MT0026 ;<1 SEC RANDOM DATA TEST
7419 046030 024370 MT0027 ; 1 SEC UNIQUE BANK TEST
7420 046032 024754 MT0030 ; 1 SEC FLUSH OUT DBE'S TEST
7421 046034 025214 MT0031 ; 3 SEC SOB-A-LONG TEST
7422 046036 025772 MT0999:MT0032
7423 046040 025340 MT0033 ;<1 SEC WRITE RECOVERY TEST
7424 046042 025530 MT0034 ;35 SEC BRANCH GOBBLE TEST
7425 046044 025772 MT0999:MT0035
7426 046046 025772 MT0999:MT0036
7427 046050 025772 MT0999:MT0037
7428
7429 046052 013706 002374 CMD5C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
7430 046056 042777 000100 134706 BIC #BIT6,@$TKS
7431 046064 POP TKVEC+2,TKVEC
7432 046074 117700 134674 MOV @TKB,R0 ;GET CHARACTER TO GET RID OF FLAG
  
```

```
7433 046100 POP PCBUMP, CSRFIRST, MARGIN
7434 046114 004737 043300 CALL COREHISTORY ;RESTORE OLD MARGINS
7435 046120 POP PATTERN, BANK
7436 046130 MAP BANK ;REMAP OLD BANK
7437 046144 004737 042422 CALL EXBANK
7438 046150 000207 RETURN
```

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

```
7441 046152 004737 035402 CALL PCONFIG
7442 046156 000207 RETURN
7443
```

```

7447 046160 FSCMD7: SUBTST <<FS COMMAND 7 BATTERY BACKUP TEST>>
:*****
:*SUBTEST FS COMMAND 7 BATTERY BACKUP TEST
:*****
7448 046160 PUSH BANK,PATTERN,MARGIN,CSRNO,TKVEC,TKVEC+2
7449 046210 SET FS7FLAG
7450 046216 IF #SWO SET.IN @SWR
7451 046226 104470 ECCDIS ;DISABLE ERROR CORRECTION
7452 046230 ELSE
7453 046232 104502 CLRCSR ;CLEAR MK11 CSRS
7454 046234 END ;OF IF
7455 046234 TYPE MSG050 ;BATTERY BACKUP TEST
7456 ;WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND
7457 046240 004737 024370 CALL MT0027 ;CREATE & CHECK BACKGROUND
7458 046244 012737 000012 002434 MOV #1^,TIME
7459 046252 TYPE MSG052 ;REMOVE SYSTEM POWER FOR
7460 046256 TYPDEC TIME
7461 046264 TYPE MSG053 ;SECONDS MAX!
7462 046270 TYPE MSG046 ;TO ESCAPE TYPE ANY KEY
7463 046274 012737 046336 000060 MOV #CMD7A,TKVEC
7464 046302 012737 000340 000062 MOV #340,TKVEC+2
7465 046310 017700 134460 MOV @STKB,R0 ;KILL ANY OLD INTERRUPT
7466 046314 042737 000200 177776 BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
7467 046322 052777 000100 134442 BIS #BIT6,@STKS ;ENABLE KEYBOARD INTERRUPTS
7468 046330 010637 002374 MOV SP,FSSTACK
7469
7470 046334 000001 WAIT
7471
7472 046336 013706 002374 CMD7A: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
7473 046342 042777 000100 134422 BIC #BIT6,@STKS
7474 046350 POP TKVEC+2,TKVEC
7475 046360 117700 134410 MOVB @STKB,R0 ;GET CHARACTER TO GET RID OF FLAG
7476 046364 TYPE MSG054 ;NOW STARTING READ TEST OF MEMORY BANKS
7477 046370 FOR BANK := #0 TO #167
7478 046374 004737 042422 CALL EXBANK
7479 046400 IF ACFLAG IS TRUE AND RRFLAG IS FALSE
7480 046414 104511 INVALIDATE
7481 046416 LET R2 := BANK
7482 046422 012700 060000 MOV #FIRST,R0
7483 046426 010004 MOV R0,R4
7484 046430 012701 040000 MOV #SIZE,R1
7485 046434 010103 MOV R1,R3
7486 046436 004737 026220 CALL SUPD03
7487 046442 END ;OF IF ACFLAG
7488 046442 END ;OF FOR BANK
7489 046456 TYPE MSG047 ;TEST COMPLETE
7490 046462 005037 002546 CLR FS7FLAG
7491 046466 POP CSRNO,MARGIN,PATTERN,BANK
7492 046506 000207 RETURN
7493
  
```

```

7496 046510          FSCMD8: SUBTST <<FS  COMMAND 8  SOB-A-LONG TEST>>
:*****
:*SUBTEST          FS  COMMAND 8  SOB-A-LONG TEST
:*****
7497 046510          PUSH  BANK,PATTERN,MARGIN,TKVEC,TKVEC+2,NOPAR
7498 046540 010637 002374  MOV   SP,FSSTACK          ;SAVE LAST GOOD STACK POINTER
7499 046544          TYPE  MSG055          ;SOB-A-LONG TEST
7500 046550          CMD8A: TYPE  MSG044          ;MARGIN(0,2,3,4,5)?
7501 046554 104413  RDOCT          ;READ AN OCTAL NUMBER ONTO THE STACK
7502 046556          POP   MARGIN          ;PUT IN IN MARGIN
7503 046562          IF MARGIN GT #5 OR MARGIN EQ #1
7504 046602          GOTO  CMD8A
7505 046604          END ;OF IF
7506 046604          IF MARGIN NE #0 AND #SW11 SET.IN @SWR
7507 046622          TYPE  MSG045          ;MARGINS INHIBITED BY SW11
7508 046626          GOTO  CMD8A
7509 046630          END ;OF IF MARGIN
7510
7511 046630          IF #SW0 SET.IN @SWR
7512 046640 104470  ECCDIS          ;DISABLE ERROR CORRECTION
7513 046642          ELSE
7514 046644 104502  CLRCSR          ;CLEAR MK11 CSRS
7515 046646          END ;OF IF
7516 046646          TYPE  MSG056          ;BELL = EACH PASS COMPLETE
7517
7518 046652          TYPE  MSG046          ;TO ESCAPE TYPE ANY KEY!
7519 046656 012737 047014 000060  MOV   #CMD8C,TKVEC
7520 046664 012737 000340 000062  MOV   #340,TKVEC+2
7521 046672 017700 134076          MOV   @STKB,R0          ;KILL ANY OLD INTERRUPT
7522 046676 042737 000200 177776  BIC   #BIT7,PSW          ;LOWER CPU PRIORITY TO 140
7523 046704 052777 000100 134060  BIS   #BIT6,@STKS          ;ENABLE KEYBOARD INTERRUPTS
7524
7525 046712 013700 002112          MOV   MARGIN,R0          ;SET MARGINS
7526 046716 006300          ASL   R0          ;SET MARGINS
7527 046720 010037 177750          MOV   R0,MAINT
7528
7529 046724          SET   HEADER,MUT
7530
7531 046740          CMD8B: FOR BANK := #0 TO #167
7532 046744 004737 042422          CALL  EXBANK
7533 046750          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
7534 046764 104511          INVALIDATE
7535 046766 004737 025214          CALL  MTO031
7536 046772          END ;OF IF ACFLAG
7537 046772          END ;OF FOR BANK
7538 047006          TYPE  $BELL          ;RING BELL
7539 047012          GOTO  CMD8B
7540
7541 047014 013706 002374          CMD8C: MOV   FSSTACK,SP          ;RECOVER OLD STACK POINTER
7542 047020 042777 000100 133744  BIC   #BIT6,@STKS
7543 047026 117700 133742          MOVB  @STKB,R0          ;READ CHAR TO KILL FLAG
7544 047032          POP   NOPAR,TKVEC+2,TKVEC,MARGIN,PATTERN,BANK
7545 047062          MAP   BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7546 047076 004737 042422          CALL  EXBANK
7547 047102 000207          RETURN
    
```

```

7550 047104 FSCMD9: SUBTST <<FS COMMAND 9 SUPER TIGHT SCOPE LOOP>>
:*****
:*SUBTEST FS COMMAND 9 SUPER TIGHT SCOPE LOOP
:*****
7551 047104 PUSH BANK,MARGIN,TKVEC,TKVEC+2,NOPAR
7552 047130 010637 002374 MOV SP,FSSTACK ;SAVE LAST GOOD STACK
7553 047134 TYPE MSG059 ;SUPER TIGHT SCOPE LOOP
7554 047140 IF #SW0 SET.IN @SWR
7555 047150 104470 ECCDIS ;DISABLE ERROR CORRECTION
7556 047152 ELSE
7557 047154 104502 CLRCSR ;CLEAR MK11 CSRS
7558 047156 END ;OF IF
7559 047156 1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-16777776)??
7560 047162 104413 RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
7561 047164 013737 074434 002106 MOV $HIOCT,BANK ;PUT MSB'S IN BANK
7562 047172 POP RO ;PUT LSB'S IN RO
7563 047174 000241 CLC
7564 047176 006100 ROL RO
7565 047200 006137 002106 ROL BANK
7566 047204 000241 CLC
7567 047206 006000 ROR RO
7568 047210 IF BANK GT #167 THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
7569 047220 062700 060000 ADD #FIRST,RO
7570 047224 IF #BIT0 SET.IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
7571 047232 IF RO HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER 16K
7572
7573 047240 CMD9A: TYPE MSG044 ;MARGIN(0,2,3,4,5)?
7574 047244 104413 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7575 047246 POP MARGIN ;PUT IN IN MARGIN
7576 047252 IF MARGIN GT #5 OR MARGIN EQ #1
7577 047272 GOTO CMD9A
7578 047274 END ;OF IF
7579 047274 IF MARGIN NE #0 AND #SW11 SET.IN @SWR
7580 047312 TYPE MSG045 ;MARGINS INHIBITED BY SW11
7581 047316 GOTO CMD9A
7582 047320 END ;OF IF MARGIN
7583
7584 047320 012737 000003 002100 MOV #3,NOPAR ;PARITY ACTION
7585 047326 012737 047444 002372 MOV #CMD9LOOP,PARHERE
7586 047334 SET4 #CMD9B ;TRAPS TO 4 GOTO CMD9B
7587 047342 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7588 047356 104511 INVALIDATE
7589 047360 TYPE MSG060 ;SWR ----> MEM, MEM ----> DISPLAY
7590
7591 047364 TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
7592 047370 012737 047464 000060 MOV #CMD9C,TKVEC
7593 047376 012737 000340 000062 MOV #340,TKVEC+2
7594 047404 017701 133364 MOV @$TKB,R1 ;KILL ANY OLD INTERRUPT
7595 047410 042737 000200 177776 BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
7596 047416 052777 000100 133346 BIS #BIT6,@$TKS ;ENABLE KEYBOARD INTERRUPTS
7597
7598 047424 013701 002112 MOV MARGIN,R1 ;SET MARGINS
7599 047430 006301 ASL R1 ;SET MARGINS
7600 047432 010137 177750 MOV R1,MAINT
7601
7602 047436 SUPERVISOR ;ENTER SUPERVISOR MODE
7603 047444 017710 133316 CMD9LOOP:MOV @SWR,(R0) ;WRITE SWITCH REGISTER INTO MEMORY
  
```

```

7604 047450 011077 133314      MOV      (R0),@DISPLAY      ;READ MEMORY INTO DISPLAY REGISTER
7605 047454 000773              BR          CMD9LOOP        ;LOOP TILL KEYBOARD INTERRUPT
7606
7607 047456 012716 047444      CMD9B:  MOV      #CMD9LOOP,(SP)
7608 047462 000002              RTI
7609
7610 047464 013706 002374              CMD9C:  MOV      FSSTACK,SP      ;RECOVER OLD STACK POINTER
7611 047470 042777 000100 133274  BIC      #BIT6,@$TKS
7612 047476 117700 133272      MOV      @$TKB,R0          ;READ CHAR TO KILL FLAG
7613 047502              POP      NOPAR,TKVEC+2,TKVEC,MARGIN,BANK
7614 047526              MAP      BANK              ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7615 047542 004737 042422      CALL    EXBANK
7616 047546 000207              RETURN
7617
7618 047550
  
```

```

FCMD10: SUBTST <<FS COMMAND 10 ERROR SUMMARY>>
:*****
:*SUBTEST FS COMMAND 10 ERROR SUMMARY
:*****
  
```

```

7619 047550      PUSH    R0,R2,R3,BANK
7620 047562      TYPDEC $ERTTL
7621 047570      TYPE   MSG079          ;ERROR(S) DETECTED
7622 047574      IF $ERTTL NE #0
7623 047602 005037 002424      CLR    SUCCESS
7624 047606      FOR BANK := #0 TO #167
7625 047612 013703 002106      MOV    BANK,R3
7626 047616 070327 000006      MUL   #6,R3
7627 047622 062703 000005      ADD   #5,R3
7628 047626      IFB CONFIG(R3) NE #0
7629 047634      IF SUCCESS IS FALSE
7630 047642      TYPE   MSG076          ;BANK ERRORS
7631 047646      SET SUCCESS
7632 047654      END ;OF IF SUCCESS
7633 047654      TYPCS BANK,3
7634 047664 116300 003016      MOV    CONFIG(R3),R0
7635 047670 042700 177400      BIC   #^C377,R0
7636 047674      TYPDEC R0
7637 047700      TYPE   $CRLF
7638 047704      END ;OF IFB CONFIG(R3)
7639 047704      END ;OF FOR BANK
7640 047720      END ;OF IF $ERTTL
7641 047720      POP   BANK,R3,R2,R0
7642 047732 000207      RETURN
  
```

7645 047734
 7646 047734
 7647 047764 010637 002374
 7648 047770
 7649 047774
 7650 050000 104413
 7651 050002
 7652 050006
 7653 050026
 7654 050030
 7655 050030
 7656 050046
 7657 050052
 7658 050054
 7659
 7660 050054
 7661 050064 104470
 7662 050066
 7663 050070 104502
 7664 050072
 7665 050072
 7666
 7667 050076
 7668 050102 012737 050240 000060
 7669 050110 012737 000340 000062
 7670 050116 017700 132652
 7671 050122 042737 000200 177776
 7672 050130 052777 000100 132634
 7673
 7674 050136 013700 002112
 7675 050142 006300
 7676 050144 010037 177750
 7677
 7678 050150
 7679
 7680 050164
 7681 050170 004737 042422
 7682 050174
 7683 050210 104511
 7684 050212 004737 023734
 7685 050216
 7686 050216
 7687 050232
 7688 050236
 7689
 7690 050240 013706 002374
 7691 050244 042777 000100 132520
 7692 050252 117700 132516
 7693 050256
 7694 050306
 7695 050322 004737 042422
 7696 050326 000207
 7697

```

FCMD11: SUBTST <<FS COMMAND 11 REFRESH TEST>>
:*****
:*SUBTEST FS COMMAND 11 REFRESH TEST
:*****
PUSH BANK,PATTERN,MARGIN,TKVEC,TKVEC+2,NOPAR
MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
TYPE MSG073 ;REFRESH TEST
CMD11A: TYPE MSG044 ;MARGIN(0,2,3,4,5)?
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP MARGIN ;PUT IN IN MARGIN
IF MARGIN GT #5 OR MARGIN EQ #1
GOTO CMD11A
END ;OF IF
IF MARGIN NE #0 AND #SW11 SET.IN @SWR
TYPE MSG045 ;MARGINS INHIBITED BY SW11
GOTO CMD11A
END ;OF IF MARGIN

IF #SW0 SET.IN @SWR
ECCDIS ;DISABLE ERROR CORRECTION
ELSE
CLRCSR ;CLEAR MK11 CSRS
END ;OF IF
TYPE MSG056 ;BELL = EACH PASS COMPLETE

TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
MOV #CMD11C,TKVEC
MOV #340,TKVEC+2
MOV @TKB,R0 ;KILL ANY OLD INTERRUPT
BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
BIS #BIT6,@TKS ;ENABLE KEYBOARD INTERRUPTS

MOV MARGIN,R0 ;SET MARGINS
ASL R0 ;SET MARGINS
MOV R0,MAINT

SET HEADER,MUT

CMD11B: FOR BANK := #0 TO #167
CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
INVALIDATE
CALL MTO022
END ;OF IF ACFLAG
END ;OF FOR BANK
TYPE $BELL ;RING BELL
GOTO CMD11B

CMD11C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
BIC #BIT6,@TKS
MOVB @TKB,R0 ;READ CHAR TO KILL FLAG
POP NOPAR,TKVEC+2,TKVEC,MARGIN,PATTERN,BANK
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
CALL EXBANK
RETURN
  
```

7700 050330
7701 050330
7702 050332
7703 050336 104413
7704 050340
7705 050342 042700 177760
7706 050346 110037 002451
7707 050352
7708 050354 000207
7709
7710 050356

```
FCMD12: SUBTST <<FS COMMAND 12 SET FILL COUNT>>  
:*****  
:*SUBTEST FS COMMAND 12 SET FILL COUNT  
:*****  
PUSH R0  
TYPE MSG085 ;FILL COUNT(OCTAL)?  
RDOCT  
POP R0  
BIC #^C17,R0  
MOVB R0,$FILLS  
POP R0  
RETURN
```

7711 050356
7712 050362
7713 050376 000207
7714
7715 050400

```
FCMD13: SUBTST <<FS COMMAND 13 ENTER KAMIKAZE MODE>>  
:*****  
:*SUBTEST FS COMMAND 13 ENTER KAMIKAZE MODE  
:*****  
TYPE MSG101 ;ENTERING KAMIKAZE MODE  
SET KAMIKAZE,SKIPKAMI  
RETURN
```

7716 050400
7717 050404 005037 002010
7718 050410
7719 050416 000207
7720
7721 050420

```
FCMD14: SUBTST <<FS COMMAND 14 EXIT KAMIKAZE MODE>>  
:*****  
:*SUBTEST FS COMMAND 14 EXIT KAMIKAZE MODE  
:*****  
TYPE MSG102 ;LEAVING KAMIKAZE MODE  
CLR KAMIKAZE  
SET SKIPKAMI  
RETURN
```

7722 050420
7723 050424 012737 001415 002700
7724 050432 104424
7725 050434 000207
7726
7727 050436

```
FCMD15: SUBTST <<FS COMMAND 15 TURN CACHE OFF>>  
:*****  
:*SUBTEST FS COMMAND 15 TURN CACHE OFF  
:*****  
TYPE MSG106 ;CACHE IS OFF  
MOV #BIT0!BIT2!BIT3!BIT8!BIT9,CACHK  
CACHOFF ;TURN CACHE OFF  
RETURN
```

7728 050436
7729 050442 012737 000001 002700
7730 050450 104423
7731 050452 000207
7732
7733 050454

```
FCMD16: SUBTST <<FS COMMAND 16 TURN CACHE ON>>  
:*****  
:*SUBTEST FS COMMAND 16 TURN CACHE ON  
:*****  
TYPE MSG107 ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)  
MOV #BIT0,CACHK  
CACHON ;TURN CACHE ON  
RETURN
```

7734 050454
7735 050460
7736 050466 000207

```
FCMD17: SUBTST <<FS COMMAND 17 RUN ONLY MULTIPOINT TESTS>>  
:*****  
:*SUBTEST FS COMMAND 17 RUN ONLY MULTIPOINT TESTS  
:*****  
TYPE MSG108 ;ONLY MULTIPOINT TESTS WILL BE RUN  
SET MULTIONLY  
RETURN
```

7739 050470
7740 050470
7741 050474 005037 002004
7742 050500 000207
7743
7744 050502

7745 050502
7746 050506 004737 050576
7747 050512
7748 050512
7749 050512
7750 050516 104413
7751 050520
7752 050522
7753 050534
7754 050536
7755 050536 070127 000006
7756 050542 052761 040000 003020
7757 050550
7758 050552
7759 050552
7760 050556
7761 050564 000207
7762
7763 050566

7764 050566
7765 050572 005037 002002
7766
7767
7768 050576
7769 050600 042761 040000 003020
7770 050606 042761 020000 003020
7771 050614
7772 050626 000207

```
FCMD18: SUBTST <<FS COMMAND 18 RESUME RUNNING BOTH SINGLE & MULTIPORT TESTS>>
:*****
:*SUBTEST FS COMMAND 18 RESUME RUNNING BOTH SINGLE & MULTIPORT TESTS
:*****
TYPE MSG109 ;BOTH SINGLE & MULTIPORT TESTS WILL BE RUN
CLR MULTIONLY
RETURN

FCMD19: SUBTST <<FS COMMAND 19 TEST ONLY SELECTED BANKS>>
:*****
:*SUBTEST FS COMMAND 19 TEST ONLY SELECTED BANKS
:*****
TYPE MSG105 ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
CALL CMD20A ;ERASE OLD SELECTIONS
BEGIN CMD19LOOP
REPEAT
TYPE MSG030 ;BANK(0-167)?
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP R1 ;PUT IT IN R1
IF R1 GT #167 OR R1 LT #0
LEAVE CMD19LOOP
END ;OF IF R1
MUL #6,R1
BIS #BIT14,CONFIG+2(R1)
END ;OF REPEAT
END CMD20LOOP
TYPE MSG110 ;ONLY SELECTED BANKS WILL BE TESTED
SET SELONLY
RETURN

FCMD20: SUBTST <<FS COMMAND 20 RESUME TESTING ALL BANKS>>
:*****
:*SUBTEST FS COMMAND 20 RESUME TESTING ALL BANKS
:*****
TYPE MSG111 ;ALL BANKS WILL BE TESTED
CLR SELONLY

:ENTRY POINT FROM CMD19
CMD20A: FOR R1 := #0 TO #167*3*2 BY #6
BIC #BIT14,CONFIG+2(R1) ;DESELECT BANK
BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
END ;OF FOR R1
RETURN
```

```

7775 050630
7776 050630 013700 002334
7777 050634 122700 000200
7778 050640 001003
7779 050642 005037 002256
7780 050646 000207
7781
7782 050650
7783 050654 104413
7784 050656
7785 050660 020027 000007
7786 050664 101371
7787 050666 012701 177774
7788 050672 005200
7789 050674 062701 000004
7790 050700 077003
7791 050702 010137 002256
7792 050706 000207
7793
    
```

```

WHICHCSR:SUBTST <<SUBR DETERMINE CORRECT CSR>>
:*****
:*SUBTEST SUBR DETERMINE CORRECT CSR
:*****
MOV MKCSRS,R0 ;GET CSR'S FLAG
CMPB #BIT7,R0 ;CSR 0?
BNE 1$ ;NO - SKIP
CLR CSRNO ;YES - SET IT UP
RETURN

1$: TYPE MSG022 ;WHICH CSR(0-7)
RDOCT ;GET OCTAL NUMBER
POP R0 ;PUT IN R0
CMP R0,#7 ;CHECK LIMIT
BHI 1$ ;IF BAD LOOP TILL HE TYPES IT RIGHT

2$: ADD #4,R1
SOB R0,2$
MOV R1,CSRNO
RETURN
    
```

```
1 .TITLE CEMKABO 1170 MAIN MEMORY DIAG 2
2 .IDENT /X01.60/
3 .SBTTL MULTIPROCESSOR SLAVE DISPATCHER
4 050710 TASKRET: IF #MASTER NE CPUBIT
5 050720 000402 BR TASKDO
6 050722 ELSE
7 050724 000207 RETURN
8 050726 END; OF IF #MASTER
9
10 050726 104424 TASKDO: CACHOFF ;TURN CACHE OFF
11 050730 MAP #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
12 050744 SET R0
13 050750 FOR R1 := #0 TO #6 BY #2
14 050752 IF R0 EQ ONES
15 050760 IF NOIIST IS TRUE
16 050766 013702 177764 MOV SYSTID,R2
17 050772 ELSE
18 050774 017702 131736 MOV @IISTACR,R2
19 051000 072227 177770 ASH #-8,R2
20 051004 042702 177774 BIC #^C3,R2
21 051010 END ;OF IF NOIIST
22 051010 SUPERVISOR ;ENTER SUPERVISOR MODE
23 051016 IF FIRST+PORTDIR(R1) EQ R2 THEN LET R0 := R1
24 051026 104417 KERNEL ;ENTER KERNEL MODE
25 051030 END ;OF IF R0
26 051030 END ;OF FOR R1
27 051042 IF R0 EQ ONES
28 051050 FATAL 41
29 051056 END ;OF IF R0
30 051056 SUPERVISOR ;ENTER SUPERVISOR MODE
31 051064 005760 062616 1$: TST FIRST+TASK(R0) ;WAIT FOR COMMAND
32 051070 003775 BLE 1$
33 051072 013737 062676 002676 MOV FIRST+MDISPLAY,MDISPLAY
34 051100 013777 002676 131662 MOV MDISPLAY,@DISPLAY
35 051106 116001 062616 MOVB FIRST+TASK(R0),R1 ;GET COMMAND
36 051112 104417 KERNEL ;ENTER KERNEL MODE
37 051114 PUSH R0
38 051116 CASE R1
39 051126 051234 TASK00 ;0 IS NOT A COMMAND
40 051130 051236 TASK01 ;MOVE PROGRAM TO BANK
41 051132 051330 TASK02 ;WRITE ONLY PATTERN IN HI BYTE
42 051134 051430 TASK03 ;READ ONLY PATTERN IN HI BYTE
43 051136 051664 TASK04 ;EXECUTION CONTENTION TEST
44 051140 051774 TASK05 ;PORT ARBITRATION SYMMETRY TEST
45 051142 052300 TASK06 ;STOP PROGRAM
46 051144 052326 TASK07 ;WRITE ONLY ADDRESS PATTERN
47 051146 052336 TASK10 ;READ ONLY ADDRESS PATTERN
48 051150 052536 TASK11 ;ASYNCHRONOUS ADDRESS UNIQUENESS TEST
49 051152 052644 TASK12 ;SKEWED UP ADDRESS TEST
50 051154 052726 TASK13 ;SETUP FOR ASRB TEST
51 051156 053276 TASK14 ;RESET FROM ASRB TEST
52 051160 053310 TASK15 ;RUN CSR TESTS
53 051162 053316 TASK16 ;SET MARGIN 6, CPU HAS I/O PRIORITY
54 051164 053334 TASK17 ;CREATE STALE DATA TEST
55 051166 053422 TASK20 ;CACHE FLUSH TEST
56 051170 END ;OF CASE R1
57 051176 POP R0
```

58 051200
59 051214
60 051222 052760 100000 062616
61 051230 104417
62 051232 000626
63 051234 000207

MAP #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
SUPERVISOR ;ENTER SUPERVISOR MODE
BIS #BIT15,FIRST+TASK(R0) ;INDICATE DONE
KERNEL ;ENTER KERNEL MODE
BR TASKRET
TASK00: RETURN

66 051236

TASK01: SUBTST <<MOVE PROGRAM TO NEW BANK>>

: *SUBTEST MOVE PROGRAM TO NEW BANK

67 051236

SUPERVISOR ;ENTER SUPERVISOR MODE

68 051244 013700 062106

MOV FIRST+BANK,R0

69 051250 104417

KERNEL ;ENTER KERNEL MODE

70 051252 004737 041456

CALL RELOC1

71 051256 012737 002000 002706

MOV #STACK,KSTACK

72 051264

POP R1,R0 ;GET OLD RETURN & R0

73 051270 013706 002706

MOV KSTACK,SP

74 051274

PUSH R0,R1 ;PUT BACK R0 & RETURN

75 051300

MAP #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0

76 051314 005037 002134

CLR RLFLAG

77 051320

SET NOSCOPE

;NO SCOPE LOOPS ALLOWED IN MULTIPROCESSOR TESTS

78 051326 000207

RETURN

79

80 051330

TASK02: SUBTST <<WRITE ONLY PATTERN IN HI BYTE>>

: *SUBTEST WRITE ONLY PATTERN IN HI BYTE

81 051330

SUPERVISOR ;ENTER SUPERVISOR MODE

82 051336 116005 062617

MOVB FIRST+TASK+1(R0),R5

83 051342 013737 062106 002106

MOV FIRST+BANK,BANK

84 051350 104417

KERNEL ;ENTER KERNEL MODE

85 051352

SET WRITEONLY

86 051360 012700 060000

MOV #FIRST,R0

87 051364 012701 040000

MOV #SIZE,R1

88 051370

CASE R5

89 051400 051530

SLPAT0 ;ZEROS

90 051402 051534

SLPAT1 ;ONES

91 051404 051542

SLPAT2 ;125252

92 051406 051550

SLPAT3 ;52525

93 051410 051556

SLPAT4 ;377

94 051412 051564

SLPAT5 ;177400

95 051414

END ;OF CASE R5

96 051422 005037 002502

CLR WRITEONLY

97 051426 000207

RETURN

```

100 051430
101 051430
102 051436 116005 062617
103 051442 013737 062106 002106
104 051450 104417
105 051452
106 051460 012700 060000
107 051464 012701 040000
108 051470
109 051500 051530
110 051502 051534
111 051504 051542
112 051506 051550
113 051510 051556
114 051512 051564
115 051514
116 051522 005037 002504
117 051526 000207
118
119 051530 005002
120 051532 000416
121 051534
122 051540 000413
123 051542 012702 125252
124 051546 000410
125 051550 012702 052525
126 051554 000405
127 051556 012702 000377
128 051562 000402
129 051564 012702 177400
130 051570
131 051570
132 051600 104502
133 051602
134 051610 012737 000207 177644
135 051616 012737 177646 002360
136 051624 004737 035204
137 051630 104415
138 051632
139 051644
140 051656 104416
141 051660 104424
142 051662 000207
    
```

```

TASK03: SUBTST <<READ ONLY PATTERN IN HI BYTE>>
:*****
:*SUBTEST READ ONLY PATTERN IN HI BYTE
:*****
SUPERVISOR ;ENTER SUPERVISOR MODE
MOVB FIRST+TASK+1(R0),R5
MOV FIRST+BANK,BANK
KERNEL ;ENTER KERNEL MODE
SET READONLY
MOV #FIRST,R0
MOV #SIZE,R1
CASE R5
SLPAT0 ;ZEROS
SLPAT1 ;ONES
SLPAT2 ;125252
SLPAT3 ;52525
SLPAT4 ;377
SLPAT5 ;177400
END ;OF CASE R5
CLR READONLY
RETURN

SLPAT0: CLR R2
BR SLPAT
SLPAT1: SET R2
BR SLPAT
SLPAT2: MOV #125252,R2
BR SLPAT
SLPAT3: MOV #52525,R2
BR SLPAT
SLPAT4: MOV #377,R2
BR SLPAT
SLPAT5: MOV #177400,R2
SLPAT: SUBTST <<SLAVE PATTERNS>>
:*****
:*SUBTEST SLAVE PATTERNS
:*****
CLEAR MARGIN,MAINT
CLRCSR ;CLEAR MK11 CSRS
BMOV MTP000
MOV #207,UIPAR2 ;PUT RETURN INST AFTER WRITE ROUTINE
MOV #UIPAR3,SUPDOADD
CALL REGCOPY
SAVREG
IF WRITEONLY IS TRUE THEN $CALL SUPDO1
IF READONLY IS TRUE THEN $CALL SUPDO3
RESREG
CACHOFF ;TURN CACHE OFF
RETURN
    
```

145 051664

TASK04: SUBTST <<EXECUTION CONTENTION TEST>>

: *SUBTEST EXECUTION CONTENTION TEST

146 051664

SUPERVISOR ;ENTER SUPERVISOR MODE

147 051672 013737 062106 002106

MOV FIRST+BANK,BANK

148 051700 104417

KERNEL ;ENTER KERNEL MODE

149 C-1702 005060 002636

CLR PTYBUF(R0)

150 051706 012737 000037 002366

MOV #37,REALPAT

151 051714 012737 060000 002360

MOV #FIRST,SUPDOADD

152 051722

PUSH R0

153 051724 062700 002636

ADD #PTYBUF,R0 ;R0 = PTYBUF + INDEX FOR CPU

154 051730 004737 026220

CALL SUPDO3

155 051734 104424

CACHOFF ;TURN CACHE OFF

156 051736

MAP #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0

157 051752

POP R0

158 051754

SUPERVISOR ;ENTER SUPERVISOR MODE

159 051762 016060 002636 062636

MOV PTYBUF(R0),FIRST+PTYBUF(R0)

160 051770 104417

KERNEL ;ENTER KERNEL MODE

161 051772 000207

RETURN

164 051774

TASK05: SUBTST <<PORT ARBITRATION SYMMETRY TEST>>

 :*SUBTEST PORT ARBITRATION SYMMETRY TEST

165 051774				SUPERVISOR	;ENTER SUPERVISOR MODE
166 052002	013737	062106	002106	MOV FIRST+BANK,BANK	
167 052010	104417			KERNEL	;ENTER KERNEL MODE
168 052012				MAP BANK	
169 052026				SUPERVISOR	;ENTER SUPERVISOR MODE
170 052034				BMOV TSK05A,FIRST+2	
171 052044	104417			KERNEL	;ENTER KERNEL MODE
172 052046	005060	002646		CLR PORTCOUNT(R0)	
173 052052	010004			MOV R0,R4	
174 052054				PUSH 100,102	
175 052064	012737	052224	000100	MOV #1\$,100	
176 052072	012737	000340	000102	MOV #340,102	
177 052100	012737	000100	177546	MOV #BIT6,LKS	;ENABLE INTERRUPTS
178 052106	005000			CLR R0	
179 052110	012701	060000		MOV #FIRST,R1	
180 052114	000235			SPL 5	
181 052116	012737	060002	002360	MOV #FIRST+2,SUPDOADD	
182 052124	004737	026234		CALL SUPDO4	
183 052130	104424			CACHOFF	;TURN CACHE OFF
184 052132	005037	177546		CLR LKS	
185 052136	000237			SPL 7	
186 052140	004737	026234		CALL SUPDO4	;DO TEST AS BACKGND WHILE OTHER CPU'S FINISH
187 052144	104424			CACHOFF	
188 052146				MAP #0	;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
189 052162	010400			MOV R4,R0	
190 052164				SUPERVISOR	;ENTER SUPERVISOR MODE
191 052172	016060	002646	062646	MOV PORTCOUNT(R0),FIRST+PORTCOUNT(R0)	
192 052200	104417			KERNEL	;ENTER KERNEL MODE
193 052202				CLEAR MARGIN,MAINT	
194 052212				POP 102,100	
195 052222	000207			RETURN	
196 052224				1\$: PUSH R0	
197 052226	010400			MOV R4,R0	
198 052230	005260	002646		INC PORTCOUNT(R0)	
199 052234				POP R0	
200 052236	000002			RTI	
201 052240	010505			TSK05A: MOV R5,R5	
202 052242	010505			MOV R5,R5	
203 052244	010505			MOV R5,R5	
204 052246	010505			MOV R5,R5	
205 052250	010505			MOV R5,R5	
206 052252	010505			MOV R5,R5	
207 052254	010505			MOV R5,R5	
208 052256	010505			MOV R5,R5	
209 052260	010505			MOV R5,R5	
210 052262	010505			MOV R5,R5	
211 052264	010505			MOV R5,R5	
212 052266	010505			MOV R5,R5	
213 052270	010505			MOV R5,R5	
214 052272	010505			MOV R5,R5	
215 052274	077017			SOB R0,TSK05A	
216 052276	000207			RETURN	

219 052300
 220 052300 012706 000700
 221 052304 042737 001001 177572
 222 052312 005037 002676
 223 052316 005077 130446
 224 052322 000137 011720
 225
 226 052326
 227 052326
 228 052334 000403
 229
 230 052336
 231 052336
 232 052344
 233 052352 116001 062617
 234 052356 013737 062106 002106
 235 052364 104417
 236 052366
 237 052372
 238 052400 012700 060000
 239 052404 005002
 240 052406 010004
 241 052410 010205
 242 052412
 243 052414
 244 052422 012700 160000
 245 052426 012702 000001
 246 052432 012704 060000
 247 052436 012705 100001
 248 052442
 249 052442 012701 040000
 250 052446 010103
 251 052450 012737 000207 177650
 252 052456 012737 177652 002360
 253 052464
 254 052474 104502
 255 052476
 256 052510
 257 052522 104424
 258 052524
 259 052534 000207

```

TASK06: SUBTST <<STOP SLAVE PROGRAM>>
:*****
:*SUBTEST      STOP SLAVE PROGRAM
:*****
      MOV      #USESTK,SP
      BIC      #BIT9!BIT0,MMRO      ;DEENERGIZE MEMORY MANAGEMENT
      CLR      MDISPLAY
      CLR      @DISPLAY
      JMP      STOPCPU

TASK07: SUBTST <<WRITE ONLY ADDRESS PATTERN IN HI BYTE>>
:*****
:*SUBTEST      WRITE ONLY ADDRESS PATTERN IN HI BYTE
:*****
      SET      WRITEONLY
      BR       TASK7A

TASK10: SUBTST <<READ ONLY ADDRESS PATTERN IN HI BYTE>>
:*****
:*SUBTEST      READ ONLY ADDRESS PATTERN IN HI BYTE
:*****
      SET      READONLY

TASK7A: SUPERVISOR      ;ENTER SUPERVISOR MODE
      MOV      FIRST+TASK+1(R0),R1
      MOV      FIRST+BANK,BANK
      KERNEL      ;ENTER KERNEL MODE
      IF R1 EQ #0
      BMOV     MTP001      ;ADDRESS PATTERN
      MOV      #FIRST,R0
      CLR      R2
      MOV      R0,R4
      MOV      R2,R5
      ELSE
      BMOV     MTP002      ;COMPLEMENT ADDRESS PATTERN
      MOV      #LAST+2,R0
      MOV      #1,R2
      MOV      #FIRST,R4
      MOV      #100001,R5
      END :OF IF R1
      MOV      #SIZE,R1
      MOV      R1,R3
      MOV      #207,UIPAR4      ;RETURN INSTRUCTION
      MOV      #UIPAR5,SUPDOADD ;READ ONLY STARTING ADDRESS
      CLEAR   MARGIN,MAINT
      CLRCSR      ;CLEAR MK11 CSRS
      IF WRITEONLY IS TRUE THEN $CALL SUPDO1
      IF READONLY IS TRUE THEN $CALL SUPDO3
      CACHOFF      ;TURN CACHE OFF
      CLEAR   READONLY,WRITEONLY
      RETURN
    
```

262 052536

TASK11: SUBTST <<RUN FIVE PATTERNS IN DESIGNATED 1/4 BANK>>

 :*SUBTEST RUN FIVE PATTERNS IN DESIGNATED 1/4 BANK

263 052536
 264 052544 116001 062617
 265 052550 013737 062106 002106
 266 052556 104417
 267 052560 072127 000011
 268 052564 010100
 269 052566 012701 010000
 270 052572
 271 052606 004737 051530
 272 052612 004737 051534
 273 052616 004737 051542
 274 052622 004737 051550
 275 052626 004737 051556
 276 052632
 277 052642 000207
 278
 279 052644

SUPERVISOR ;ENTER SUPERVISOR MODE
 MOVB FIRST+TASK+1(R0),R1
 MOV FIRST+BANK,BANK
 KERNEL ;ENTER KERNEL MODE
 ASH #9,R1
 MOV R1,R0
 MOV #SIZE/4,R1
 SET WRITEONLY,READONLY
 CALL SLPAT0
 CALL SLPAT1
 CALL SLPAT2
 CALL SLPAT3
 CALL SLPAT4
 CLEAR WRITEONLY,READONLY
 RETURN

TASK12: SUBTST <<WRITE & READ A SKEWED UP ADDRESS TEST>>

 :*SUBTEST WRITE & READ A SKEWED UP ADDRESS TEST

280 052644
 281 052652 116001 062617
 282 052656 104417
 283
 284
 285
 286
 287 052660 010004
 288 052662 060104
 289
 290 052664 012705 010000
 291 052670 012703 060000
 292 052674
 293 052702 104415
 294 052704 004737 026042
 295 052710
 296 052716 104416
 297 052720 004737 026056
 298 052724 000207

SUPERVISOR ;ENTER SUPERVISOR MODE
 MOVB FIRST+TASK+1(R0),R1
 KERNEL ;ENTER KERNEL MODE

 :PAY ATTENTION NOW!
 :R0 = MULTIPOINT ID # (0,2,4,6)
 :R1 = OFFSET STARTING ADDRESS (0,2,4,6)
 MOV R0,R4
 ADD R1,R4 ;R4 IS NOW UNIQUE AMONST THE 4 CPU'S

 MOV #SIZE/4,R5
 MOV #FIRST,R3
 BMOV MTPA32
 SAVREG
 CALL SUPD01
 BMOV MTPB32
 RESREG
 CALL SUPD02
 RETURN

301 052726

TASK13: SUBTST <<SETUP FOR ASRB TEST>>

 :*SUBTEST SETUP FOR ASRB TEST

302 052726
 303 052734 012777 177640 130000
 304 052742
 305 052750 013737 062106 002106
 306 052756 052760 100000 062616
 307 052764 104417
 308 052766
 309 053002
 310 053012
 311 053016
 312 053024
 313 053032 005737 060002
 314 053036 001775
 315
 316 053040 017701 127676
 317 053044 004711
 318 053046
 319 053062
 320 053074 104417
 321 053076
 322 053076 000207
 323
 324 053100 000240
 325 053102 000240
 326 053104 000240
 327 053106 000240
 328 053110 000240
 329 053112 000240
 330 053114 000240
 331 053116 000240
 332 053120 000240
 333 053122
 334 053130 106212
 335 053132 000137 053136

```

BMOV TSK13A
MOV #UIPAR0,@IISTVEC ;CAUSE WE SIMULATE TRAPS
SUPERVISOR ;ENTER SUPERVISOR MODE
MOV FIRST+BANK,BANK
BIS #BIT15,FIRST+TASK(R0) ;ACK
KERNEL ;ENTER KERNEL MODE
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
IF CPUBIT NE #MASTER
FOR R2 := #FIRST TO #FIRST+34 BY #4
FOR I := #1 TO #1000.
SUPERVISOR ;ENTER SUPERVISOR MODE
1$: TST FIRST+2
BEQ 1$
;DISPATCH THRU THE IIST VECTOR
MOV @IISTVEC,R1
CALL (R1)
END ;OF FOR I
END ;OF FOR R2
KERNEL ;ENTER KERNEL MODE
END ;OF IF CPUBIT
RETURN
    
```

```

TSK13A: NOP ;V177640
NOP ;V177642
NOP ;V177644
NOP ;V177646
NOP ;V177650
NOP ;V177652
NOP ;V177654
NOP ;V177656
NOP ;V177660
SUPERVISOR ;V177662 ;ENTER SUPERVISOR MODE
ASRB (R2) ;V177670
JMP TSK13B ;V177672
    
```

338 053136

TSK13B: SUBTST <<RECOVER FROM TASK13 - TRAP FROM IIST>>
:*****
:*SUBTEST RECOVER FROM TASK13 - TRAP FROM IIST
:*****

339 053136
340 053140
341 053146 163777 002730 127566
342 053154 163777 002730 127560
343 053162
344 053200
345 053202 005060 002656
346 053206 062777 000002 127526
347 053214
348 053232
349 053232
350 053246 016060 002656 062656
351 053254
352 053270 005037 060002
353 053274 000207

ON.ERROR
SET SLFLAG(R0) ;I LOCKED IT!!!
SUB SLAVES,@IISTVEC
SUB SLAVES,@IISTVEC
IF @IISTVEC LO #UIPAR0 THEN LET @IISTVEC := #UIPAR0
ELSE
CLR SLFLAG(R0) ;IT WAS LOCKED BY SOMEONE ELSE
ADD #2,@IISTVEC
IF @IISTVEC HI #UDPAR1 THEN LET @IISTVEC := #UDPAR1
END ;OF ON.NOERROR
MAP #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
MOV SLFLAG(R0),FIRST+SLFLAG(R0)
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
CLR FIRST+2
RETURN

356 053276

TASK14: SUBTST <<RESET FROM ASRB TEST>>
 :*****
 :*SUBTEST RESET FROM ASRB TEST
 :*****

357 053276 000237
 358 053300 012777 061216 127434
 359 053306 000207

SPL 7
 MOV #SLAVUP,@IISTVEC
 RETURN

360
 361 053310

TASK15: SUBTST <<RUN SLAVE IN CSR TESTS>>
 :*****
 :*SUBTEST RUN SLAVE IN CSR TESTS
 :*****

362 053310 004737 016664
 363 053314 000207

CALL MKCONTROL
 RETURN

364
 365 053316

TASK16: SUBTST <<SET MARGIN 6, CPU HAS I/O PRIORITY>>
 :*****
 :*SUBTEST SET MARGIN 6, CPU HAS I/O PRIORITY
 :*****

366 053316 012737 000014 177750
 367 053324 012737 000006 002112
 368 053332 000207

MOV #14,MAINT
 MOV #6,MARGIN
 RETURN

369
 370 053334

TASK17: SUBTST <<CREATE STALE DATA TEST>>
 :*****
 :*SUBTEST CREATE STALE DATA TEST
 :*****

371 053334
 372 053342 013737 062106 002106
 373 053350 104417

SUPERVISOR
 MOV FIRST+BANK,BANK
 KERNEL
 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 SUPERVISOR ;ENTER SUPERVISOR MODE

374 053352
 375 053366
 376 053374 012700 060000
 377 053400 012701 060004
 378 053404 022711 047507
 379 053410 001375
 380 053412 012710 177777
 381 053416 104417
 382 053420 000207

1\$:
 MOV #FIRST,R0
 MOV #FIRST+4,R1
 CMP #'GO,(R1) ;TIME TO GO?
 BNE 1\$;NO - LOOP
 MOV #-1,(R0) ;WRITE WORD
 KERNEL ;ENTER KERNEL MODE
 RETURN

385 053422

TASK20: SUBTST <<CACHE FLUSH TEST>>

 :*SUBTEST CACHE FLUSH TEST

386 053422
 387 053430
 388 053436 013737 062106 002106
 389 053444 104417
 390 053446
 391 053462 012700 060000
 392 053466 012701 060004
 393 053472 005002
 394
 395
 396 053474 077201
 397
 398 053476
 399 053504 004737 177640
 400 053510 005060 000004
 401 053514 104417
 402
 403
 404 053516 022701 000000
 405 053522 001411
 406 053524 012737 000000 002050
 407 053532 010137 002056
 408 053536 012737 060000 002040
 409 053544 104055
 410
 411
 412 053546 022702 177777
 413 053552 001411
 414 053554 012737 177777 002050
 415 053562 010237 002056
 416 053566 012737 060000 002040
 417 053574 104056
 418
 419 053576 104424
 420 053600 000207
 421
 422 053602

BMOV TSK20A
 SUPERVISOR ;ENTER SUPERVISOR MODE
 MOV FIRST+BANK,BANK
 KERNEL ;RETURN TO KERNEL MODE
 MAP BANK
 MOV #FIRST,R0
 MOV #FIRST+4,R1
 CLR R2
 1\$: ;WAIT FOR ANY FLUSH TO FINISH
 SOB R2,1\$
 SUPERVISOR ;ENTER SUPERVISOR MODE
 CALL FASTCITY
 CLR 4(R0) ;KILL THE GO SIGNAL
 KERNEL ;ENTER KERNEL MODE
 ;VERIFY STALE DATA
 CMP #0,R1 ;CHECK FOR STALE DATA
 BEQ 2\$;OK - SKIP
 MOV #0,GOOD ;REPORT ERROR
 MOV R1,BAD
 MOV #FIRST,ADDRESS
 ERROR +55
 2\$: ;VERIFY THAT FLUSHING GETS RID OF STALE DATA
 CMP #-1,R2 ;CHECK FOR GOOD DATA
 BEQ 5\$;OK - SKIP
 MOV #-1,GOOD ;REPORT ERROR
 MOV R2,BAD
 MOV #FIRST,ADDRESS
 ERROR +56
 5\$: CACHOFF ;TURN CACHE OFF
 RETURN

TSK20A: SUBTST <<CACHE FLUSH TEST (FOR PAR'S)>>

 :*SUBTEST CACHE FLUSH TEST (FOR PAR'S)

423 053602 012737 000401 177746
 424 053610 012710 000000
 425 053614 005710
 426 053616 012711 047507
 427 053622 077201
 428 053624 011001
 429 053626 012737 000401 177746
 430 053634 011002
 431 053636 000207

1\$: MOV #BIT8!BIT0,CONTRL ;V177640 ;TURN ON CACHE & FLUSH IT!
 MOV #0,(R0) ;V177646 ;WRITE FRESH DATA
 TST (R0) ;V177652 ;READ - CREATING A CACHE HIT
 MOV #'GO,(R1) ;V177654 ;SIGNAL SLAVES TO MAKE STALE DATA
 SOB R2,1\$;V177660 ;WAIT FOR SLAVES TO FINISH
 MOV (R0),R1 ;V177662 ;STALE DATA ---> R1
 MOV #BIT8!BIT0,CONTRL ;V177664 ;FLUSH CACHE AGAIN
 MOV (R0),R2 ;V177672 ;FRESH DATA ---> R2
 RETURN ;V177674

434 053640
435 053640
436 053654 012700 060000
437 053660
438 053666 004737 074720
439 053672 013701 002714
440 053676 012702 000020
441 053702
442 053710
443 053710 006101
444 053712
445 053714 012720 005210
446 053720
447 053722 012720 005710
448 053726
449 053726
450 053730 104417
451 053732
452 053734
453 053742 012740 000207
454 053746 104417
455 053750 000207

```
MRUNSETUP:SUBTST      <<SETUP EXECUTION CONTENTION TEST>>
:*****
:*SUBTEST      SETUP EXECUTION CONTENTION TEST
:*****
MAP      BANK      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
MOV      #FIRST,R0
WHILE R0 LO #LAST
CALL $RAND
MOV      SEEDHI,R1
MOV      #16.,R2
SUPERVISOR      ;ENTER SUPERVISOR MODE
THRU R2
ROL R1
ON.ERROR
MOV      #5210,(R0)+      ;INC (R0) INSTRUCTION
ELSE
MOV      #5710,(R0)+      ;TST (R0) INSTRUCTION
END ;OF ON.ERROR
END ;OF THRU R2
KERNEL      ;ENTER KERNEL MODE
END ;OF WHILE R0
SUPERVISOR      ;ENTER SUPERVISOR MODE
MOV      #207,-(R0)      ;RETURN INSTRUCTION
KERNEL      ;ENTER KERNEL MODE
RETURN
```

458 053752
 459
 460 053752 012737 177765 002676
 461 053760 013777 002676 127002
 462 053766
 463 053774 013737 002510 002114
 464 054002 004737 042422
 465 054006 012737 000020 002114
 466 054014
 467 054030
 468 054044
 469 054052
 470 054076 104417
 471 054100
 472 054102 010001
 473 054104 006301
 474 054106 012761 000013 002616
 475 054114
 476 054124 004737 050726
 477 054130
 478 054144
 479 054146 010001
 480 054150 006301
 481 054152 012737 061165 002574
 482 054160 004737 072154 2\$:
 483 054164
 484 054166 010037 002056
 485 054172 104060
 486 054174
 487 054200
 488 054200 005761 002616
 489 054204 100365
 490 054206
 491
 492 054216 005000
 493 054220 077001 1\$:
 494 054222
 495 054226
 496 054234
 497 054242 012712 000001
 498 054246 004737 054544
 499
 500
 501 054252 005000
 502 054254
 503 054256
 504 054264 005261 002646
 505 054270 005200
 506 054272
 507 054272
 508
 509
 510 054304
 511 054312 104033

```

SUPM13: SUBTST <<MULTIPOINT ASRB TEST (IN SUPERVISOR SPACE)>>
:*****
:*SUBTEST      MULTIPOINT ASRB TEST (IN SUPERVISOR SPACE)
:*****
      .ENABL  LSB
      MOV    #-13,MDISPLAY
      MOV    MDISPLAY,@DISPLAY
      FOR BANK := #1 TO #167
      MOV    ALLCPUS,CPUBIT
      CALL   EXBANK
      MOV    #MASTER,CPUBIT
      IF PFLAG IS FALSE AND ACFLAG IS TRUE
      MAP BANK                                ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      SUPERVISOR                             ;ENTER SUPERVISOR
      CLEAR PORTCOUNT,PORTCOUNT+2,PORTCOUNT+4,PORTCOUNT+6,FIRST+2
      KERNEL                                  ;ENTER KERNEL MODE
      FOR R0 := #0 TO SLAVES
      MOV    R0,R1
      ASL    R1
      MOV    #13,TASK(R1)                    ;ORDERS TO SETUP FOR ASRB TEST
      END ;OF FOR R0
      CALL   TASKDO
      MAP BANK
      FOR R0 := #0 TO SLAVES
      MOV    R0,R1
      ASL    R1
      MOV    #5*SECONDS,HUNGTIME
      CALL   QUE
      ON.ERROR
      MOV    R0,BAD
      ERROR  +60                             ;SLAVE HUNG SYSTEM!
      $RETURN ERROR
      END ;OF ON.ERROR
      TST    TASK(R1)
      BPL    2$
      END ;OF FOR R0
      ;I ASSUME THEY HAVE FINISHED COMPLETELY AFTER THIS DELAY
      CLR R0
      SOB R0,1$
      FOR R2 := #FIRST TO #FIRST+34 BY #4
      FOR I := #1 TO #1000.
      SUPERVISOR                             ;ENTER SUPERVISOR MODE
      MOV    #1,(R2)                          ;UNLOCK TEST LOCATION
      CALL   WAKEUP

      ;RECORD WINNER(S)
      CLR R0
      FOR R1 := #0 TO #6 BY #2
      IF SLFLAG(R1) IS TRUE
      INC PORTCOUNT(R1)
      INC R0
      END ;OF IF SLFLAG
      END ;OF FOR R1

      ;CHECK FOR ONE & ONLY ONE LOCK
      IF R0 NE #1
      ERROR +33
  
```

512	054314				END ;OF IF R0	
513	054314				END ;OF FOR I	
514	054330				END ;OF FOR R2	
515	054342	104417			KERNEL	
516	054344	004737	054472		CALL M13BAL	;CALL BALANCE TEST
517	054350				END ;OF IF PFLAG	
518	054350				END ;OF FOR BANK	
519	054364				FOR R0 := #1 TO SLAVES	
520	054370	010001			MOV R0,R1	
521	054372	006301			ASL R1	
522	054374	012761	000014	002616	MOV #14,TASK(R1)	;ORDERS TO RESET FROM ASRB TEST
523	054402				END ;OF FOR R0	
524	054412				FOR R0 := #1 TO SLAVES	
525	054416	010001			MOV R0,R1	
526	054420	006301			ASL R1	
527	054422	012737	061165	002574	MOV #5*SECONDS,HUNGTIME	
528	054430	004737	072154	3\$:	CALL QUE	
529	054434				ON.ERROR	
530	054436	010037	002056		MOV R0,BAD	
531	054442	104060			ERROR +60	;SLAVE HUNG SYSTEM!
532	054444				\$RETURN ERROR	
533	054450				END ;OF ON.ERROR	
534	054450	005761	002616		TST TASK(R1)	;WAIT FOR ALL CPU'S DONE
535	054454	100365			BPL 3\$	
536	054456				END ;OF FOR R0	
537	054466				\$RETURN NOERROR	
538					.DSABL LSB	

541 054472

M13BAL: SUBTST <<BALANCE TEST>>

:SUBTEST BALANCE TEST

542 054472
543 054472
544 054474 010001
545 054476 006301
546 054500 013702 002646
547 054504 166102 002646
548 054510 062702 000400
549 054514
550 054526 104053
551 054530
552 054532
553 054532
554 054542
555 054542 000207
556
557
558 054544
559
560 054552 000240
561 054554 000240
562 054556 005000
563 054560 017701 126156
564 054564 004711
565 054566 000207

BEGIN ASRBALANCE
FOR R0 := #0 TO SLAVES
MOV R0,R1
ASL R1
MOV PORTCOUNT,R2
SUB PORTCOUNT(R1),R2
ADD #256.,R2
IF R2 LT #0 OR R2 GT #256.*2
ERROR +53
LEAVE ASRBALANCE
END :OF IF R2
END :OF FOR R0
END ASRBALANCE
RETURN

:# HERE & IN NEXT INST IS + OR - COUNT SPEC

WAKEUP: :WAKE UP EVERY CPU
SET FIRST+2
:INCLUDING MYSELF
NOP
NOP
CLR R0
MOV @IISTVEC,R1
CALL (R1)
RETURN

568
 569 054570
 570 054602
 571 054610
 572 054616
 573 054624 104417
 574 054626
 575 054626
 576 054634
 577 054640
 578 054642
 579 054646
 580 054646 000137 065330
 581
 582 054652

 583 054652
 584 054664
 585 054666 104505
 586 054670 104426
 587 054672 013700 002144
 588 054676 000300
 589 054700 042700 177600
 590 054704 104503
 591 054706
 592 054712 005037 002050
 593 054716
 594 054724 011137 002056
 595 054730 016137 000002 002060
 596 054736 104417
 597 054740 110037 002062
 598 054744 105037 002063
 599 054750 004737 065564
 600 054754 104034
 601 054756
 602 054760 000002

```

.SBTTL ERROR DATA (SUPERVISOR) SETUP STUFF
$PER25: LET ADDRESS := R1 - #2
        IF ABORTFLAG IS FALSE
            SUPERVISOR ;ENTER SUPERVISOR 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 R0 ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
        CHK1DIS
        READCSR
        MOV CSR,R0
        SWAB R0
        BIC #^C177,R0
        CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
        LET ADDRESS := R1
        CLR GOOD
        SUPERVISOR ;ENTER SUPERVISOR MODE
        MOV (R1),BAD
        MOV 2(R1),BAD2
        KERNEL ;ENTER KERNEL MODE
        MOVB R0,BAD3
        CLRB BAD3+1
        CALL PERBNK
        ERROR +34
        POP R0
        RTI
    
```

605 054762
 606 054766
 607 055000
 608 055006
 609 055014
 610 055022 104417
 611 055024
 612 055024 000137 065330
 613
 614 055030

```
$PER30: LET GOOD := R1
        LET ADDRESS := (SP) - 16
        IF ABORTFLAG IS FALSE
            SUPERVISOR ;ENTER SUPERVISOR 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
:*****
```

615 055030
 616 055042 010637 055156
 617 055046 012737 055122 000004
 618 055054 012737 055122 000114
 619 055062 013700 002040
 620 055066 052737 000001 177750
 621 055074 052737 004000 177746
 622 055102
 623 055110 011037 002056
 624 055114 104417
 625 055116 005037 002142
 626 055122 013706 055156
 627 055126 042737 004000 177746
 628 055134 042737 000001 177750
 629 055142
 630 055154 000207
 631 055156 000000

```
        PUSH R0,4,114
        MOV SP,GETDA1
        MOV #1$,4
        MOV #1$,114
        MOV ADDRESS,R0
        BIS #BIT0,MAINT
        BIS #BIT11,CONTRL ;DISABLE ABORTS
        SUPERVISOR
        MOV (R0),BAD
        KERNEL
        CLR ABORTFLAG
        1$: MOV GETDA1,SP ;RESTORE KNOWN GOOD STACK POINTER
        BIC #BIT11,CONTRL ;ENABLE ABORTS
        BIC #BIT0,MAINT
        POP 114,4,R0
        RETURN
GETDA1: 0
```

634				.SBTTL POWER FAIL AUTO RESTART
635				.SBTTL ROUTINE POWER DOWN AND UP
636				*****
637				:POWER DOWN ROUTINE
638	055160			\$PWRDN: IF CPUBIT NE #MASTER
639	055170	000000		DNSLAV: HALT
640	055172	000776		BR DNSLAV
641	055174			END ;OF IF CPUBIT
642	055174	012737	000376 002760	MOV #376,PWLOCK
643				;SAVE CACHE & MARGIN STATUS
644	055202			PUSH CONTRL,MAINT
645	055212	104423		CACHON ;TURN CACHE ON
646	055214	012737	056134 000024	MOV #SILLUP,PWRVEC ;:SET FOR FAST UP
647	055222	012737	000340 000026	MOV #340,PWRVEC+2 ;:PRIO:7
648	055230			PUSH R0,R1,R2,R3,R4,R5,CSRNO
649				;SAVE USER PAR'S & PDR7
650	055250	012700	177700	MOV #177700,R0
651	055254	012701	000021	MOV #17.,R1
652	055260			1\$: PUSH -(R0)
653	055262	077102		SOB R1,1\$
654				;SAVE SUPERVISOR PAR'S
655	055264	012700	172300	MOV #172300,R0
656	055270	012701	000020	MOV #16.,R1
657	055274			2\$: PUSH -(R0)
658	055276	077102		SOB R1,2\$
659	055300			IF RLFLAG IS TRUE THEN \$CALL WOOPS
660				;COPY KERNEL MAP TO USER & SUPERVISOR
661	055312	012700	172300	MOV #KIPDR0,R0
662	055316	012701	177600	MOV #UIPDR0,R1
663	055322	012702	172200	MOV #SIPDR0,R2
664	055326	012703	000040	MOV #32.,R3
665	055332	011021		3\$: MOV (R0),(R1)+
666	055334	012022		MOV (R0)+,(R2)+
667	055336	077303		SOB R3,3\$

```

669 ;SAVE USER & SUPERVISOR STACK POINTERS
670 055340 USER
671 055346 010600 MOV USP,R0
672 055350 104417 KERNEL ;ENTER KERNEL MODE
673 055352 PUSH R0
674 055354 SUPERVISOR ;ENTER SUPERVISOR MODE
675 055362 010600 MOV SSP,R0
676 055364 104417 KERNEL ;ENTER KERNEL MODE
677 055366 PUSH R0
678 ;SAVE MK11 REGISTERS
679 055370 013701 002334 MOV MKCSRS,R1 ;GET CSR'S BYTE
680 055374 BEGIN LCSRSAVE
681 055374 FOR CSRNO := #0 TO #34 BY #4
682 055400 106301 ASLB R1
683 055402 ON.ERROR
684 055404 104426 READCSR
685 055406 PUSH CSR,CSR+2
686 055416 END ;OF ON.ERROR
687 055416 IFB R1 EQ #0 THEN LEAVE LCSRSAVE
688 055422 END ;OF FOR CSRNO
689 055440 END LCSRSAVE
690 ;SAVE MMR0,1,2,3
691 055440 PUSH MMR0,MMR1,MMR2,MMR3
692 ;SAVE KERNEL PAR'S
693 055460 012700 172400 MOV #172400,R0
694 055464 012701 000020 MOV #16.,R1
695 055470 4$: PUSH -(R0)
696 055472 077102 SOB R1,4$
697 ;SAVE UNIBUS MAP REGISTERS
698 055474 PUSH MAPH36,MAPL36,MAPH0,MAPL0
699 ;SAVE POSSIBLE SOFTWARE SWITCH REGISTER
700 055514 PUSH @SWR
701 ;SAVE STACK POINTER
702 055520 010637 056140 MOV SP,$SAVR6 ;;SAVE SP
703 ;NOW SET UP REAL VECTOR
704 055524 012737 055536 000024 MOV #$PWRUP,PWRVEC ;;SET UP VECTOR
705 055532 000000 $DOWN: HALT
706 055534 000776 BR $DOWN ;;HANG UP
    
```

```

709 :*****
710 :POWER UP ROUTINE
711 055536 $PWRUP: IF MULTIPOST IS TRUE THEN JUMPTO MULTIUP
712 055550 012737 056134 000024 MOV #SILLUP,PWRVEC ;;SET FOR FAST DOWN
713 :RESTORE STACK POINTER
714 055556 013706 056140 MOV $SAVR6,SP ;;GET SP
715 055562 005037 056140 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
716 055566 005237 056140 1$: INC $SAVR6 ;;WAIT FOR THE INC
717 055572 001375 BNE 1$ ;;OF A WORD
718 :RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
719 055574 POP @SWR
720 :RESTORE UNIBUS MAP
721 055600 POP MAPLO,MAPHO,MAPL36,MAPH36
722 055620 004737 042026 CALL LOWMAP ;SETUP LOWER 16K OF UNIBUS MAP
723 :RESTORE KERNEL PAR'S & PDR'S
724 055624 012700 172340 MOV #172340,R0
725 055630 012702 172300 MOV #KIPDR0,R2
726 055634 012701 000020 MOV #16.,R1
727 055640 6$: POP (R0)+
728 055642 012722 077406 MOV #77406,(R2)+
729 055646 077104 SOB R1,6$
730 :RESTORE MMR3,2,1,0
731 055650 POP MMR3,MMR2,MMR1,MMR0
732 :RESTORE MK11 REGISTERS
733 055670 013701 002334 MOV MKCSRS,R1 ;GET CSR'S BYTE
734 055674 042701 177400 BIC #177400,R1
735 055700 BEGIN LCSRRESTORE
736 055700 FOR CSRNO := #34 DOWNT0 #0 BY #4
737 055706 106201 ASRB R1
738 055710 ON.ERROR
739 055712 POP CSR+2,CSR
740 055722 104425 LOADCSR
741 055724 END ;OF ON.ERROR
742 055724 IFB R1 EQ #0 THEN LEAVE LCSRRESTORE
743 055730 END ;OF FOR CSRNO
744 055746 END LCSRRESTORE
745 :COPY KERNEL MAP TO USER & SUPERVISOR
746 055746 012700 172300 MOV #KIPDR0,R0
747 055752 012701 177600 MOV #UIPDR0,R1
748 055756 012702 172200 MOV #SIPDR0,R2
749 055762 012703 000040 MOV #32.,R3
750 055766 011021 3$: MOV (R0),(R1)+
751 055770 012022 MOV (R0)+,(R2)+
752 055772 077303 SOB R3,3$
  
```


788 056142

```
MULTIUP:SUBTST <<MULTIPLE CPU'S POWERING UP>>
:*****
:*SUBTEST      MULTIPLE CPU'S POWERING UP
:*****
```

789 056142 106237 002760
 790 056146 103002
 791 056150 000000
 792 056152 000776
 793 056154 000137 004352
 794
 795 056160

```
ASRB PWLOCK
BCC UP2
MULHLT: HALT ;BECAUSE I'VE BEEN LOCKED OUT
BR MULHLT
UP2: JMP START
```

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

796 056160
 797 056164 005037 002106
 798 056170
 799 056204
 800 056212 013737 060024 056542
 801 056220 013737 060026 056544
 802 056226
 803 056240 012737 056336 060024
 804 056246 012737 000340 060026
 805 056254
 806 056266 012700 172340
 807 056272 012701 136512
 808 056276 012702 000010
 809 056302 012021
 810 056304 077202
 811 056306 013721 172516
 812 056312 013721 177576
 813 056316 013721 177574
 814 056322 013721 177572
 815 056326 104417
 816 056330
 817 056334 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,WOOPEND-WOOPUP/2+12.
MOV #WOOPUP,FIRST+PWRVEC
MOV #340,FIRST+PWRVEC+2
BMOV WOOPUP,FIRST+WOOPUP,WOOPEND-WOOPUP/2
MOV #KIPAR0,R0
MOV #FIRST+WOOPEND,R1
MOV #8,R2
1$: MOV (R0)+,(R1)+
SOB R2,1$
MOV MMR3,(R1)+
MOV MMR2,(R1)+
MOV MMR1,(R1)+
MOV MMR0,(R1)+
KERNEL ;ENTER KERNEL MODE
POP BANK
RETURN
```

820 056336

```

WOOPUP: SUBTST <<POWER UP FROM BANK 0 TO RELOCATION>>
:*****
:*SUBTEST      POWER UP FROM BANK 0 TO RELOCATION
:*****
      MOV      #WOOPEND,R0
      MOV      #KIPAR0,R1
      MOV      #KIPDR0,R3
      MOV      #8,R2
1$:    MOV      (R0)+,(R1)+
      MOV      #77406,(R3)+
      SOB      R2,1$
      MOV      (R0)+,MMR3
      MOV      (R0)+,MMR2
      MOV      (R0)+,MMR1
      MOV      (R0)+,MMR0
      MOV      $SAVR6,SP
      PUSH     BANK
      CLR      BANK
      MAP      BANK
      SUPERVISOR
      :MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      :ENTER SUPERVISOR MODE
      MOV      WOOPSAV,FIRST+PWRVEC
      MOV      WOOPSAV+2,FIRST+PWRVEC+2
      :SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
      :BMOV     WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
      MOV      #WOOPSAV+4,R0
      MOV      #WOOPEND-WOOPUP/2+12.,R1
      MOV      #FIRST+WOOPUP,R2
2$:    MOV      (R0)+,(R2)+
      SOB      R1,2$

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

821 056336 012700 056512
 822 056342 012701 172340
 823 056346 012703 172300
 824 056352 012702 000010
 825 056356 012021
 826 056360 012723 077406
 827 056364 077204
 828 056366 012037 172516
 829 056372 012037 177576
 830 056376 012037 177574
 831 056402 012037 177572
 832 056406 013706 056140
 833 056412
 834 056416 005037 002106
 835 056422
 836 056436
 837 056444 013737 056542 060024
 838 056452 013737 056544 060026
 839
 840
 841 056460 012700 056546
 842 056464 012701 000102
 843 056470 012702 136336
 844 056474 012022
 845 056476 077102
 846
 847 056500 104417
 848 056502
 849 056506 000137 055536
 850 056512 000014
 853 056542 000104

```

858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877 056752 105737 002452
878 056756 100407
879 056760 010046
880 056762 017600 000002
881 056766 112046
882 056770 001005
883 056772 005726
884 056774 012600
885 056776 062716 000002
886 057002 000002
887 057004 122716 000011
888 057010 001002
889 057012 112716 000040
890 057016 122716 000200
891 057022 001006
892 057024 005726
893 057026
894 057030 003010
895 057032 105037 057356
896 057036 000753
897 057040 004737 057076
898 057044 123726 003002
899 057050 001346
900 057052 013746 002450
901
902 057056 105366 000001
903 057062 002770
904 057064 004737 057076
905 057070 105337 057356
906 057074 000770
907 057076
908 057100 116601 000004
909 057104
910 057112 104424
911 057114
912 057122
913 057130 013700 177764
914 057134

```

```

.SBTTL IO SUBROUTINES
.SBTTL ROUTINE TYPE
*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*
$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
BMI 6$ ;;BR IF NO
1$: MOV RO,-(SP) ;;SAVE RO
MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
4$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 7$ ;;BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
5$: MOV (SP)+,RO ;;RESTORE RO
6$: ADD #2,(SP) ;;ADJUST RETURN PC
RTI ;;RETURN
7$: CMPB #HT,(SP) ;;BRANCH IF NOT <HT>
BNE 11$
MOV #',(SP) ;;REPLACE TAB WITH SPACE
11$: CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
BNE 8$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
BR 4$ ;;GET NEXT CHARACTER
8$: CALL $TYPEC ;;GO TYPE THIS CHARACTER
9$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
BNE 4$ ;;IF NO GO GET NEXT CHAR.
MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
10$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
BLT 9$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
CALL $TYPEC ;;GO TYPE A NULL
DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
BR 10$ ;;LOOP
$TYPEC: PUSH R1
MOVB 4(SP),R1
PUSH RO,CONTRL
CACHOFF ;TURN CACHE OFF
IF MULTIPOST IS TRUE
IF NOIIST IS TRUE
MOV SYSTID,RO
ELSE

```

```

915 057136 017700 123574      MOV @IISTACR,R0
916 057142 072027 177770      ASH #-8.,R0
917 057146 042700 177774      BIC #^C3,R0
918 057152                      END :OF IF NOIIST
919 057152                      END :OF IF MULTIPOINT
920 057152                      IF PORTDIR NE R0 AND MULTIPOINT IS TRUE ;IF I,M NOT THE MASTER IN A MULTIPOINT SYSTEM
921 057166                      IF CPUBIT EQ #SLAVE1 THEN LET R0 := #2
922 057202                      IF CPUBIT EQ #SLAVE2 THEN LET R0 := #4
923 057216                      IF CPUBIT EQ #SLAVE3 THEN LET R0 := #6
924 057232                      MAP #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
925 057246                      SUPERVISOR ;ENTER SUPERVISOR MODE
926 057254 005760 062626      2$: TST FIRST+PTYFLAG(R0)
927 057260 100775                      BMI 2$
928 057262 110160 062636      MOV R1,FIRST+PTYBUF(R0)
929 057266                      SET FIRST+PTYFLAG(R0)
930 057274                      MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
931 057310 104417                      KERNEL ;ENTER KERNEL MODE
932 057312                      ELSE ;IF I AM THE MASTER OR WE'RE NOT MULTIPOINTING
933 057314 105777 123456      3$: TSTB @$TPS ;:WAIT UNTIL PRINTER IS READY
934 057320 100375                      BPL 3$
935 057322 110177 123452      MOV R1,$@TPB ;:LOAD CHAR TO BE TYPED INTO DATA REG.
936 057326                      END :OF IF PORTDIR
937 057326 122766 000015 000002      CMPB #CR,2(SP) ;:IS CHARACTER A CARRIAGE RETURN?
938 057334 001003                      BNE 1$ ;:BRANCH IF NO
939 057336 105037 057356      CLRB $CHARCNT ;:YES--CLEAR CHARACTER COUNT
940 057342 000406                      BR $TYPEX ;:EXIT
941 057344 122766 000012 000002      1$: CMPB #LF,2(SP) ;:IS CHARACTER A LINE FEED?
942 057352 001402                      BEQ $TYPEX ;:BRANCH IF YES
943 057354 105227                      INCB (PC)+ ;:COUNT THE CHARACTER
944 057356 000000      $CHARCNT: .WORD 0 ;:CHARACTER COUNT STORAGE
945 057360      $TYPEX: POP CONTRL,R0,R1
946 057370 000207      RETURN
947 057372      SUPLIMIT:;!!!!!!!!!!!!!!!!THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE
    
```

949 057372
 950
 951 057372 104424
 952 057374
 953 057402 012737 177777 002676
 954 057410 013777 002676 123352
 955
 956 057416 013737 002766 002674
 957 057424 012737 000176 002766
 958 057432 017777 123236 123326
 959 057440
 960 057446
 961 057452 005037 002670
 962 057456 162737 000200 002706
 963 057464 010001
 964 057466 006301
 965 057470
 966 057474 010261 002636
 967 057500 012703 000200
 968 057504 077301 1\$:
 969 057506
 970 057516 012703 000200
 971 057522 077301 2\$:
 972 057524 012737 023542 002574
 973 057532 004737 072154 4\$:
 974 057536
 975 057540 010037 002056
 976 057544 104060
 977 057546 005037 003014
 978 057552 012737 002000 002706
 979 057560 000137 060746
 980 057564
 981 057564 005761 002616
 982 057570 100360
 983 057572
 984 057602
 985 057604 013700 002742
 986 057610 012720 061216
 987 057614 012710 000340
 988 057620
 989 057622 005037 002670
 990 057626 162737 000200 002706
 991 057634 012701 000001
 992 057640 072100
 993 057642 004737 061174
 994 057646 012777 000000 123062
 995 057654 010177 123060
 996 057660 012777 000001 123052
 997
 998
 999 057666 012737 061165 002574
 1000 057674 012777 000001 123034 6\$:
 1001 057702
 1002 057704 104061

```

MAST01: SUBTST <<MULTIPOINT SLAVE STARTER>>
:*****
:*SUBTEST MULTIPOINT SLAVE STARTER
:*****
    .ENABL LSB
    CACHOFF ;TURN CACHE OFF
    SET NOSCOPE ;NO SCOPE LOOPS IN MULTIPROCESSOR TESTS
    MOV #-1,MDISPLAY
    MOV MDISPLAY,@DISPLAY
    ;PASS PSEUDO SWR TO SLAVE
    MOV SWR,MASWR ;SAVE REAL SWR
    MOV #SWREG,SWR ;SETUP SOFTWARE SWR
    MOV @MASWR,@SWR ;FILL IT WITH REAL SWR STUFF
    IF NOIIST IS TRUE
        FOR R0 := #1 TO SLAVES
            CLR LOCK
            SUB #200,KSTACK
            MOV R0,R1
            ASL R1
            FOR R2 := #10, DOWNT0 #0
                MOV R2,PTYBUF(R1)
                MOV #200,R3 ;DELAY FOR THE SLAVE TO GAIN ACCESS TO CNT-DOWN
                SOB R3,1$
            END ;OF FOR R2
            MOV #200,R3 ;DELAY FOR THE SLAVE TO SETUP HIS STACK
            SOB R3,2$
            MOV #2*SECONDS,HUNGTIME
            CALL QUE
            ON.ERROR
                MOV R0,BAD
                ERROR +60 ;SLAVE HUNG SYSTEM!
                CLR MULTIPOINT
                MOV #STACK,KSTACK
                JMP MULTIEXIT
            END ;OF ON.ERROR
            TST TASK(R1)
            BPL 4$
        END ;OF FOR R0
    ELSE
        MOV IISTVEC,R0
        MOV #SLAVUP,(R0)+
        MOV #340,(R0)
        FOR R0 := #0 TO #3
            CLR LOCK
            SUB #200,KSTACK
            MOV #1,R1
            ASH R0,R1
            CALL IISTOFF ;INIT IIST & DISABLE BOOTS & INTERRUPTS
            MOV #0,@IISTACR
            MOV R1,@IISTADR ;INTERRUPT SOME CPU
            MOV #1,@IISTADR ;NOW!
        ;WAIT FOR READY FLAG
        MOV #5*SECONDS,HUNGTIME
        MOV #1,@IISTACR
        ON.ERROR
            ERROR +61 ;IIST HUNG SYSTEM!
    
```

```

1003 057706 005037 003014          CLR      MULTIPOINT
1004 057712 012737 002000 002706  MOV      #STACK,KSTACK
1005 057720 000137 060746          JMP      MULTIEXIT
1006 057724          END ;OF ON.ERROR
1007 057724 032777 004000 123006  BIT #BIT11,@IISTADR
1008 057732 001760          BEQ 6$
1009 057734 012703 000200          MOV #200,R3 ;DELAY FOR THE IIST TO REALLY BE READY
1010 057740 077301          SOB R3,3$ ;AND FOR SLAVE TO GO THRU VECTOR AND SETUP HIS STACK
1011 057742          END ;OF FOR R0
1012 057752          FOR R0 := #1 TO SLAVES
1013 057756 010001          MOV R0,R1
1014 057760 006301          ASL R1
1015 057762 012737 061165 002574  MOV #5*SECONDS,HUNGTIME
1016 057770 004737 072154          CALL    QUE
1017 057774          ON.ERROR
1018 057776 010037 002056          MOV      R0,BAD
1019 060002 104060          ERROR   +60 ;SLAVE HUNG SYSTEM!
1020 060004 005037 003014          CLR      MULTIPOINT
1021 060010 012737 002000 002706  MOV      #STACK,KSTACK
1022 060016 000137 060746          JMP      MULTIEXIT
1023 060022          END ;OF ON.ERROR
1024 060022 005761 002616          TST TASK(R1)
1025 060026 100360          BPL 5$
1026 060030          END ;OF FOR R0
1027 060040          END ;OF IF NOIIST
1028 060040 012737 002000 002706  MOV      #STACK,KSTACK
1029          .DSABL LSB
    
```

1032 060046

MAST02: SUBTST <<HAVE SLAVES RUN CSR TESTS>>
 :*****
 :*SUBTEST HAVE SLAVES RUN CSR TESTS
 :*****

1033
 1034 060046 012737 000000 002676
 1035 060054 012777 177776 122706
 1036 060062 005037 002670
 1037 060066
 1038 060072 012701 000020
 1039 060076 072100
 1040 060100 010137 002114
 1041 060104 010001
 1042 060106 006301
 1043 060110 012761 000015 002616
 1044 060116 012737 000031 002572
 1045 060124 012737 177777 002574
 1046 060132 004737 072154
 1047 060136
 1048 060140 162737 000001 002572
 1049 060146
 1050 060150 010037 002056
 1051 060154 012737 000020 002114
 1052 060162 104060
 1053 060164 005037 003014
 1054 060170 000137 060746
 1055 060174
 1056 060174
 1057 060174 005761 002616
 1058 060200 100354
 1059 060202
 1060 060212 012737 000020 002114
 1061

```

    .ENABL  LSB
    MOV     #0,MDISPLAY      ;LET SLAVES SHOW INDIVIDUAL CSR TESTS IN DISPLAY
    MOV     #-2,@DISPLAY    ;FAKE OUT MASTER'S DISPLAY (NOT SLAVES)
    CLR     LOCK
    FOR RO := #1 TO SLAVES
    MOV     #BIT4,R1
    ASH    R0,R1
    MOV    R1,CPUBIT
    MOV    R0,R1
    ASL    R1
    MOV    #15,TASK(R1)      ;RUN CSR TESTS COMMAND
    MOV    #25,HUNGMSB
    MOV    #-1,HUNGTIME     ;13 SECONDS
    1$:    CALL  QUE
    ON.ERROR
    SUB    #1,HUNGMSB
    ON.ERROR
    MOV    R0,BAD
    MOV    #MASTER,CPUBIT
    ERROR  +60              ;SLAVE HUNG SYSTEM!
    CLR    MULTIPOINT
    JMP    MULTIEXIT
    END ;OF ON.ERROR
    END ;OF ON.ERROR
    TST   TASK(R1)
    BPL   1$
    END ;OF FOR RO
    MOV    #MASTER,CPUBIT   ;SET UP AS MASTER AGAIN
    .DSABL LSB
    
```

1064 060220

MAST03: SUBTST <<MOVE SLAVES TO THERE OWN BANKS>>
 :*****
 :*SUBTEST MOVE SLAVES TO THERE OWN BANKS
 :*****

1065
 1066 060220 012737 177775 002676
 1067 060226 013777 002676 122534
 1068 060234 005037 002670
 1069 060240
 1070 060244 012701 000020
 1071 060250 072100
 1072 060252 010137 002114
 1073 060256
 1074 060256
 1075 060264 004737 042422
 1076 060270
 1077 060312 104511
 1078 060314 010001
 1079 060316 006301
 1080 060320 012761 000001 002616
 1081 060326 012737 061165 002574
 1082 060334 004737 072154 1\$:
 1083 060340
 1084 060342 010037 002056
 1085 060346 012737 000020 002114
 1086 060354 104060
 1087 060356 005037 003014
 1088 060362 000137 060746
 1089 060366
 1090 060366 005761 002616
 1091 060372 100360
 1092 060374 013701 002110
 1093 060400 052761 100000 003016
 1094 060406 042761 020000 003020
 1095 060414
 1096 060416
 1097 060416
 1098 060432 010037 002056
 1099 060436 104062
 1100
 1101 060440 005037 003014
 1102 060444 000137 060746
 1103 060450
 1104 060450
 1105 060460 012737 002000 002706
 1106
 1107 060466 013737 002674 002766
 1108

```

.ENABL  LSB
MOV     #-3,MDISPLAY
MOV     MDISPLAY,@DISPLAY
CLR     LOCK
FOR RO := #1 TO SLAVES
  MOV   #BIT4,R1
  ASH  RO,R1
  MOV  R1,CPUBIT
  BEGIN MOVELOOP
    FOR BANK := #1 TO #167
      CALL EXBANK
      IF PFLAG IS FALSE AND ACFLAG IS TRUE AND BMFLAG IS FALSE
        INVALIDATE
        MOV   RO,R1
        ASL   R1           ;R1 = 2 OR 4 OR 6
        MOV  #1,TASK(R1)  ;MOVE COMMAND
        MOV  #5*SECONDS,HUNGTIME
        CALL QUE
        ON.ERROR
          MOV  RO,BAD
          MOV  #MASTER,CPUBIT
          ERROR +60      ;SLAVE HUNG SYSTEM!
          CLR  MULTIPOINT
          JMP  MULTIEXIT
        END ;OF ON.ERROR
        TST  TASK(R1)
        BPL  1$
        MOV  BANKINDEX,R1
        BIS  #BIT15,CONFIG(R1)
        BIC  #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
        LEAVE MOVELOOP
      END ;OF IF PFLAG
    END ;OF FOR BANK
    MOV  RO,BAD
    ERROR +62      ;NOT ENOUGH GOOD MEMORY TO TEST SLAVES
                  ;ABORTING MULTIPROCESSOR TESTS
  CLR  MULTIPOINT
  JMP  MULTIEXIT
END MOVELOOP
END ;OF FOR RO
MOV  #STACK,KSTACK
;POINT THE MASTER BACK TO THE NORMAL SWR
MOV  MASWR,SWR
.DSABL  LSB
    
```

1111 060474

SUBTST <<RUN NORMAL MULTIPROCESSOR TESTS>>

*SUBTEST RUN NORMAL MULTIPROCESSOR TESTS

1112 060474 012737 000020 002114
1113 060502 052737 000020 002510
1114 060510
1115 060514 006337 002510
1116 060520 052737 000020 002510
1117 060526
1118 060536 004737 061322
1119 060542
1120 060544 005037 003014
1121 060550 000137 060746
1122 060554
1123 060554 004737 061706
1124 060560
1125 060562 005037 003014
1126 060566 000137 060746
1127 060572
1128 060572 004737 062270
1129 060576
1130 060600 005037 003014
1131 060604 000137 060746
1132 060610
1133 060610 004737 062550
1134 060614
1135 060616 005037 003014
1136 060622 000137 060746
1137 060626
1138 060626 004737 062776
1139 060632
1140 060634 005037 003014
1141 060640 000137 060746
1142 060644
1143 060644 004737 063260
1144 060650
1145 060652 005037 003014
1146 060656 000137 060746
1147 060662
1148 060662 004737 063732
1149 060666
1150 060670 005037 003014
1151 060674 000137 060746
1152 060700
1153 060700 004737 064200
1154 060704
1155 060706 005037 003014
1156 060712 000137 060746
1157 060716
1158 060716 004737 064204
1159 060722
1160 060724 005037 003014
1161 060730 000137 060746
1162 060734
1163 060734 004737 064472
1164 060740

MCV #MASTER, CPUBIT
BIS #BIT4, ALLCPUS
FOR R0 := #1 TO SLAVES
ASL ALLCPUS
BIS #BIT4, ALLCPUS
END ;OF FOR R0
CALL MAST04 ;ACCESS PATHS TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST05 ;ADDRESS TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST06 ;ASYNCHRONOUS ADDRESS UNIQUENESS TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST07 ;SKEWED UP ADDRESS TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST10 ;PORT ARBITRATION SYMMETRY TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST11 ;I/O PRIORITY TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST12 ;EXECUTION CONTENTION TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST13 ;ASRB TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST14 ;CACHE FLUSH TEST
ON.ERROR
CLR MULTIPOINT
JMP MULTIEXIT
END ;OF ON.ERROR
CALL MAST15 ;STOP SLAVES
ON.ERROR

```

1165 060742 005037 003014          CLR  MULTIPOINT
1166 060746                                END ;OF ON.ERROR
1167 060746          MULTIEXIT:FOR BANK := #1 TO #167
1168 060754 004737 042422          CALL  EXBANK
1169 060760 013701 002110          MOV   BANKINDEX,R1
1170 060764 042761 100000 003016  BIC   #BIT15,CONFIG(R1)
1171 060772                                END ;OF FOR BANK
1172
1173                                ;MARK THE LOADERS AS PROTECTED AGAIN
1174 061006 013737 002710 002106  MOV   LOADHOME,BANK
1175 061014 004737 042422          CALL  EXBANK
1176 061020 013701 002110          MOV   BANKINDEX,R1
1177 061024 052761 100000 003016  BIS   #BIT15,CONFIG(R1)
1178
1179 061032 005037 002676          CLR   MDISPLAY
1180 061036 005077 121726          CLR   @DISPLAY
1181 061042                                SET   WRITEONLY
1182 061050 004737 015104          CALL  DOBACK ;REWRITE BACKGROUND PATTERN
1183 061054                                CLEAR  NOSCOPE,LOCK,WRITEONLY
1184 061070 000137 014656          JMP   FLUSH

```

1187 061074

```
IISTINIT:SUBTST <<INITIALIZE THE IIST>>
:*****
:*SUBTEST INITIALIZE THE IIST
:*****
```

1188
 1189 061074 012777 100000 121634
 1190 061102 012777 000005 121626
 1191 061110 013777 002726 121622
 1192 061116 013777 002726 121614
 1193 061124 012737 061165 002574
 1194 061132 012777 000001 121576
 1195 061140
 1196 061150
 1197 061152 104061
 1198 061154 005037 003014
 1199 061160 000207
 1200 061162
 1201 061162
 1202 061162 032777 004000 121550
 1203 061170 001760
 1204 061172 000207
 1205
 1206
 1207 061174

```
.ENABL LSB
MOV #BIT15,@IISTACR
MOV #5,@IISTACR
MOV ONES,@IISTADR
MOV ONES,@IISTADR
MOV #5*SECONDS,HUNGTIME
2$: MOV #1,@IISTACR
IF CPUBIT EQ #MASTER
ON.ERROR
ERROR +61 ;IIST HUNG SYSTEM!
CLR MULTIPOINT
RETURN
END ;OF ON.ERROR
END ;OF IF CPUBIT
BIT #BIT11,@IISTADR
BEQ 2$
RETURN
.DSABL LSB
```

```
IISTOFF:SUBTST <<INIT IIST & DISABLE BOOTS & INTERRUPTS>>
:*****
:*SUBTEST INIT IIST & DISABLE BOOTS & INTERRUPTS
:*****
```

1208 061174 004737 061074
 1209 061200 012777 000004 121530
 1210 061206 013777 002726 121524
 1211 061214 000207
 1212
 1213 061216 013706 002706
 1214 061222 004737 061244
 1215 061226 104420
 1216 061230 000237
 1217 061232 052763 100000 002616
 1218 061240 000137 050726

```
CALL IISTINIT
MOV #4,@IISTACR
MOV ONES,@IISTADR
RETURN
SLAVUP: MOV KSTACK,SP
CALL SLAVEMAP
ENERGIZE ;TURN ON MEMORY MANAGEMENT
SPL 7
BIS #BIT15,TASK(R3)
JMP TASKDO
```

1221 061244

SLAVEMAP:SUBTST <<SLAVES MAP>>

: *SUBTEST SLAVES MAP

1222

:MAP KERNEL

1223 061244 104415
1224 061246 012700 002550
1225 061252 012701 172340
1226 061256 012702 077406
1227 061262 012703 172300
1228 061266 012705 000006
1229 061272 012021
1230 061274 010223
1231 061276 077503
1232 061300 012721 177400
1233 061304 010223
1234 061306 012711 177600
1235 061312 010213
1236 061314 104420
1237 061316 104416
1238 061320 000207

SAVREG
MOV #KIO,R0
MOV #KIPAR0,R1
MOV #77406,R2
MOV #KIPDR0,R3
MOV #6,R5
1\$: MOV (R0)+,(R1)+
MOV R2,(R3)+
SOB R5,1\$
MOV #177400,(R1)+
MOV R2,(R3)+
MOV #177600,(R1)
MOV R2,(R3)
ENERGIZE
RESREG
RETURN

1241 061322

MAST04: SUBTST <<MULTIPOINT ACCESS PATHS TEST>>
 :*****
 :*SUBTEST MULTIPOINT ACCESS PATHS TEST
 :*****

1242
 1243 061322 012737 177774 002676
 1244 061330 013777 002676 121432
 1245 061336
 1246 061340 010001
 1247 061342 006301
 1248 061344
 1249 061352 013737 002510 002114
 1250 061360 004737 042422
 1251 061364 012737 000020 002114
 1252 061372
 1253 061406 104511
 1254 061410
 1255 061412 010502
 1256 061414 000302
 1257 061416 062702 000002
 1258 061422 010261 002616
 1259 061426
 1260 061432
 1261 061442 004737 050726
 1262 061446
 1263 061456
 1264 061456 012737 061165 002574
 1265 061464 004737 072154
 1266 061470
 1267 061472 010037 002056
 1268 061476 104060
 1269 061500
 1270 061504
 1271 061504 005761 002616
 1272 061510 100365
 1273 061512 062702 000001
 1274 061516
 1275 061520 010304
 1276 061522 006304
 1277 061524 010264 002616
 1278 061530
 1279 061540
 1280 061552 004737 050726
 1281 061556
 1282 061570
 1283 061572 010304
 1284 061574 006304
 1285 061576
 1286 061602 012737 061165 002574
 1287 061610 004737 072154
 1288 061614
 1289 061616 010337 002056
 1290 061622 104060
 1291 061624
 1292 061630
 1293 061630 005764 002616
 1294 061634 100365

```

    .ENABL LSB
    MOV #-4,MDISPLAY
    MOV MDISPLAY,@DISPLAY
    FOR R0 := #0 TO SLAVES
    MOV R0,R1
    ASL R1
    FOR BANK := #1 TO #167
    MOV ALLCPUS,CPUBIT
    CALL EXBANK
    MOV #MASTER,CPUBIT
    IF PFLAG IS FALSE AND ACFLAG IS TRUE
    INVALIDATE
    FOR R5 := #0 TO #5 ;FOR PATTERNS 0-5
    MOV R5,R2
    SWAB R2
    ADD #2,R2 ;WRITE ONLY PATTERN COMMAND
    MOV R2,TASK(R1) ;SET TASK ORDER
    IF R1 EQ #0
    PUSH R0,R1,R2,R5
    CALL TASKDO
    POP R5,R2,R1,R0
    END ;OF IF R1
    MOV #5*SECONDS,HUNGTIME
    CALL QUE
    ON.ERROR
    MOV R0,BAD
    ERROR +60 ;SLAVE HUNG SYSTEM!
    $RETURN ERROR
    END ;OF ON.ERROR
    TST TASK(R1)
    BPL 1$
    ADD #1,R2 ;CHANGE ORDER TO READ ONLY
    FOR R3 := #0 TO SLAVES
    MOV R3,R4
    ASL R4
    MOV R2,TASK(R4)
    END ;OF FOR R3
    PUSH R0,R1,R3,R4,R5
    CALL TASKDO
    POP R5,R4,R3,R1,R0
    FOR R3 := #0 TO SLAVES
    MOV R3,R4
    ASL R4
    IF R4 NE R1
    MOV #5*SECONDS,HUNGTIME
    CALL QUE
    ON.ERROR
    MOV R3,BAD
    ERROR +60 ;SLAVE HUNG SYSTEM!
    $RETURN ERROR
    END ;OF ON.ERROR
    TST TASK(R4)
    BPL 2$
    
```

1\$:

2\$:

1295 061636
1296 061636
1297 061646
1298 061656
1299 061656
1300 061672
1301 061702
1302

END :OF IF R4
END :OF FOR R3
END :OF FOR R5
END :OF IF PFLAG
END :OF FOR BANK
END :OF FOR R0
\$RETURN NOERROR
.DSABL LSB

1306 061706

MAST05: SUBTST <<MULTIPOINT ADDRESS TEST>>

 :*SUBTEST MULTIPOINT ADDRESS TEST

1307
 1308 061706 012737 177773 002676
 1309 061714 013777 002676 121046
 1310 061722
 1311 061724 010001
 1312 061726 006301
 1313 061730
 1314 061736 013737 002510 002114
 1315 061744 004737 042422
 1316 061750 012737 000020 002114
 1317 061756
 1318 061772
 1319 061774 010502
 1320 061776 000302
 1321 062000 062702 000007
 1322 062004 010261 002616
 1323 062010
 1324 062014
 1325 062024 004737 050726
 1326 062030
 1327 062040
 1328 062040 012737 061165 002574
 1329 062046 004737 072154
 1330 062052
 1331 062054 010037 002056
 1332 062060 104060
 1333 062062
 1334 062066
 1335 062066 005761 002616
 1336 062072 100365
 1337 062074 062702 000001
 1338 062100
 1339 062102 010304
 1340 062104 006304
 1341 062106 010264 002616
 1342 062112
 1343 062122
 1344 062134 004737 050726
 1345 062140
 1346 062152
 1347 062154 010304
 1348 062156 006304
 1349 062160
 1350 062164 012737 061165 002574
 1351 062172 004737 072154
 1352 062176
 1353 062200 010337 002056
 1354 062204 104060
 1355 062206
 1356 062212
 1357 062212 005764 002616
 1358 062216 100365
 1359 062220

```

.ENABL  LSB
MOV     #-5,MDISPLAY
MOV     MDISPLAY,@DISPLAY
FOR R0 := #0 TO SLAVES
  MOV   R0,R1
  ASL  R1
  FOR BANK := #1 TO #167
    MOV ALLCPUS,CPUBIT
    CALL EXBANK
    MOV #MASTER,CPUBIT
    IF PFLAG IS FALSE AND ACFLAG IS TRUE
      FOR R5 := #0 TO #1 ;FOR ADDRESS & COMPLEMENT ADDRESS PATTERNS
        MOV   R5,R2
        SWAB  R2
        ADD   #7,R2           ;WRITE ONLY ADDRESS PATTERN
        MOV   R2,TASK(R1)    ;SET TASK ORDER
        IF R1 EQ #0
          PUSH R0,R1,R2,R5
          CALL TASKDO
          POP  R5,R2,R1,R0
        END ;OF IF R1
        MOV   #5*SECONDS,HUNGTIME
        CALL  QUE
        ON.ERROR
          MOV   R0,BAD
          ERROR +60           ;SLAVE HUNG SYSTEM!
          $RETURN ERROR
        END ;OF ON.ERROR
        TST   TASK(R1)
        BPL  1$
        ADD   #1,R2           ;CHANGE ORDER TO READ ADDRESS PATTERNS ONLY
        FOR R3 := #0 TO SLAVES
          MOV   R3,R4
          ASL  R4
          MOV   R2,TASK(R4)
        END ;OF FOR R3
        PUSH  R0,R1,R3,R4,R5
        CALL  TASKDO
        POP   R5,R4,R3,R1,R0
        FOR R3 := #0 TO SLAVES
          MOV   R3,R4
          ASL  R4
          IF R4 NE R1
            MOV #5*SECONDS,HUNGTIME
            CALL  QUE
            ON.ERROR
              MOV   R3,BAD
              ERROR +60           ;SLAVE HUNG SYSTEM!
              $RETURN ERROR
            END ;OF ON.ERROR
            TST  TASK(R4)
            BPL  2$
          END ;OF IF R4
    END ;OF FOR BANK
  END ;OF FOR R0

```

1\$:

2\$:

1360 062220
1361 062230
1362 062240
1363 062240
1364 062254
1365 062264
1366

END :OF FOR R3
END :OF FOR R5
END :OF IF PFLAG
END :OF FOR BANK
END :OF FOR R0
\$RETURN NOERROR
.DSABL LSB

```

1369 062270
1370
1371 062270 012737 177772 002676
1372 062276 013777 002676 120464
1373 062304
1374 062312 013737 002510 002114
1375 062320 004737 042422
1376 062324 012737 000020 002114
1377 062332
1378 062346
1379 062350 010003
1380 062352 072327 000015
1381 062356 062703 060000
1382 062362
1383 062364 062703 020000
1384 062370
1385 062402
1386 062404 000241
1387 062406 006003
1388 062410 010201
1389 062412 006301
1390 062414 062703 000011
1391 062420 010361 002616
1392 062424
1393 062426
1394 062436
1395 062440 004737 050726
1396 062444
1397 062446
1398 062450 010201
1399 062452 006301
1400 062454 012737 061165 002574
1401 062462 004737 072154 1$:
1402 062466
1403 062470 010237 002056
1404 062474 104060
1405 062476
1406 062502
1407 062502 005761 002616
1408 062506 100365
1409 062510
1410 062520
1411 062530
1412 062530
1413 062544
1414

```

```

MAST06: SUBTST <<MULTIPOINT ASYNCHRONOUS ADDRESS UNIQUENESS TEST>>
:*****
:*SUBTEST MULTIPOINT ASYNCHRONOUS ADDRESS UNIQUENESS TEST
:*****
.ENABL LSB
MOV #-6,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR BANK := #1 TO #167
MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE
FOR R0 := #0 TO #3 ;R0 := STARTING ADDRESS INDEX
MOV R0,R3
ASH #13,R3
ADD #FIRST,R3
FOR R2 := #0 TO SLAVES
ADD #SIZE/2,R3
IF R3 EQ #LAST+2 THEN LET R3 := #FIRST
PUSH R3
CLC
ROR R3
MOV R2,R1 ;R1 := SLAVE #
ASL R1
ADD #11,R3 ;ORDER FOR ASYNCHRONOUS ADD UNIQUE TEST
MOV R3,TASK(R1)
POP R3
END ;OF FOR R2
PUSH R0 ;MASTER ASSUMES THE SLAVES ROLE
CALL TASKDO
POP R0
FOR R2 := #0 TO SLAVES
MOV R2,R1
ASL R1
MOV #5*SECONDS,HUNGTIME
CALL QUE
ON.ERROR
MOV R2,BAD ;SLAVE HUNG SYSTEM!
ERROR +60
$RETURN ERROR
END ;OF ON.ERROR
TST TASK(R1)
BPL 1$
END ;OF FOR R2
END ;OF FOR R0
END ;OF IF PFLAG
END ;OF FOR BANK
$RETURN NOERROR
.DSABL LSB

```

```

1417 062550
1418
1419 062550 012737 177771 002676
1420 062556 013777 002676 120204
1421 062564
1422 062572 013737 002510 002114
1423 062600 004737 042422
1424 062604 012737 000020 002114
1425 062612
1426 062626
1427 062630
1428 062632 010001
1429 062634 006301
1430 062636 010203
1431 062640 000303
1432 062642 062703 000012
1433 062646 010361 002616
1434 062652
1435 062662
1436 062664 004737 050726
1437 062670
1438 062672
1439 062674 010001
1440 062676 006301
1441 062700 012737 061165 002574
1442 062706 004737 072154
1443 062712
1444 062714 010037 002056
1445 062720 104060
1446 062722
1447 062726
1448 062726 005761 002616
1449 062732 100365
1450 062734
1451 062744
1452 062756
1453 062756
1454 062772
1455

```

```

MAST07: SUBTST <<MULTIPOINT SKEWED UP ADDRESS TEST>>
:*****
:*SUBTEST MULTIPOINT SKEWED UP ADDRESS TEST
:*****
.ENABL LSB
MOV #-7,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR BANK := #1 TO #167
MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE
FOR R2 := #0 TO #6 BY #2
FOR R0 := #0 TO SLAVES
MOV R0,R1
ASL R1
MOV R2,R3 ;ADDRESS OFFSET (0,2,4,6)
SWAB R3
ADD #12,R3
MOV R3,TASK(R1) ;ORDER SKEWED UP ADDRESS TEST
END ;OF FOR R0
PUSH R2
CALL TASKDO
POP R2
FOR R0 := #0 TO SLAVES
MOV R0,R1
ASL R1
MOV #5*SECONDS,HUNGTIME
CALL QUE
ON.ERROR
MOV R0,BAD ;SLAVE HUNG SYSTEM!
ERROR +60
$RETURN ERROR
END ;OF ON.ERROR
TST TASK(R1)
BPL 1$
END ;OF FOR R0
END ;OF FOR R2
END ;OF IF PFLAG
END ;OF FOR BANK
$RETURN NOERROR
.DSABL LSB

```

1\$:

1458 062776

MAST10: SUBST <<MULTIPOINT PORT ARBITRATION SYMMETRY TEST>>
:*****
:*SUBTEST MULTIPOINT PORT ARBITRATION SYMMETRY TEST
:*****

1459

1460 062776
1461 063010 012737 177770 002676
1462 063016 013777 002676 117744
1463 063024
1464 063032 013737 002510 002114
1465 063040 004737 042422
1466 063044 012737 000020 002114
1467 063052
1468 063066
1469 063070 010001
1470 063072 006301
1471 063074 012761 000405 002616
1472 063102
1473 063112 004737 050726
1474 063116
1475 063120 010001
1476 063122 006301
1477 063124 012737 061165 002574
1478 063132 004737 072154 1\$:
1479 063136
1480 063140 010037 002056
1481 063144 104060
1482 063146
1483 063152
1484 063152 005761 002616
1485 063156 100365
1486 063160
1487 063170
1488 063170
1489 063172 010001
1490 063174 006301
1491 063176 013702 002646
1492 063202 166102 002646
1493 063206 062702 000006
1494 063212
1495 063224 104036
1496 063226
1497 063230
1498 063230
1499 063240
1500 063240
1501 063240
1502 063254
1503

.ENABL LSB
IF NOCLOCK IS TRUE THEN \$RETURN NOERROR
MOV #-10,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR BANK := #1 TO #167
MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE
FOR R0 := #0 TO SLAVES
MOV R0,R1
ASL R1
MOV #405,TASK(R1) ;ORDERS FOR PORT ARBITRATION SYMMETRY TEST
END ;OF FOR R0
CALL TASKDO
FOR R0 := #0 TO SLAVES
MOV R0,R1
ASL R1
MOV #5*SECONDS,HUNGTIME
CALL QUE
ON ERROR
MOV R0,BAD
ERROR +60 ;SLAVE HUNG SYSTEM!
\$RETURN ERROR
END ;OF ON ERROR
TST TASK(R1)
BPL 1\$
END ;OF FOR R0
BEGIN BALANCE
FOR R0 := #0 TO SLAVES
MOV R0,R1
ASL R1
MOV PORTCOUNT,R2
SUB PORTCOUNT(R1),R2
ADD #6,R2 ;SPECKED FOR + OR - 6 = 5%
IF R2 LT #0 OR R2 GT #12.
ERROR +36
LEAVE BALANCE
END ;OF IF R2
END ;OF FOR R0
END BALANCE
END ;OF IF PFLAG
END ;OF FOR BANK
\$RETURN NOERROR
.DSABL LSB

1506 063260

1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518 063260
1519
1520
1521 063272
1522
1523 063306 012737 177767 002676
1524 063314 013777 002676 117446

```
MAST11: SUBST <<MULTIPOINT PORT I/O PRIORITY TEST>>  
:*****  
:*SUBTEST      MULTIPOINT PORT I/O PRIORITY TEST  
:*****  
:THIS TESTS THE ABILITY OF THE MEMORY PORT TO GIVE I/O HIGHER  
:PRIORITY THEN CPU ACCESSES  
:  
:IT DOES THIS BY USING MARGIN #6 IN THE CPU'S MAINT REGISTER  
:TO RAISE THE CPU PRIORITY TO THAT OF I/O.  
:  
:EACH CPU (EXCEPT ONE) HAS HIS PRIORITY RAISED AND  
:A TEST IS PERFORMED TO INSURE THAT THESE CPUS ARE GRANTED MORE  
:CYCLES THEN THE OTHER CPU.  
:  
:NEED LINE CLOCK  
IF NOCLOCK IS TRUE THEN $RETURN NOERROR  
:  
:CAN'T DO UNLESS 4 CPU'S  
IF SLAVES LT #3 THEN $RETURN NOERROR  
  
MOV    #-11,MDISPLAY  
MOV    MDISPLAY,@DISPLAY
```

```

1527
1528 063322
1529 063330 013737 002510 002114
1530 063336 004737 042422
1531 063342 012737 000020 002114
1532 063350 013701 002110
1533 063354
1534 063364
1535
1536
1537 063400
1538 063404
1539 063410
1540 063420 013701 002604
1541 063424 006301
1542 063426 012761 000016 002616
1543 063434
1544 063442 004737 050726
1545 063446
1546 063450 004737 063674
1547 063454
1548 063454
1549 063454
1550
1551
1552 063470
1553 063472 010001
1554 063474 006301
1555 063476 012761 000005 002616
1556 063504
1557 063514 004737 050726
1558
1559
1560 063520
1561 063522 010001
1562 063524 006301
1563 063526 004737 063674
1564 063532
1565
1566
1567 063542 004737 063602
1568 063546
1569 063562
1570 063562
1571 063562
1572 063576

```

```

;FOR EACH BANK THAT IS ACCESSABLE BY ALL CPU'S
FOR BANK := #1 TO #167
MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
MOV BANKINDEX,R1
IF #BIT3 OFF.IN CONFIG(R1) ;IF NO EXTERNAL INTERLEAVE
IF PFLAG IS FALSE AND ACFLAG IS TRUE

;EACH CPU HAS LOW PRIORITY ONCE
FOR I := #0 TO SLAVES
FOR J := #0 TO SLAVES
IF I NE J ;IF NOT THE LOW PRIORITY CPU
MOV J,R1
ASL R1
MOV #16,TASK(R1) ;ORDER MARGIN #6
IF J EQ #0
CALL TASKDO
ELSE
CALL WAIT5
END ;OF IF J EQ #0
END ;OF IF I NE J
END ;OF FOR J := #0 TO SLAVES

;EVERY CPU WILL DO PORT ARBITRATION TEST
FOR RO := #0 TO SLAVES
MOV RO,R1
ASL R1
MOV #5,TASK(R1) ;ORDER PORT ARBITRATION TEST
END ;OF FOR RO
CALL TASKDO

;WAIT FOR DONE
FOR RO := #0 TO SLAVES
MOV RO,R1
ASL R1
CALL WAIT5
END ;OF FOR RO

;CHECK RESULTS FOR PROPER SKEWING
CALL MST11A
END ;OF FOR I := #0 TO SLAVES
END ;OF IF PFLAG
END ;OF IF #BIT3 OFF.IN CONFIG(R1)
END ;OF FOR BANK
$RETURN NOERROR

```

1575 063602
1576 063602
1577
1578 063602
1579 063604 010001
1580 063606 006301
1581 063610 013703 002602
1582 063614 006303
1583 063616 016302 002646
1584 063622 016104 002646
1585 063626 160402
1586 063630 006204
1587 063632 006204
1588 063634 006204
1589 063636
1590 063644
1591 063650 013737 002602 002056
1592 063656 104066
1593 063660
1594 063662
1595 063662
1596 063662
1597 063672
1598 063672 000207
1599
1600 063674

1601
1602 063674 012737 061165 002574
1603 063702 004737 072154
1604 063706
1605 063710 010037 002056
1606 063714 104060
1607 063716
1608 063722
1609 063722 005761 002616
1610 063726 100365
1611 063730 000207
1612

```

MST11A: SUBTST <<CHECK FOR PROPER SKEWING OF ACCESSES>>
:*****
:*SUBTEST      CHECK FOR PROPER SKEWING OF ACCESSES
:*****
      BEGIN NOBALANCE
      :FOR EACH CPU
      FOR R0 := #0 TO SLAVES
      MOV R0,R1
      ASL R1
      MOV I,R3
      ASL R3
      MOV PORTCOUNT(R3),R2      ;LOW PRIORITY PORT
      MOV PORTCOUNT(R1),R4      ;HIGH PRIORITY PORT
      SUB R4,R2
      ASR R4
      ASR R4
      ASR R4
      :R4 = 12.5% OF HIGH PRIORITY PORT
      :IF THE CPU SELECTED IS NOT THE LOW PRIORITY ONE
      IF R0 NE I
      IF R2 LT R4
      MOV I,BAD
      ERROR +66
      LEAVE NOBALANCE
      END :OF IF R2
      END :OF IF R0 NE I
      END :OF FOR R0
      END NOBALANCE
      RETURN

WAITS: SUBTST <<WAIT FOR 5 SECONDS FOR A SLAVE>>
:*****
:*SUBTEST      WAIT FOR 5 SECONDS FOR A SLAVE
:*****
      .ENABL LSB
      MOV #5*SECONDS,HUNGTIME
1$: CALL QUE
      ON.ERROR
      MOV R0,BAD
      ERROR +60      ;SLAVE HUNG SYSTEM!
      $RETURN ERROR
      END :OF ON.ERROR
      TST TASK(R1)
      BPL 1$
      RETURN
      .DSABL LSB
    
```

1615 063732

MAST12: SUBTST <<MULTIPOINT EXECUTION CONTENTION TEST>>

1616
1617 063732 012737 177766 002676
1618 063740 013777 002676 117022
1619 063746
1620 063754 013737 002510 002114
1621 063762 004737 042422
1622 063766 012737 000020 002114
1623 063774
1624 064010 004737 053640
1625 064014
1626 064016 010001
1627 064020 006301
1628 064022 005061 002636
1629 064026 012761 000004 002616
1630 064034
1631 064044 004737 050726
1632 064050
1633 064052 010001
1634 064054 006301
1635 064056 012737 061165 002574
1636 064064 004737 072154 1\$:
1637 064070
1638 064072 010037 002056
1639 064076 104060
1640 064100
1641 064104
1642 064104 005761 002616
1643 064110 100365
1644 064112
1645 064122
1646 064122
1647 064124 010001
1648 064126 006301
1649 064130 013702 002636
1650 064134 166102 002636
1651 064140
1652 064144 104052
1653 064146
1654 064150
1655 064150
1656 064160
1657 064160
1658 064160
1659 064174
1660
1661
1662 064200

:*****
: *SUBTEST MULTIPOINT EXECUTION CONTENTION TEST
:*****

```
.ENABL LSB
MOV #-12,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR BANK := #1 TO #167
  MOV ALLCPUS,CPUBIT
  CALL EXBANK
  MOV #MASTER,CPUBIT
  IF PFLAG IS FALSE AND ACFLAG IS TRUE
    CALL MRUNSETUP
    FOR RO := #0 TO SLAVES
      MOV RO,R1
      ASL R1
      CLR PTYBUF(R1) ;CLEAR INC COUNTER
      MOV #4,TASK(R1) ;ORDER FOR EXECUTION CONTENTION TEST
    END ;OF FOR RO
    CALL TASKDO
    FOR RO := #0 TO SLAVES
      MOV RO,R1
      ASL R1
      MOV #5*SECONDS,HUNGTIME
      CALL QUE
      ON ERROR
        MOV RO,BAD
        ERROR +60 ;SLAVE HUNG SYSTEM!
        $RETURN ERROR
    END ;OF ON ERROR
    TST TASK(R1)
    BPL 1$
  END ;OF FOR RO
  BEGIN RUNCOUNT
  FOR RO := #0 TO SLAVES
    MOV RO,R1
    ASL R1
    MOV PTYBUF,R2
    SUB PTYBUF(R1),R2
    IF R2 NE #0
      ERROR +52
      LEAVE RUNCOUNT
    END ;OF IF R2
  END ;OF FOR RO
  END RUNCOUNT
END ;OF IF PFLAG
END ;OF FOR BANK
$RETURN NOERROR
.DSABL LSB
```

MAST13: SUBTST <<MULTIPOINT ASRB TEST>>

:*****
: *SUBTEST MULTIPOINT ASRB TEST
:*****

1663 064200 000137 053752

```
JMP SUPM13 ;ACTUAL ROUTINE MUST BE IN SUPERVISOR SPACE!
```

1666 064204

MAST14: SUBTST <<MULTI PORT CACHE FLUSH TEST>>

:SUBTEST MULTI PORT CACHE FLUSH TEST

1667

:WARNING FAILURES HERE ARE USUALLY DUE TO A CACHE PROBLEM

1668

:NOT A MAIN MEMORY PROBLEM!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

1669

.ENABL LSB

1670 064204

IF #BIT3 SET.IN CACHK OR #BIT2 SET.IN CACHK

1671 064224

\$RETURN NOERROR ;OPERATOR HAS DISABLED PART OF CACHE

1672 064230

END ;OF IF #BIT3

1673

1674 064230 012737 177764 002676

MOV #-14,MDISPLAY

1675 064236 013777 002676 116524

MOV MDISPLAY,@DISPLAY

1676

FOR R0 := #0 TO SLAVES

1677 064244

MOV R0,R1

1678 064246 010001

ASL R1

1679 064250 006301

FOR BANK := #1 TO #167

1680 064252

MOV ALLCPUS,CPUBIT

1681 064260 013737 002510 002114

CALL EXBANK

1682 064266 004737 042422

MOV #MASTER,CPUBIT

1683 064272 012737 000020 002114

IF PFLAG IS FALSE AND ACFLAG IS TRUE

1684 064300

FOR R3 := #0 TO SLAVES

1685 064314

MOV R3,R4

1686 064316 010304

ASL R4

1687 064320 006304

IF R4 EQ R1

1688 064322

MOV #20,TASK(R4) ;CACHE FLUSH TEST COMMAND

1689 064326 012764 000020 002616

ELSE

1690 064334

MOV #17,TASK(R4) ;CREATE STALE DATA COMMAND

1691 064336 012764 000017 002616

END ;OF IF R4

1692 064344

END ;OF FOR R3

1693 064344

PUSH R0,R1

1694 064354

CALL TASKDO

1695 064360 004737 050726

POP R1,R0

1696 064364

FOR R3 := #0 TO SLAVES

1697 064370

MOV R3,R4

1698 064372 010304

ASL R4

1699 064374 006304

MOV #5*SECONDS,HUNGTIME

1700 064376 012737 061165 002574

CALL QUE

1701 064404 004737 072154 2\$:

ON.ERROR

1702 064410

MOV R3,BAD

1703 064412 010337 002056

ERROR +60

1704 064416 104060

\$RETURN ERROR

;SLAVE HUNG SYSTEM!

1705 064420

END ;OF ON.ERROR

1706 064424

TST TASK(R4)

1707 064424 005764 002616

BPL 2\$

1708 064430 100365

END ;OF FOR R3

1709 064432

END ;OF IF PFLAG

1710 064442

END ;OF FOR BANK

1711 064442

END ;OF FOR R0

1712 064456

\$RETURN NOERROR

1713 064466

.DSABL LSB

1714

1717 064472

MAST15: SUBTST <<STOP SLAVES>>

:*SUBTEST STOP SLAVES

1718

1719 064472 012737 177763 002676

1720 064500 013777 002676 116262

1721 064506

1722 064512 010001

1723 064514 006301

1724 064516 012761 000006 002616

1725 064524 012737 023542 002574

1726 064532 004737 072154

1727 064536

1728 064540 010037 002056

1729 064544 104060

1730 064546

1731 064552

1732 064552 005761 002616

1733 064556 100365

1734 064560

1735 064570

1736

1\$:

```
.ENABL LSB
MOV #-15,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR RO := #1 TO SLAVES
  MOV RO,R1
  ASL R1
  MOV #6,TASK(R1) ;ORDER STOP TASK
  MOV #2*SECONDS,HUNGTIME
  CALL QUE
  ON.ERROR
  MOV RO,BAD
  ERROR +60 ;SLAVE HUNG SYSTEM!
  $RETURN ERROR
END ;OF ON.ERROR
TST TASK(R1)
BPL 1$
END ;OF FOR RO
$RETURN NOERROR
.DSABL LSB
```

```

1739          .SBTTL  ERROR DATA SETUP
1740
1741          USE THIS   IF THIS CONDITION DISCRIBES THE ERROR
1742
1743          PERR01     TRAP
1744                   BAD DATA IN R0 UNLESS ABORTED
1745                   THEN BAD DATA IS POINTED TO BY -(R4)
1746                   GOOD DATA IN R5
1747
1748          PERR02     TRAP
1749                   BAD DATA IN R1 UNLESS ABORTED
1750                   THEN BAD DATA IS POINTED TO BY -(R4)
1751                   GOOD DATA IN R2
1752
1753          PERR03     TRAP
1754                   BAD DATA IS POINTED TO BY -(R1)
1755                   GOOD DATA IN R4
1756
1757          PERR04     TRAP
1758                   BAD DATA IN R4 UNLESS ABORTED
1759                   THEN BAD DATA IS POINTED TO BY -2(R0)
1760                   GOOD DATA IN R2
1761
1762          PERR05     JSR    PC
1763                   BAD DATA IS POINTED TO BY -(R0)
1764                   GOOD DATA IN R2
1765                   RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
1766
1767          PERR06     JSR    PC
1768                   BAD DATA IS POINTED TO BY -(R0)
1769                   GOOD DATA IS ZERO
1770                   RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
1771
1772          PERR07     TRAP
1773                   BAD DATA IN R2 UNLESS ABORTED
1774                   THEN BAD DATA IS POINTED TO BY (R1)
1775                   GOOD DATA IN DATBUF
1776
1777          PERR10     TRAP
1778                   BAD DATA IN R2 UNLESS ABORTED
1779                   THEN BAD DATA IS POINTED TO BY 2(R1)
1780                   GOOD DATA IN DATBUF+2
1781
1782          PERR11     TRAP
1783                   BYTE TEST
1784                   BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
1785                   THEN BAD DATA IS POINTED TO BY (R1)
1786                   GOOD DATA IS A ZERO BYTE
1787
1788          PERR12     TRAP
1789                   BYTE TEST
1790                   BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
1791                   THEN BAD DATA IS POINTED TO BY (R1)
1792                   GOOD DATA IS A BYTE OF ONES
1793
1794          PERR13     TRAP
1795                   BAD DATA IN R0 UNLESS ABORTED
    
```

1796	:		THEN BAD DATA IS POINTED TO BY (R1)
1797	:		GOOD DATA IS ZERO
1798	:		
1799	:	PERR14	TRAP
1800	:		BAD DATA IN R0 UNLESS ABORTED
1801	:		THEN BAD DATA IS POINTED TO BY (R1)
1802	:		GOOD DATA IS ONES
1803	:		
1804	:	PERR15	TRAP
1805	:		BAD DATA IN R0 UNLESS ABORTED
1806	:		THEN BAD DATA IS POINTED TO BY (R1)
1807	:		GOOD DATA IN TSTDAT
1808	:		
1809	:	PERR16	TRAP
1810	:		BAD DATA IN R0 UNLESS ABORTED
1811	:		THEN BAD DATA IS POINTED TO BY (R1)
1812	:		GOOD DATA IN TSTDAT+2
1813	:		
1814	:	PERR17	TRAP
1815	:		BAD DATA IN R0 UNLESS ABORTED
1816	:		THEN BAD DATA IS POINTED TO BY (R1)
1817	:		GOOD DATA IN R2
1818	:		
1819	:	PERR20	TRAP
1820	:		BAD DATA IN R0 UNLESS ABORTED
1821	:		THEN BAD DATA IS POINTED TO BY (R1)
1822	:		GOOD DATA IN R3
1823	:		
1824	:	PERR21	TRAP
1825	:		7 BIT BYTE TEST
1826	:		BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
1827	:		THEN BAD DATA IS POINTED TO BY (R1)
1828	:		GOOD DATA IS A 7 BIT BYTE ON ONES
1829	:		
1830	:	PERR22	TRAP
1831	:		BAD DATA IN R2 UNLESS ABORTED
1832	:		THEN BAD DATA IS POINTED TO BY (R1)
1833	:		GOOD DATA IN R0
1834	:		
1835	:	PERR23	TRAP
1836	:		BAD DATA IN R0 UNLESS ABORTED
1837	:		THEN BAD DATA IS POINTED TO BY (R1)
1838	:		GOOD DATA IN R4
1839	:		
1840	:	PERR24	TRAP
1841	:		BAD DATA IN R0 UNLESS ABORTED
1842	:		THEN BAD DATA IS POINTED TO BY (R2)
1843	:		GOOD DATA IN R3
1844	:		
1845	:	PERR25	TRAP
1846	:		BAD DATA POINTED TO BY -(R1)
1847	:		GOOD DATA IN R2 UNLESS LOC V177654 IS SET
1848	:		THEN GOOD DATA IS IN R3
1849	:		
1850	:	PERR26	TRAP
1851	:		BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
1852	:		GOOD DATA IS 000000,,100000,,100

```
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880
```

 : 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

1883	064574	010437	002040		\$PER01: MOV R4,ADDRESS	
1884	064600	162737	000002	002040	SUB #2,ADDRESS	
1885	064606	010037	002056		MOV R0,BAD	
1886	064612	010537	002050		MOV R5,GOOD	
1887	064616	000137	065330		JMP PERRAW	
1888						
1889	064622	010437	002040		\$PER02: MOV R4,ADDRESS	
1890	064626	162737	000002	002040	SUB #2,ADDRESS	
1891	064634	010137	002056		MOV R1,BAD	
1892	064640	010237	002050		MOV R2,GOOD	
1893	064644	000137	065330		JMP PERRAW	
1894						
1895	064650	010137	002040		\$PER03: MOV R1,ADDRESS	
1896	064654	162737	000002	002040	SUB #2,ADDRESS	
1897	064662	010437	002050		MOV R4,GOOD	
1898	064666	016137	177776	002056	MOV -2(R1),BAD	
1899	064674	000137	065330		JMP PERRAW	
1900						
1901	064700	010037	002040		\$PER04: MOV R0,ADDRESS	
1902	064704	162737	000002	002040	SUB #2,ADDRESS	
1903	064712	010437	002056		MOV R4,BAD	
1904	064716	010237	002050		MOV R2,GOOD	
1905	064722	000137	065330		JMP PERRAW	
1906						
1907	064726	010237	002050		PERR05: MOV R2,GOOD	
1908	064732	014037	002056		PERA05: MOV -(R0),BAD	
1909	064736	010037	002040		MOV R0,ADDRESS	
1910	064742	062700	000002		ADD #2,R0	;RESTORE R0
1911	064746	004737	036770		CALL BADSTACK	
1912	064752	000207			RETURN	
1913						
1914	064754	005037	002050		PERR06: CLR GOOD	
1915	064760	000764			BR PERA05	
1916						
1917	064762	010137	002040		\$PER07: MOV R1,ADDRESS	
1918	064766	010237	002056		MOV R2,BAD	
1919	064772	013737	002340	002050	MOV DATBUF,GOOD	
1920	065000	000137	065330		JMP PERRAW	
1921						
1922	065004				\$PER10: LET ADDRESS := R1 + #2	
1923	065016				LET BAD := R2	
1924	065022				LET GOOD := DATBUF+2	
1925	065030	000137	065330		JMP PERRAW	
1926						
1927	065034				\$PER11: LET ADDRESS := R1	
1928	065040				LET BAD := R0	
1929	065044				LET GOOD := #0	
1930	065050	000137	065402		JMP PERRAB	
1931						
1932	065054				\$PER12: LET ADDRESS := R1	
1933	065060				LET BAD := R0	
1934	065064				LET GOOD := #377	
1935	065072	000137	065402		JMP PERRAB	

1938	065076		\$PER13: LET ADDRESS := R1
1939	065102		LET BAD := R0
1940	065106		LET GOOD := #0
1941	065112	000137 065330	JMP PERRAW
1942			
1943	065116		\$PER14: LET ADDRESS := R1
1944	065122		LET BAD := R0
1945	065126		LET GOOD := ONES
1946	065134	000137 065330	JMP PERRAW
1947			
1948	065140		\$PER15: LET ADDRESS := R1
1949	065144		LET BAD := R0
1950	065150		LET GOOD := TSTDAT
1951	065156	000137 065330	JMP PERRAW
1952			
1953	065162		\$PER16: LET ADDRESS := R1
1954	065166		LET BAD := R0
1955	065172		LET GOOD := TSTDAT+2
1956	065200	000453	BR PERRAW
1957			
1958	065202		\$PER17: LET ADDRESS := R1
1959	065206		LET BAD := R0
1960	065212		LET GOOD := R2
1961	065216	000444	BR PERRAW
1962			
1963	065220		\$PER20: LET ADDRESS := R1
1964	065224		LET BAD := R0
1965	065230		LET GOOD := R3
1966	065234	000435	BR PERRAW
1967			
1968	065236		\$PER21: LET ADDRESS := R1
1969	065242		LET BAD := R0
1970	065246		LET GOOD := #177
1971	065254	000477	BR PERRAW
1972			
1973	065256		\$PER22: LET ADDRESS := R1
1974	065262		LET BAD := R2
1975	065266		LET GOOD := R0
1976	065272	000416	BR PERRAW
1977			
1978	065274		\$PER23: LET ADDRESS := R1
1979	065300		LET BAD := R0
1980	065304		LET GOOD := R4
1981	065310	000407	BR PERRAW
1982			
1983	065312		\$PER24: LET ADDRESS := R2
1984	065316		LET BAD := R0
1985	065322		LET GOOD := R3
1986	065326	000400	BR PERRAW

1988 065330

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

1989 065330 004737 065564
1990 065334
1991 065346
1992 065360 004737 065540
1993 065364
1994 065372 104011
1995 065374
1996 065376 104012
1997 065400
1998 065400 000002
1999
2000 065402

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
:*****

2001 065402 004737 065564
2002 065406
2003 065420
2004 065432 004737 065540
2005 065436
2006 065444 104014
2007 065446
2008 065450 104015
2009 065452
2010 065452 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

2013 065454

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

2014 065454
2015 065466 004737 065540
2016 065472 004737 065564
2017 065476 104022
2018 065500 000002
2019

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

2020 065502
2021 065510
2022 065516 000137 054652
2023
2024 065522 005037 002052
2025 065526
2026 065534 000137 054652
2027

\$PER26: LET GOOD2 := #100000
LET GOOD3 := #100
JMP PERRA3

\$PER27: CLR GOOD2
LET GOOD3 := #077
JMP PERRA3

2028 065540

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

2029 065540
2030 065542 013700 002050
2031 065546 013737 002056 002064
2032 065554 074037 002064
2033 065560
2034 065562 000207

PUSH R0
MOV GOOD,R0
MOV BAD,BAD XOR
XOR R0,BAD XOR
POP R0
RETURN

2037 065564

PERBNK: SUBST<<LOG ERROR ON BAD BANK>>
 :*****
 :*SUBTEST LOG ERROR ON BAD BANK
 :*****

2038
 2039 065564
 2040 065570 013701 002106
 2041 065574 070127 000006
 2042 065600 052761 004000 003016
 2043 065606 062701 000005
 2044 065612 105261 003016
 2045 065616 001002
 2046 065620 105361 003016
 2047 065624 126137 003016 002702 12\$:
 2048 065632 101403
 2049 065634
 2050 065642 11\$:
 2051 065646 000207
 2052
 2053 065650 010137 002040
 2054 065654 010037 002056
 2055 065660
 2056 065666 013737 002344 002050
 2057 065674
 2058 065676 013737 002346 002050
 2059 065704
 2060 065704 004737 065540
 2061 065710
 2062 065716 000207
 2063
 2064 065720
 2065 065732 004737 065650
 2066 065736
 2067 065746 104042
 2068 065750
 2069 065750
 2070 065760 104045
 2071 065762
 2072 065762
 2073 065772 104046
 2074 065774
 2075 065774
 2076 066004 104047
 2077 066006
 2078 066006
 2079 066014 000002

```

:WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE
PUSH R0,R1
MOV BANK,R1
MUL #6,R1
BIS #BIT11,CONFIG(R1)
ADD #5,R1 ;POINT TO MAX ERROR BYTE
INCB CONFIG(R1) ;BUMP BANK COUNTER
BNE 12$ ;NO OVERFLOW - SKIP
DECB CONFIG(R1) ;SET BACK TO 255.
CMPB CONFIG(R1),ERRMAX ;IS IT PAST MAX?
BLOS 11$ ;NO - SKIP
SET TOOMANY ;YES
POP R1,R0
RETURN

PERECC: MOV R1,ADDRESS
MOV R0,BAD
IF R1 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 +42
END ;OF IF REALPAT
IF REALPAT EQ #14
ERROR +45
END ;OF IF REALPAT
IF REALPAT EQ #15
ERROR +46
END ;OF IF REALPAT
IF REALPAT EQ #16
ERROR +47
END ;OF IF REALPAT
SET HEADER
RTI
    
```

```

2082 066016
2083 066030 010137 002040
2084 066034 010037 002056
2085 066040 010237 00205C
2086 066044
2087 066052 104043
2088 066054
2089 066062 000002
2090
2091 066064
2092 066076 010137 002040
2093 066102 010037 002056
2094 066106 105037 002057
2095 066112 012737 000377 002050
2096 066120 004737 065540
2097 066124
2098 066132 104044
2099 066134
2100 066142 000002
2101
2102 066144
2103 066156
2104 066166 104016
2105 066170
2106 066172 104001
2107 066174
2108 066174 000002
2109
2110
2111 066176 004737 065564
2112 066202 004737 036770
2113 066206 013737 002036 002056
2114 066214 012737 000012 002050
2115 066222 104054
2116 066224 062706 000004
2117 066230 000207

$PER32: IF BADPC EQ #0 THEN $CALL BADSTACK
        MOV R1,ADDRESS
        MOV R0,BAD
        MOV R2,GOOD
        SET HEADER
        ERROR +43
        SET HEADER
        RTI

$PER33: IF BADPC EQ #0 THEN $CALL BADSTACK
        MOV R1,ADDRESS
        MOV R0,BAD
        CLRB BAD+1
        MOV #377,GOOD
        CALL PERXOR
        SET HEADER
        ERROR +44
        SET HEADER
        RTI

$PER34: IF BADPC EQ #0 THEN $CALL BADSTACK
        IF #BIT15!BIT4 OFF.IN CSR
            ERROR +16 ;NO SBE OR DBE
        ELSE
            ERROR +1 ;EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
        END ;OF IF #BIT15!BIT4
        RTI

$PER35: ;DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG
        CALL PERBNK
        CALL BADSTACK
        MOV BADPSW,BAD
        MOV #12,GOOD
        ERROR +54
        ADD #4,SP ;FIX STACK FROM TRAP
        RETURN ;ABORTING TEST
    
```

```

2120
2121
2122
2123
2124
2125
2126
2127
2128
2129 066232 005237 075144
2130 066236
2131 066240 005037 075144
2132 066244 105237 075146
2133 066250
2134 066250 104410
2135 066252
2136 066262
2137 066312 004737 072154
2138 066316
2139 066316
2140 066316
2141 066334 005037 002520
2142 066340
2143 066344 000137 043522
2144 066350
2145 066350
2146 066356 000002
2147 066360
2148 066360
2149
2150 066370 000421
2151
2152 066372 013746 000004
2153 066376 012737 066416 000004
2154 066404 005737 177060
2155 066410 012637 000004
2156 066414 000424
2157 066416 062706 000004
2158 066422 005037 177766
2159 066426 012637 000004
2160 066432 000407
2161 066434
2162 066434 105737 002016
2163 066440 001412
2164 066442 032777 001000 114316
2165 066450 001404
2166 066452 013737 002750 002746
2167 066460 000410
2168 066462 105037 002016
2169 066466 011637 002746
2170 066472 011637 002750
2171 066476 005037 002454
2172 066502 004737 066514
2173 066506 013716 002746
2174 066512 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=IOT
$SCOPE: INC $DEVCT ;TELL APT WE ARE ALIVE
IF RESULT IS LT
CLR $DEVCT
INCB $UNIT
END ;OF IF RESULT
CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
IF CPUBIT EQ #MASTER ;:IF I'M THE MASTER
IF PTYFLAG IS TRUE OR PTYFLAG+2 IS TRUE OR PTYFLAG+4 IS TRUE OR PTYFLAG+6 IS TRUE
CALL QUE
END ;OF IF PTYFLAG
END ;OF IF CPUBIT
IF STOPOK IS TRUE AND #SW8 SET.IN @SWR
CLR STOPOK
TYPE MSG112 ;PROGRAM HALTING BECAUSE OF SWITCH #8
JMP EXIT
END ;OF IF STOPOK
IF NOSCOPE IS TRUE
RTI
END ;OF IF NOSCOPE
1$: 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
MOV ERRVEC,-(SP) ;:SET FOR TIMEOUT
MOV #1$,ERRVEC ;:RESTORE THE ERROR VECTOR
TST 177060 ;:TIME OUT ON XOR?
MOV (SP)+,ERRVEC ;:GO TO THE NEXT TEST
BR $SVLAD ;:FIX STACK FROM TRAP
1$: ADD #4,SP ;:RESET CPU ERROR REGISTER
CLR CPUERR ;:RESTORE THE ERROR VECTOR
MOV (SP)+,ERRVEC ;:LOOP ON THE PRESENT TEST
BR 4$
2$:;#####END OF CODE FOR THE XOR TESTER#####
3$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
BEQ $SVLAD ;:BR IF NO
BIT #SW9,@SWR ;:LOOP ON ERROR?
BEQ 5$ ;:BR IF NO
4$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
5$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
$SVLAD: MOV (SP),$LPADR ;:SAVE SCOPE LOOP ADDRESS
MOV (SP),$LPERR ;:SAVE ERROR LOOP ADDRESS
CLR $ESCAPE ;:CLEAR THE ESCAPE FROM ERROR ADDRESS
$OVER: CALL GETDIS ;:FUDGE RETURN ADDRESS
MOV $LPADR,(SP) ;:FIXES PS
RTI
  
```

```

2176 066514          GETDIS: SUBTST <<SUBR DISPLAY>>
:*****
:*SUBTEST          SUBR    DISPLAY
:*****
2177 066514  113737  002106  002015          MOVB    BANK,$BANK
2178 066522  113737  002366  002014          MOVB    REALPAT,$PATMAR
2179 066530          PUSH    R0,4
2180 066536          SET4    #2$           ;TRAPS TO 4 GOTO 2$
2181 066544  005000          CLR     R0
2182 066546  013700  177750          MOV     MAINT,R0
2183 066552  072027  000004          ASH    #4,R0
2184 066556  000402          BR     3$
2185 066560  062706  000004          2$:    ADD    #4,SP           ;FIX STACK FROM TRAP
2186 066564  042700  177437          3$:    BIC    #^C340,R0
2187 066570  150037  002014          BISB   R0,$PATMAR
2188 066574  005737  002134          TST    RLFLAG           ;ARE WE RELOCATED?
2189 066600  001403          BEQ    1$               ;NO - SKIP
2190 066602  052737  100000  002014          BIS    #BIT15,$PATMAR   ;YES - SET MSB
2191 066610          1$:    IF MDISPLAY NE #0 THEN LET $PATMAR := MDISPLAY
2192 066624  013777  002014  114136          MOV    $PATMAR,@DISPLAY
2193 066632  013737  002014  000174          MOV    $PATMAR,DISPREG  ;SOFTWARE DISPLAY REGISTER
2194 066640          POP    4,R0
2195 066646          RES4
2196 066660  000207          RETURN           ;RESTORE TRAPS TO 4 TO DEFAULT
    
```

```

2199          .SBTTL  ROUTINE ERROR HANDLER
2200          ;*****
2201          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2202          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2203          ;*AND GO TO $ERRTYP ON ERROR
2204          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2205          ;*SW15=1      HALT ON ERROR
2206          ;*SW13=1      INHIBIT ERROR TYPEOUTS
2207          ;*SW10=1      BELL ON ERROR
2208          ;*SW9=1 LOOP ON ERROR
2209          ;*CALL
2210          ;*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2211
2212          .ENABL  LSB
2213 066662 $ERROR: IF NOERROR IS FALSE
2214 066670      104410      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
2215 066672      105237      002016      1$:      INCB  $ERFLG      ;;SET THE ERROR FLAG
2216 066676      001775      BEQ  1$      ;;DON'T LET THE FLAG GO TO ZERO
2217 066700      004737      066514      CALL  GETDIS      ;;SETUP DISPLAY STUFF
2218 066704      013737      002014      075140      MOV  $PATMAR,$TESTN ;FOR APT
2219 066712      032777      002000      114046      BIT  #SW10,@SWR      ;;BELL ON ERROR?
2220 066720      001404      BEQ  2$      ;;NO - SKIP
2221 066722      TYPE  $BELL      ;;RING BELL
2222 066726      TYPE  MSG014      ;;CONTROL Z
2223 066732      005237      002754      2$:      INC  $ERTTL      ;;COUNT THE NUMBER OF ERRORS
2224 066736
2225 066740      012737      077777      002754      IF RESULT IS MI
2226 066746      MOV  #77777,$ERTTL
2227 066746      END ;OF IF RESULT
2228 066746      011637      002024      END ;OF IF NOERROR
2229 066752      162737      000002      002024      MOV  (SP),ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
2230 066760      010637      002030      SUB  #2,ERRPC
2231 066764      016637      000002      002034      MOV  SP,ERRSP
2232 066772      117737      113026      002017      MOV  2(SP),ERRPSW
2233 067000      MOVB  @ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
2234 067006      IF NOERROR IS FALSE
2235 067014      013737      002026      002024      IF BADPC NE #0
2236 067022      162737      000002      002024      MOV  BADPC,ERRPC
2237 067030      013737      002032      002030      SUB  #2,ERRPC
2238 067036      013737      002036      002034      MOV  BADSP,ERRSP
2239 067044      005037      002026      MOV  BADPSW,ERRPSW
2240 067050      CLR  BADPC
2241 067050      013737      002024      075136      END ;IF
2242 067056      MOV  ERRPC,$FATAL      ;FOR APT
2243 067074      000412      IF  #SW13 SET.IN @SWR OR SLAERROR IS TRUE
2244 067076      BR  3$
2245 067076      END ;OF IF #SW13
2246 067114      IF  #SW5 SET.IN @SWR AND TOOMANY IS TRUE
2247 067116      GOTO 3$
2248 067116      END ;OF IF #SW5
2249 067116      004737      067324      END ;OF IF NOERROR
          CALL  $ERRTYP      ;;GO TO USER ERROR ROUTINE
  
```

```

2251 067122
2252 067130 005777 113632
2253 067134 100002
2254 067136 000000
2255 067140 104410
2256 067142
2257 067160 013716 002750
2258 067164
2259 067164 005737 002454
2260 067170 001402
2261 067172 013716 002454
2262 067176
2263 067204
2264 067226 012737 000001 075134
2265 067234 000137 043522
2266 067240
2267 067240
2268 067256
2269 067262 013700 000042
2270 067266 005037 000042
2271 067272 000137 014760
2272 067276
2273 067276
2274 067276
2275 067300
2276 067306
2277 067306
2278 067322 000002
2279

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
    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,CPUERR,NOERROR
      RTI          ;;RETURN
      .DSABL LSB
  
```

.SBTTL ROUTINE ERROR MESSAGE TYPEOUT

```

2282
2283
2284
2285
2286
2287
2288
2289 067324 104415
2290 067326
2291 067332 005000
2292 067334 153700 002017
2293 067340 001004
2294
2295 067342
2296 067350 000512
2297 067352 005300
2298 067354 006300
2299 067356 006300
2300 067360 006300
2301 067362 062700 101372
2302 067366 012037 067424
2303 067372 001417
2304 067374 005737 002530
2305 067400 001003
2306 067402 005737 002724
2307 067406 100011
2308 067410 005737 002070
2309 067414 001402
2310 067416
2311 067422
2312 067424 000000
2313 067426
2314 067432 012037 067456
2315 067436 001412
2316 067440 005737 002530
2317 067444 001003
2318 067446 005737 002724
2319 067452 100004
2320 067454
2321 067456 000000
2322 067460
2323 067464 012001
2324 067466 001436
2325 067470 012002

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

```

2328 067472 112203          7$:  MOVB  (R2)+,R3
2329 067474 006303          ASL   R3           ;MAKE IT A WORD ADDRESS
2330 067476 004773 067504      CALL  @8$(R3)
2331 067502 000421          BR    9$
2332 067504 067636          8$:  TAG70$
2333 067506 067646          TAG71$
2334 067510 067656          TAG72$
2335 067512 067722          TAG73$
2336 067514 067730          TAG74$
2337 067516 067742          TAG75$
2338 067520 067754          TAG76$
2339 067522 067776          TAG77$
2340 067524 070000          TAG78$
2341 067526 070006          TAG79$
2342 067530 070066          TAG80$
2343 067532 070100          TAG81$
2344 067534 070122          TAG82$
2345 067536 070324          TAG83$
2346 067540 070410          TAG84$
2347 067542 070502          TAG85$
2348 067544 070612          TAG86$
2349 067546 062701 000002      9$:  ADD   #2,R1       ;UPDATE DATA TABLE POINTER
2350 067552 005711          TST  (R1)         ;:IS THERE ANOTHER NUMBER?
2351 067554 001403          BEQ  10$         ;:BR IF NO
2352 067556          TYPE  MSG018    ;:TYPE 2 SPACES
2353 067562 000743          BR   7$         ;:LOOP
2354
2355 067564 005737 002116      10$: TST  MUT        ;:IS THERE A MEMORY UNDER TEST
2356 067570 001402          BEQ  11$         ;:NO - SKIP
2357 067572 005237 002724      INC  HEADER      ;:YES - BUMP HEADER FLAG
2358 067576 104416          11$: RESREG
2359 067600          IF #SW7 SET.IN @SWR AND DETFLAG IS FALSE AND NOERROR IS FALSE
2360 067624 004737 071364      CALL  DETAIL
2361 067630          END ;OF IF #SW7
2362 067630          TYPE  MSG104   ;:CONTROL Z
2363 067634 000207          RETURN
  
```

2366
 2367
 2368
 2369 067636
 2370 067644 000207
 2371
 2372
 2373
 2374
 2375 067646
 2376 067654 000207
 2377
 2378
 2379
 2380
 2381 067656
 2382 067662 013701 002106
 2383 067666 070127 000006
 2384 067672
 2385 067700 004737 036012
 2386 067704 005037 002464
 2387 067710
 2388 067714
 2389 067720 000207
 2390
 2391
 2392
 2393
 2394 067722 004737 070672
 2395 067726 000207
 2396
 2397
 2398
 2399
 2400 067730
 2401 067740 000207
 2402
 2403
 2404
 2405
 2406 067742
 2407 067752 000207
 2408
 2409
 2410
 2411
 2412 067754 005737 002126
 2413 067760 001003
 2414 067762
 2415 067766 000207
 2416 067770
 2417 067774 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 #6,R1
      SET NOTAB ;:INDICATE NO TABLE TO BE PRINTED - NOW
      CALL TCFIG1
      CLR NOTAB
      POP R5,R1
      TYPE MSG014 ;:1 SPACE
      RETURN

:*****
:*** BOX ***
:*****
TAG73$: CALL WHOBOX
      RETURN

:*****
:*** PATTERN ***
:*****
TAG74$: TYPOCS REALPAT,<TYPE (0-77)>,.2,Z
      RETURN

:*****
:*** BANK ***
:*****
TAG75$: TYPOCS BANK,<TYPE (0-167)>,.3
      RETURN

:*****
:*** MTYPE ***
:*****
TAG76$: TST MKFLAG
      BNE 1$
      TYPE MSG068 ;:MJ11
      RETURN
1$: TYPE MSG069 ;:MK11
      RETURN

```

2420
2421
2422
2423 067776 000207
2424
2425
2426
2427
2428 070000
2429 070004 000207
2430
2431
2432
2433
2434 070006 013737 002040 002044
2435 070014 162737 060000 002044
2436 070022 013737 002106 002046
2437 070030 006237 002046
2438 070034 103003
2439 070036 052737 100000 002044
2440 070044 012746 002044
2441 070050 004737 075014
2442 070054 062706 000002
2443 070060
2444 070064 000207
2445
2446
2447
2448 070066
2449 070076 000207
2450
2451
2452
2453
2454 070100
2455 070104
2456 070114
2457 070120 000207

```
*****  
*** SPARE FUNCTION ***  
*****  
TAG77$: RETURN  
  
*****  
*** UNKNOWN DATA ***  
*****  
TAG78$: TYPE MSG061  
RETURN  
  
*****  
*** PHYSICAL ADDRESS ***  
*****  
TAG79$: MOV ADDRESS,PHYADD  
SUB #FIRST,PHYADD  
MOV BANK,PHYADD+2  
ASR PHYADD+2  
BCC 1$  
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  
  
*****  
*** MARGIN ***  
*****  
TAG80$: TYPOCS MARGIN,<TYPE (0,2,3,4, OR 5)>,1  
RETURN  
  
*****  
*** OCTAL BYTE ***  
*****  
TAG81$: TYPE MSG018 ;2 SPACES  
TYPOCS @(R1),<TYPE BYTE>,3,2  
TYPE MSG014 ;SPACE  
RETURN
```

2460
 2461
 2462
 2463
 2464
 2465
 2466
 2467
 2468
 2469
 2470
 2471
 2472 070122 012737 037477 115336 TAG82\$:
 2473 070130
 2474 070136
 2475 070160 013737 002000 002256
 2476 070166
 2477 070172 104426
 2478 070174
 2479 070204 013702 002144
 2480 070210 072227 177774
 2481 070214 042702 177761
 2482 070220
 2483 070234
 2484 070234
 2485 070240
 2486 070246 162702 000012
 2487 070252 000302
 2488 070254 062702 030061
 2489 070260
 2490 070262 000302
 2491 070264 062702 030040
 2492 070270
 2493 070270 010237 115336
 2494 070274
 2495 070274
 2496 070316
 2497 070316
 2498 070322 000207

```

:*****
:*** ARRAY ***
:*****
: SOMETIMES THE ARRAY INFORMATION MAY BE MISLEADING
: IF YOU ARE LOOKING FOR SBE'S (SWITCH 0 IS UP)
: IF AN ERROR IS BEING REPORTED AND THE CODE THAT TYPES THE
: ERROR HAS A SINGLE BIT ERROR
: THE OLD SBE INFORMATION WILL BE WRITTEN OVER AND THE ARRAY
: TYPED MAY BE THE ARRAY THAT HAS THE ERROR REPORT CODE IN IT
: AND NOT THE ARRAY THAT WAS UNDER TEST AND THAT THE REST
: OF THE ERROR REPORT IS TALKING ABOUT.
: HOWEVER THE ARRAY IS STILL BAD (IF YOU'RE LOOKING FOR SBE'S).
:
TAG82$: MOV #'??,MSG074
IF MKFLAG IS TRUE
  PUSH CSR,CSR+2,R0,R1,R2,CSRNO
  MOV BADCSR,CSRNO
  SET R2
  READCSR
  IF #BIT15!BIT4 SET.IN CSR
    MOV CSR,R2
    ASH #-4,R2
    BIC #^C16,R2
    IF #BIT9 SET.IN CSR+2 THEN LET R2 := R2 SET.BY #BIT0
  END :OF IF #BIT15!BIT4
  IF R2 GE #0
    IF R2 GT #9.
      SUB #10.,R2
      SWAB R2
      ADD #'10,R2
    ELSE
      SWAB R2
      ADD #'0,R2
    END :OF IF #BIT3
  MOV R2,MSG074
  END :OF IF R2
  POP CSRNO,R2,R1,R0,CSR+2,CSR
  END :OF IF MKFLAG
  TYPE MSG074
  RETURN

```

2501
2502
2503
2504 070324
2505 070330
2506 070334 017100 000000
2507 070340
2508 070344 106100
2509 070346
2510 070350 112737 000061 112134
2511 070356
2512 070360 112737 000060 112134
2513 070366
2514 070366
2515 070372
2516 070402
2517 070406 000207
2518
2519
2520
2521
2522 070410
2523 070414
2524 070420 017100 000000
2525 070424 012702 000004
2526 070430 072002
2527 070432
2528 070436 106100
2529 070440
2530 070442 112737 000061 112134
2531 070450
2532 070452 112737 000060 112134
2533 070460
2534 070460
2535 070464
2536 070474
2537 070500 000207

```
*****  
*** BINARY BYTE ***  
*****  
TAG83$: PUSH R0,R2  
TYPE MSG018 ;2 SPACES  
MOV @(R1),R0  
FOR R2 := #1 TO #8.  
ROLB R0  
ON.ERROR  
MOVB #'1,MSG015  
ELSE  
MOVB #'0,MSG015  
END ;OF ON.ERROR  
TYPE MSG015  
END ;OF FOR R2  
POP R2,R0  
RETURN  
  
*****  
*** 4 BINARY BITS ***  
*****  
TAG84$: PUSH R0,R2  
TYPE MSG018 ;2 SPACES  
MOV @(R1),R0  
MOV #4,R2  
ASH R2,R0 ;SHIFT 4 PLACES LEFT  
FOR R2 := #1 TO #4  
ROLB R0  
ON.ERROR  
MOVB #'1,MSG015  
ELSE  
MOVB #'0,MSG015  
END ;OF ON.ERROR  
TYPE MSG015  
END ;OF FOR R2  
POP R2,R0  
RETURN
```

2540
2541
2542
2543 070502
2544 070504 012737 037477 115336
2545 070512 013702 002150
2546 070516 072227 177774
2547 070522 042702 177761
2548 070526
2549 070542
2550 070546
2551 070554 162702 000012
2552 070560 000302
2553 070562 062702 030061
2554 070566
2555 070570 000302
2556 070572 062702 030040
2557 070576
2558 070576 010237 115336
2559 070602
2560 070602
2561 070606
2562 070610 000207
2563
2564
2565
2566
2567 070612
2568 070620
2569 070624
2570 070624
2571 070634
2572 070640
2573 070640
2574 070650
2575 070654
2576 070654
2577 070664
2578 070670
2579 070670 000207

```
*****  
*** ARRAY FROM SBE FROM OLD CSR ***  
*****  
TAG85$: PUSH R2  
MOV #'??,MSG074  
MOV SBECR,R2  
ASH #-4,R2  
BIC #^C16,R2  
IF #BIT9 SET.IN SBECR+2 THEN LET R2 := R2 SET.BY #BIT0  
IF R2 GE #0  
IF R2 GT #9.  
SUB #10.,R2  
SWAB R2  
ADD #'10,R2  
ELSE  
SWAB R2  
ADD #'0,R2  
END ;OF IF #BIT3  
MOV R2,MSG074  
END ;OF IF R2  
TYPE MSG074  
POP R2  
RETURN
```

```
*****  
*** MASTER OR SLAVE1 OR SLAVE2 OR SLAVE3 FROM BAD  
*****  
TAG86$: IF BAD EQ #0  
TYPE MSG113 ;MASTER  
END IF  
IF BAD EQ #1  
TYPE MSG114 ;SLAVE1  
END IF  
IF BAD EQ #2  
TYPE MSG115 ;SLAVE2  
END IF  
IF BAD EQ #3  
TYPE MSG116 ;SLAVE3  
END IF  
RETURN
```

2581 070672

```

WHOBOX: SUBST <<SUBR WHO'S BOX IS THIS?>>
:*****
:*SUBTEST      SUBR      WHO'S BOX IS THIS?
:*****
    
```

2582 070672 005737 002126

2583 070676 001003

2584

2585 070700

2586 070704 000207

2587

2588 070706

2589 070722 005037 002424

2590 070726 012703 000200

2591 070732 012701 000060

2592 070736

2593 070736

2594 070740

2595 070746 016437 002214 002260

2596 070754 016437 002234 002262

2597 070762 010437 002000

2598 070766 006337 002000

2599 070772

2600 071012 016402 002174

2601 071016 072227 177764

2602 071022 042702 177770

2603 071026

2604 071036 071152

2605 071040 071162

2606 071042 071202

2607 071044 071232

2608 071046 071262

2609 071050 071264

2610 071052 071304

2611 071054 071334

2612 071056

2613 071064

2614 071072

2615 071072

2616 071072 006003

2617 071074 005201

2618 071076

2619 071110

2620 071110

2621 071116 110137 114717

2622 071122

2623 071122

2624 071126 112737 000077 114717

2625 071134

2626 071150 000207

```

TST      MKFLAG          ;MK11?
BNE      1$              ;YES - SKIP
;NO - MJ11 - DON'T KNOW ABOUT BOX NUMBERS
TYPE     MSG057          ; ?
RETURN

1$:      PUSH      CSRNO,R1,R2,R3,R4
        CLR      SUCCESS
        MOV      #200,R3
        MOV      #'0,R1
        BEGIN WHODUNIT
        FOR R4 := #0 TO #16 BY #2
        IFB R3 SET.IN MKCSRS
        MOV CSRFB(R4),FIRSTB
        MOV CSRLB(R4),LASTB
        MOV R4,BADCSR
        ASL BADCSR
        IF FIRSTB LE BANK AND LASTB GE BANK
        MOV      CSR2(R4),R2
        ASH      #-12.,R2
        BIC      #^C7,R2
        CASE R2
        ADD..0
        ADD..1
        ADD..2
        ADD..3
        ADD..4
        ADD..5
        ADD..6
        ADD..7
        END ;OF CASE R2
        IF SUCCESS IS TRUE THEN LEAVE WHODUNIT
        END ;OF IF FIRSTB
        END ;OF IFB R3
        ROR      R3
        INC      R1
        END ;OF FOR CSRNO
        END WHODUNIT
        IF SUCCESS IS TRUE
        MOVB R1,MSG057+1
        END ;OF IF SUCCESS
        TYPE     MSG057
        MOVB     #'?,MSG057+1
        POP      R4,R3,R2,R1,CSRNO
        RETURN
    
```

2629 071152
 2630 071160 000207
 2631 071162
 2632 071200 000207
 2633 071202
 2634 071230 000207
 2635 071232
 2636 071260 000207
 2637 071262 000207
 2638 071264
 2639 071302 000207
 2640 071304
 2641 071332 000207
 2642 071334
 2643 071362 000207
 2644
 2645 071364

ADD..0: SET SUCCESS
 RETURN
 ADD..1: IF #BIT2 OFF.IN ADDRESS THEN LET SUCCESS := ONES
 RETURN
 ADD..2: IF #BIT3 OFF.IN ADDRESS AND #BIT2 OFF.IN ADDRESS THEN LET SUCCESS := ONES
 RETURN
 ADD..3: IF #BIT3 SET.IN ADDRESS AND #BIT2 OFF.IN ADDRESS THEN LET SUCCESS := ONES
 RETURN
 ADD..4: RETURN
 ADD..5: IF #BIT2 SET.IN ADDRESS THEN LET SUCCESS := ONES
 RETURN
 ADD..6: IF #BIT3 OFF.IN ADDRESS AND #BIT2 SET.IN ADDRESS THEN LET SUCCESS := ONES
 RETURN
 ADD..7: IF #BIT3 SET.IN ADDRESS AND #BIT2 SET.IN ADDRESS THEN LET SUCCESS := ONES
 RETURN

DETAIL: SUBTST <<SUBR DETAILED ERROR REPORT>>
 :*****
 :*SUBTEST SUBR DETAILED ERROR REPORT
 :*****

2646 071364 005237 002326
 2647 071370 022737 000003 002326
 2648 071376 101465
 2649 071400 022737 000002 002326
 2650 071406 001435
 2651 071410
 2652 071420
 2653 071426 005037 002116
 2654 071432 010037 002306
 2655 071436 012700 002310
 2656 071442 010120
 2657 071444 010220
 2658 071446 010320
 2659 071450 010420
 2660 071452 010520
 2661 071454 013720 002030
 2662 071460 013720 002034
 2663 071464 013700 002306
 2664 071470
 2665 071476 104013
 2666 071500 000415
 2667 071502
 2668 071512
 2669 071520 005037 002116
 2670 071524
 2671 071532 104031
 2672 071534
 2673
 2674 071544 004737 071364
 2675 071550 000207

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
 1\$: POP MUT,HEADER
 ;WARNING RECURSIVE
 CALL DETAIL
 RETURN

```
2678                                     ;SIMULATE CONTROL 'T'  
2679 071552 004737 073404      4$: CALL      CONTT                ;DISPLAY 'DISPLAY' INFO  
2680  
2681                                     ;TYPE CONTENTS OF ALL CSR'S  
2682 071556 PUSH      CSR,CSR+2,CSRNO,R1  
2683 071574 TYPE      MSG058  
2684 071600 TYPE      $CRLF  
2685 071604 013701 002334      MOV      MKCSRS,R1  
2686 071610 BEGIN DUMPCSRLOOP  
2687 071610 FOR CSRNO := #0 TO #34 BY #4  
2688 071614 106301      ASLB      R1  
2689 071616 ON.ERROR  
2690 071620 104426      READCSR  
2691 071622      TYPOCT CSR  
2692 071630      TYPE      MSG018          ;2 SPACES  
2693 071634      TYPOCT CSR+2  
2694 071642      TYPE      MSG018          ;2 SPACES  
2695 071646 END ;OF ON.ERROR  
2696 071646 IFB R1 EQ #0 THEN LEAVE DUMPCSRLOOP  
2697 071652 END ;OF FOR CSRNO  
2698 071670 END DUMPCSRLOOP  
2699 071670 POP      R1,CSRNO,CSR+2,CSR
```

```

2702                                     ;TYPE STACKS
2703 071706 PUSH R0,R1
2704
2705                                     ;TYPE KERNEL STACK
2706 071712 TYPE MSG088 ;KERNEL STACK
2707 071716 013701 002706 MOV KSTACK,R1
2708 071722 162701 000002 SUB #2,R1
2709 071726 FOR R0 := SP TO R1 BY #2
2710 071730 TYPE $CRLF
2711 071734 TYPOCT R0
2712 071740 TYPE MSG018 ;2 SPACES
2713 071744 TYPOCT (R0)
2714 071750 END ;OF FOR R0
2715
2716                                     ;TYPE SUPERVISOR STACK
2717 071760 042737 030000 177776 BIC #BIT13!BIT12,PSW ;SET PREVIOUS MODE TO SUPERVISOR
2718 071766 052737 010000 177776 BIS #BIT12,PSW
2719 071774 006506 MFPI SSP
2720 071776 POP R0
2721 072000 TYPE MSG089 ;SUPERVISOR STACK
2722 072004 IF R0 LT #SUPSTK
2723 072012 FOR R0 := R0 TO #SUPSTK-2 BY #2
2724 072012 TYPE $CRLF
2725 072016 TYPOCT R0
2726 072022 TYPE MSG018 ;2 SPACES
2727 072026 TYPOCT (R0)
2728 072032 END ;OF FOR R0
2729 072044 ELSE
2730 072046 TYPE MSG091 ;IS EMPTY
2731 072052 END ;OF IF R0
2732
2733                                     ;TYPE USER STACK
2734 072052 052737 030000 177776 BIS #BIT13!BIT12,PSW ;SET PREVIOUS MODE TO USER
2735 072060 006506 MFPI USP
2736 072062 POP R0
2737 072064 TYPE MSG090 ;USER STACK
2738 072070 IF R0 LT #USESTK
2739 072076 FOR R0 := R0 TO #USESTK-2 BY #2
2740 072076 TYPE $CRLF
2741 072102 TYPOCT R0
2742 072106 TYPE MSG018 ;2 SPACES
2743 072112 TYPOCT (R0)
2744 072116 END ;OF FOR R0
2745 072130 ELSE
2746 072132 TYPE MSG091 ;IS EMPTY
2747 072136 END ;OF IF R0
2748 072136 TYPE $CRLF
2749 072142 005037 002326 CLR DETFLAG
2750 072146 POP R1,R0
2751 072152 000207 RETURN
    
```

2754 072154

QUE: SUBTST <<QUE MESSAGES FROM SLAVE CPU'S>>
 :*****
 :*SUBTEST QUE MESSAGES FROM SLAVE CPU'S
 :*****

2755 072154 104410
 2756 072156
 2757 072166 104424
 2758 072170
 2759 072174
 2760 072202 010001
 2761 072204 006201
 2762 072206 062701 000060
 2763 072212 110137 116242
 2764 072216
 2765 072222
 2766 072232
 2767 072236 005060 002626
 2768 072242 004737 057076
 2769 072246 062706 000002
 2770 072252 005760 002626
 2771 072256 100375
 2772 072260
 2773 072262
 2774 072264
 2775 072270 005060 002626
 2776 072274
 2777 072302
 2778 072310 104000
 2779 072312 005037 002570
 2780 072316
 2781 072316
 2782 072316
 2783 072316
 2784 072330
 2785 072340 162737 000001 002574
 2786 072346 000207

```

CKSWR
PUSH    R0,R1,CONTRL
CACHOFF                                ;TURN CACHE OFF
FOR R0 := #2 TO #6 BY #2
  IF PTYFLAG(R0) IS TRUE
    MOV     R0,R1
    ASR     R1
    ADD     #'0,R1
    MOVB    R1,MSGA98
    TYPE    MSG098
    IFB PTYBUF(R0) NE #32                ;SLAVEN>           ;WHERE N = 1,2, OR 3
    PUSH    PTYBUF(R0)                   ;CONTROL Z?
    CLR     PTYFLAG(R0)
    CALL    $TYPEC
    ADD     #2,SP                          ;FIX STACK
    TST     PTYFLAG(R0)
    BPL     1$
    GOTO    QUE1
  ELSE
    TYPE    MSG099                        ;MASTER>
    CLR     PTYFLAG(R0)
    IF RLFLAG IS TRUE
      SET    SLAERROR
      ERROR  ;LOG ERROR
      CLR    SLAERROR
    END ;OF IF RLFLAG IS TRUE
  END ;OF IF PTYBUF(R0)
END ;OF IF PTYFLAG(R0)
END ;OF FOR R0
POP      CONTRL,R1,R0
SUB      #1,HUNGTIME                      ;CARRY IS ERROR FLAG ON RETURN
RETURN
    
```

QUE1:

1\$:

```

2789          .SBTTL  ROUTINE BINARY TO OCTAL (ASCII) AND TYPE
2790
2791          ;*****
2792          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2793          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2794          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2795          ;*CALL:
2796          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
2797          ;*      TYPOS          ;;CALL FOR TYPEOUT
2798          ;*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2799          ;*      .BYTE   M              ;;M=1 OR 0
2800          ;*                                          ;;1=TYPE LEADING ZEROS
2801          ;*                                          ;;0=SUPPRESS LEADING ZEROS
2802
2803          ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2804          ;*$TYPOS OR $TYPOC
2805          ;*CALL:
2806          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
2807          ;*      TYPON          ;;CALL FOR TYPEOUT
2808
2809          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2810          ;*CALL:
2811          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
2812          ;*      TYPOC          ;;CALL FOR TYPEOUT
2813
2814 072350 017646 000000          $TYPOS: MOV      @(SP),-(SP)          ;;PICKUP THE MODE
2815 072354 116637 000001 072573  MOVVB   1(SP),$OFILL          ;;LOAD ZERO FILL SWITCH
2816 072362 112637 072575          MOVVB   (SP)+,$OMODE+1          ;;NUMBER OF DIGITS TO TYPE
2817 072366 062716 000002          ADD     #2,(SP)              ;;ADJUST RETURN ADDRESS
2818 072372 000406          BR      $TYPON
2819 072374 112737 000001 072573  $TYPOC: MOVVB   #1,$OFILL          ;;SET THE ZERO FILL SWITCH
2820 072402 112737 000006 072575  MOVVB   #6,$OMODE+1          ;;SET FOR SIX(6) DIGITS
2821 072410 112737 000005 072572  $TYPON: MOVVB   #5,$OCNT          ;;SET THE ITERATION COUNT
2822 072416 010346          MOV     R3,-(SP)            ;;SAVE R3
2823 072420 010446          MOV     R4,-(SP)            ;;SAVE R4
2824 072422 010546          MOV     R5,-(SP)            ;;SAVE R5
2825 072424 113704 072575          MOVVB   $OMODE+1,R4          ;;GET THE NUMBER OF DIGITS TO TYPE
2826 072430 005404          NEG     R4
2827 072432 062704 000006          ADD     #6,R4              ;;SUBTRACT IT FOR MAX. ALLOWED
2828 072436 110437 072574          MOVVB   R4,$OMODE          ;;SAVE IT FOR USE
2829 072442 113704 072573          MOVVB   $OFILL,R4          ;;GET THE ZERO FILL SWITCH
2830 072446 016605 000012          MOV     12(SP),R5          ;;PICKUP THE INPUT NUMBER
2831 072452 005003          CLR     R3                  ;;CLEAR THE OUTPUT WORD
2832 072454 006105          1$:    ROL     R5              ;;ROTATE MSB INTO 'C'
2833 072456 000404          BR      3$
2834 072460 006105          2$:    ROL     R5              ;;GO DO MSB
2835 072462 006105          ROL     R5                  ;;FORM THIS DIGIT
2836 072464 006105          ROL     R5
2837 072466 010503          MOV     R5,R3
2838 072470 006103          3$:    ROL     R3              ;;GET LSB OF THIS DIGIT
2839 072472 105337 072574          DECB   $OMODE              ;;TYPE THIS DIGIT?
2840 072476 100016          BPL    6$                  ;;BR IF NO
2841 072500 042703 177770          BIC    #177770,R3          ;;GET RID OF JUNK
2842 072504 001002          BNE    4$                  ;;TEST FOR 0
2843 072506 005704          TST    R4                  ;;SUPPRESS THIS 0?
2844 072510 001403          BEQ    5$                  ;;BR IF YES
2845 072512 005204          4$:    INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
    
```

2846	072514	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
2847	072520	052703	000040	5\$:	BIS	#' ,R3	::MAKE ASCII IF NOT ALREADY
2848	072524	110337	072570		MOVB	R3,8\$::SAVE FOR TYPING
2849	072530				TYPE	8\$::GO TYPE THIS DIGIT
2850	072534	105337	072572	6\$:	DECB	\$OCNT	::COUNT BY 1
2851	072540	003347			BGT	2\$::BR IF MORE TO DO
2852	072542	002402			BLT	7\$::BR IF DONE
2853	072544	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
2854	072546	000744			BR	2\$::GO DO THE LAST DIGIT
2855	072550	012605		7\$:	MOV	(SP)+,R5	::RESTORE R5
2856	072552	012604			MOV	(SP)+,R4	::RESTORE R4
2857	072554	012603			MOV	(SP)+,R3	::RESTORE R3
2858	072556	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
2859	072564	012616			MOV	(SP)+,(SP)	
2860	072566	000002			RTI		::RETURN
2861	072570	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
2862	072571	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
2863	072572	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
2864	072573	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
2865	072574	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

```

2867          .SBTTL  ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
2868          :*****
2869          :*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2870          :*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2871          :*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2872          :*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2873          :*REPLACED WITH SPACES.
2874          :*CALL:
2875          :*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
2876          :*      TYPDS  ;;GO TO THE ROUTINE
2877 072576     $TYPDS: PUSH  R0,R1,R2,R3,R5
2878 072610     MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
2879 072614     MOV      20(SP),R5        ;;GET THE INPUT NUMBER
2880 072620     BPL      1$              ;;BR IF INPUT IS POS.
2881 072622     NEG      R5              ;;MAKE THE BINARY NUMBER POS.
2882 072624     MOVVB   #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
2883 072632     1$:    CLR      R0        ;;ZERO THE CONSTANTS INDEX
2884 072634     MOV      #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
2885 072640     MOVVB   #' ,(R3)+        ;;SET THE FIRST CHARACTER TO A BLANK
2886 072644     2$:    CLR      R2        ;;CLEAR THE BCD NUMBER
2887 072646     MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
2888 072652     3$:    SUB      R1,R5     ;;FORM THIS BCD DIGIT
2889 072654     BLT      4$              ;;BR IF DONE
2890 072656     INC      R2              ;;INCREASE THE BCD DIGIT BY 1
2891 072660     BR      3$
2892 072662     4$:    ADD      R1,R5     ;;ADD BACK THE CONSTANT
2893 072664     TST      R2              ;;CHECK IF BCD DIGIT=0
2894 072666     BNE      5$              ;;FALL THROUGH IF 0
2895 072670     TSTB   (SP)              ;;STILL DOING LEADING 0'S?
2896 072672     BMI      7$              ;;BR IF YES
2897 072674     5$:    ASLB   (SP)        ;;MSD?
2898 072676     BCC      6$              ;;BR IF NO
2899 072700     MOVVB   1(SP),-1(R3)     ;;YES--SET THE SIGN
2900 072706     6$:    BIS      #'0,R2   ;;MAKE THE BCD DIGIT ASCII
2901 072712     7$:    BIS      #' ,R2   ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2902 072716     MOVVB   R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2903 072720     TST      (R0)+          ;;JUST INCREMENTING
2904 072722     CMP      R0,#10         ;;CHECK THE TABLE INDEX
2905 072726     BLT      2$              ;;GO DO THE NEXT DIGIT
2906 072730     BGT      8$              ;;GO TO EXIT
2907 072732     MOV      R5,R2          ;;GET THE LSD
2908 072734     BR      6$              ;;GO CHANGE TO ASCII
2909 072736     8$:    TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
2910 072740     BPL      9$              ;;BR IF NO
2911 072742     MOVVB   -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
2912 072750     9$:    CLRB   (R3)       ;;SET THE TERMINATOR
2913 072752     POP      R5,R3,R2,R1,R0
2914 072764     TYPE   $DBLK           ;;NOW TYPE THE NUMBER
2915 072770     MOV      2(SP),4(SP)     ;;ADJUST THE STACK
2916 072776     MOV      (SP)+,(SP)
2917 073000     RTI                    ;;RETURN TO USER
2918 073002     $DTBL: 10000.
2919 073004         1000.
2920 073006         100.
2921 073010         10.
2922 073012     $DBLK: .WORD 0,0,0,0
                073020 000000
  
```

```

2924 .SBTTL ROUTINE TTY INPUT
2925 :*****
2926 :*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2927 :*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2928 :*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2929 :*WHEN OPERATING IN TTY FLAG MODE.
2930 .ENABLE LSB
2931 $CKSWR: IF CPUBIT NE #MASTER ;SLAVES DON'T GET KEYBOARDS!
2932 RTI
2933 END ;OF IF CPUBIT
2934 TSTB @STKS ;:CHAR THERE?
2935 BPL 12$ ;:IF NO, DON'T WAIT AROUND
2936 MOVB @STKB,-(SP) ;:SAVE THE CHAR
2937 BIC #^C177,(SP) ;:STRIP-OFF THE ASCII
2938 CONTS1: CMP #6,(SP) ;:IS IT CONTROL F?
2939 BNE 1$ ;:NO SKIP
2940 CALL FIELDSERVICE
2941 1$: CMP #24,(SP) ;:IS IT CONTROL T?
2942 BNE 16$ ;:NO - SKIP
2943 CALL CONTT ;:YES - CALL CONTROL T ROUTINE
2944 16$: CMP #3,(SP) ;:IS IT CONTROL C?
2945 BEQ 5$ ;:YES EXIT *****NOTE***** STACK IS SCREWED UP!
2946 CMP #4,(SP) ;:IS IT CONTROL D?
2947 BNE 2$ ;:NO - SKIP
2948 CONTP: JMP @#0.ODT ;:YES - GO TO ODT
2949 2$: CMP #23,(SP) ;:IS IT CONTROL S?
2950 BNE 6$ ;:NO - SKIP
2951 CALL CONTS ;:YES - CALL CONTROL S ROUTINE
2952 6$: CMP #SWREG,SWR ;:IS THE SOFT-SWR SELECTED?
2953 BNE CKEND ;:BRANCH IF NO
2954 CMP #7,(SP)+ ;:IS IT A CONTROL G?
2955 BNE CKEND ;:NO, RETURN TO USER
2956 TST $AUTO ;:ARE WE RUNNING IN AUTO-MODE?
2957 BNE CKEND ;:BRANCH IF YES
2958 TYPE $CNTLG ;:ECHO THE CONTROL-G (^G)
2959 $GTSWR: TYPE $MSWR ;:TYPE CURRENT CONTENTS
2960 TYPOCT @SWR ;:OF THE SWR
2961 TYPE $MNEW ;:PROMPT FOR NEW SWR
2962 3$: CLR -(SP) ;:CLEAR COUNTER
2963 CLR -(SP) ;:THE NEW SWR
2964 4$: TSTB @STKS ;:CHAR THERE?
2965 BPL 4$ ;:IF NOT TRY AGAIN
2966 MOVB @STKB,-(SP) ;:PICK UP CHAR
2967 BIC #^C177,(SP) ;:MAKE IT 7-BIT ASCII
2968 CMP (SP),#3 ;:IS IT A CONTROL-C?
2969 BNE 7$ ;:BRANCH IF NOT
2970 5$: TYPE $CNTLC ;:YES, ECHO CONTROL-C (^C)
2971 ADD #6,SP ;:CLEAN UP STACK
2972 JMP BOOT ;:CONTROL-C RESTART
2973 7$: CMP (SP),#25 ;:IS IT A CONTROL-U?
2974 BNE 9$ ;:BRANCH IF NOT
2975 TYPE $CNTLU ;:YES, ECHO CONTROL-U (^U)
2976 8$: ADD #6,SP ;:IGNORE PREVIOUS INPUT
2977 BR 3$ ;:LET'S TRY IT AGAIN
2978 9$: CMP (SP),#15 ;:IS IT A <CR>?
2979 BNE 13$ ;:BRANCH IF NO
2980 TST 4(SP) ;:YES, IS IT THE FIRST CHAR?
  
```

2981	073274	001403			BEQ	10\$::BRANCH IF YES
2982	073276	016677	000002	107462	MOV	2(SP),@SWR	::SAVE NEW SWR
2983	073304	062706	000006		10\$: ADD	#6,SP	::CLEAR UP STACK
2984	073310				TYPE	\$CRLF	::ECHO <CR> AND <LF>
2985	073314	000002			12\$: RTI		::RETURN
2986	073316	062706	000002		CKEND: ADD	#2,SP	::FIX STACK
2987	073322	000002			RTI		::RETURN
2988	073324	004737	057076		13\$: CALL	\$TYPEC	::ECHO CHAR
2989	073330	021627	000060		CMP	(SP),#60	::CHAR < 0?
2990	073334	002420			BLT	15\$::BRANCH IF YES
2991	073336	021627	000067		CMP	(SP),#67	::CHAR > 7?
2992	073342	003015			BGT	15\$::BRANCH IF YES
2993	073344	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
2994	073350	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
2995	073354	001403			BEQ	14\$::BRANCH IF YES
2996	073356	006316			ASL	(SP)	::NO, SHIFT PRESENT
2997	073360	006316			ASL	(SP)	:: CHAR OVER TO MAKE
2998	073362	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
2999	073364	005266	000002		14\$: INC	2(SP)	::KEEP COUNT OF CHAR
3000	073370	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
3001	073374	000702			BR	4\$::GET THE NEXT ONE
3002	073376				15\$: TYPE	\$QUES	::TYPE ?<CR><LF>
3003	073402	000724			BR	8\$::SIMULATE CONTROL-U
3004					.DSABL	LSB	

3007 073404

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

3008 073404
3009 073406
3010 073412
3011
3012 073430 013700 002676
3013 073434 005400
3014 073436
3015 073442
3016 073450
3017 073452
3018 073460
3019 073464
3020 073464
3021 073470
3022 073500
3023 073504 013700 177750
3024 073510 006200
3025 073512 042700 177770
3026 073516
3027 073524
3028 073530
3029 073540
3030 073540
3031 073542 000207
3032
3033 073544

PUSH R0
TYPE \$CRLF
IF MDISPLAY LT #0 AND MDISPLAY GT #-256.
:WE ARE IN MULTIPROCESSOR MODE
MOV MDISPLAY,R0
NEG R0
TYPE MSG100 ;MULTIPROCESSOR TEST #
TYPOCS R0,,3 ;TYPE 3 DIGITS
ELSE
IF RLFLAG IS TRUE
TYPE MSG092 ;RELOCATED
END ;OF IF RLFLAG
TYPE MSG093 ;BANK=
TYPOCS BANK,,3 ;TYPE 3 DIGITS
TYPE MSG094 ;MAR=
MOV MAINT,R0
ASR R0
BIC #^C7,R0
TYPOCS R0,,1 ;TYPE 1 DIGIT
TYPE MSG095 ;PAT=
TYPOCS REALPAT,,2 ;TYPE 2 DIGITS
END ;OF IF @DISPLAY
POP R0
RETURN

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

3034 073544
3035 073546 105777 107220
3036 073552 100375
3037 073554 117716 107214
3038 073560 042716 177600
3039 073564
3040 073572 000137 073052
3041 073576
3042 073600 000762
3043 073602

POP R0 ;GET RID OF RETURN ADDRESS FROM STACK
CONTS2: TSTB @STKS ;WAIT FOR CHARACTER
BPL CONTS2
MOVB @STKB,(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)

```

3045
3046
3047
3048
3049
3050
3051
3052
3053 073602 011646
3054 073604 016666 000004 000C02
3055 073612 105777 107154
3056 073616 100375
3057 073620 117766 107150 000004
3058 073626 042766 177600 000004
3059 073634 026627 000004 000023
3060 073642 001013
3061 073644 105777 107122
3062 073650 100375
3063 073652 117746 107116
3064 073656 042716 177600
3065 073662 022627 000021
3066 073666 001366
3067 073670 000750
3068 073672 026627 000004 000140
3069 073700 002407
3070 073702 026627 000004 000175
3071 073710 003003
3072 073712 042766 000040 000004
3073 073720 000002
3074
3075
3076
3077
3078
3079
3080 073722 010346
3081 073724 005046
3082 073726 012703 074200
3083 073732 022703 074224
3084 073736 101467
3085 073740 104411
3086 073742 112613
3087 073744 122713 000003
3088 073750 001006
3089 073752
3090 073756 005726
3091 073760 012603
3092 073762 000137 043416
3093 073766 122713 000177
3094 073772 001022
3095 073774 005716
3096 073776 001007
3097 074000 112737 000134 074176
3098 074006
3099 074012 012716 177777
3100 074016 005303
3101 074020 020327 074200

*****
*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     @$TKB,4(SP)    ::READ THE TTY
        BIC      #^C<177>,4(SP) ::GET RID OF JUNK IF ANY
        CMP      4(SP),#23     ::IS IT A CONTROL-S?
        BNE      3$            ::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     $CNTLC       ::TYPE A CONTROL-C (^C)
        TST      (SP)+        ::CLEAN RUBOUT KEY OFF OF THE STACK
        MOV      (SP)+,R3     ::RESTORE R3
        JMP      BOOT         ::GOTO CONTROL-C RESTART
3$:     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
        TYPE     10$
4$:     MOV      #-1,(SP)     ::SET THE RUBOUT KEY
        DEC      R3           ::BACKUP BY ONE
        CMP      R3,$$TTYIN  ::STACK EMPTY?
    
```

3102	074024	103434			BLO	8\$::BR IF YES
3103	074026	111337	074176		MOVB	(R3),10\$::SETUP TO TYPEOUT THE DELETED CHAR.
3104	074032				TYPE	10\$::GO TYPE
3105	074036	000735			BR	2\$::GO READ ANOTHER CHAR.
3106	074040	005716		5\$:	TST	(SP)	::RUBOUT KEY SET?
3107	074042	001406			BEQ	6\$::BR IF NO
3108	074044	112737	000134	074176	MOVB	#'\,10\$::TYPE A BACK SLASH
3109	074052				TYPE	10\$	
3110	074056	005016			CLR	(SP)	::CLEAR THE RUBOUT KEY
3111	074060	122713	000025	6\$:	CMPB	#25,(R3)	::IS CHARACTER A CTRL U?
3112	074064	001003			BNE	7\$::BR IF NO
3113	074066				TYPE	\$CNTLU	::TYPE A CONTROL 'U'
3114	074072	000715			BR	1\$::GO START OVER
3115	074074	122713	000022	7\$:	CMPB	#22,(R3)	::IS CHARACTER A '^R'?
3116	074100	001011			BNE	9\$::BRANCH IF NO
3117	074102	105013			CLRB	(R3)	::CLEAR THE CHARACTER
3118	074104				TYPE	\$CRLF	::TYPE A 'CR' & 'LF'
3119	074110				TYPE	\$TTYIN	::TYPE THE INPUT STRING
3120	074114	000706			BR	2\$::GO PICKUP ANOTHER CHACTER
3121	074116			8\$:	TYPE	\$QUES	::TYPE A '?'
3122	074122	000701			BR	1\$::CLEAR THE BUFFER AND LOOP
3123	074124	111337	074176	9\$:	MOVB	(R3),10\$::ECHO THE CHARACTER
3124	074130				TYPE	10\$	
3125	074134	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
3126	074140	001274			BNE	2\$::LOOP IF NOT RETURN
3127	074142	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
3128	074146				TYPE	\$LF	::TYPE A LINE FEED
3129	074152	005726			TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
3130	074154	012603			MOV	(SP)+,R3	::RESTORE R3
3131	074156	011646			MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
3132	074160	016666	000004	000002	MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
3133	074166	012766	074200	000004	MOV	#\$TTYIN,4(SP)	
3134	074174	000002			RTI		::RETURN
3135	074176	000		10\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
3136	074177	000			.BYTE	0	::TERMINATOR
3137	074200	000024			\$TTYIN: .REPT	20.	::RESERVE SIZE BYTES FOR TTY INPUT
3140	074224	136	103	015	\$CNTLC: .ASCIZ	/^C/<15><12>	::CONTROL 'C'
	074227	012	000				
3141	074231	136	125	015	\$CNTLU: .ASCIZ	/^U/<15><12>	::CONTROL 'U'
	074234	012	000				
3142	074236	136	107	015	\$CNTLG: .ASCIZ	/^G/<15><12>	::CONTROL 'G'
	074241	012	000				
3143	074243	015	012	123	\$MSWR: .ASCIZ	<15><12>/SWR = /	
	074246	127	122	040			
	074251	075	040	000			
3144	074254	040	040	116	\$MNEW: .ASCIZ	/ NEW = /	
	074257	105	127	040			
	074262	075	040	000			
3145					.EVEN		

3147
 3148
 3149
 3150
 3151
 3152
 3153
 3154
 3155
 3156
 3157
 3158
 3159 074266 011646
 3160 074270 016666 000004 000002
 3161 074276
 3162 074304 104412
 3163 074306 012600
 3164 074310 010037 074414
 3165 074314 005001
 3166 074316 005002
 3167 074320 112046
 3168 074322 001420
 3169 074324 122716 000060
 3170 074330 003026
 3171 074332 122716 000067
 3172 074336 002423
 3173 074340 006301
 3174 074342 006102
 3175 074344 006301
 3176 074346 006102
 3177 074350 006301
 3178 074352 006102
 3179 074354 042716 177770
 3180 074360 062601
 3181 074362 000756
 3182 074364 005726
 3183 074366 010166 000012
 3184 074372 010237 074434
 3185 074376
 3186 074404 000002
 3187 074406 005726
 3188 074410 105010
 3189 074412
 3190 074414 000000
 3191 074416
 3192 074422
 3193 074426
 3194 074432 000724
 3195 074434 000000
 3196
 3197
 3198
 3199
 3200
 3201
 3202
 3203

```

.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$:      TST      (SP)+        ;;RETURN
CLR      (R0)              ;;CLEAN PARTIAL FROM STACK
TYPE     (R0)              ;;SET A TERMINATOR
5$:      .WORD    0           ;;TYPE UP THRU THE BAD CHAR.
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
    
```

```

3204      ;*POSITIVE 32767 TO NEGATIVE 32768.
3205      ;*CALL:
3206      ;*      RDDEC          ;;READ A DECIMAL NUMBER
3207      ;*      RETURN HERE   ;;NUMBER IS ON TOP OF THE STACK
3208      ;
3209
3210 074436 011646          $RDDEC: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR
3211 074440 016666 000004 000002 MOV      4(SP),2(SP)  ;;THE INPUT NUMBER
3212 074446          PUSH      R0,R1,R2
3213 074454 104412          1$:  RDLIN          ;;READ AN ASCII LINE
3214 074456 012600          MOV      (SP)+,R0          ;;ADDRESS OF 1ST CHAR.
3215 074460 010037 074604  MOV      R0,6$          ;;SAVE INCASE OF BAD INPUT
3216 074464 005046          CLR      -(SP)          ;;CLEAR DATA WORD
3217 074466 005002          CLR      R2          ;;SIGN SET POSITIVE
3218 074470 122710 000055  CMPB     #'-',(R0)      ;;SEE IF A MINUS SIGN WAS TYPED
3219 074474 001001          BNE     2$          ;;BR IF NO MINUS SIGN
3220 074476 112002          MOVB    (R0)+,R2      ;;SAVE FOR LATER USE
3221 074500 112001          2$:  MOVB    (R0)+,R1      ;;PICKUP THIS CHARACTER
3222 074502 001424          BEQ     3$          ;;GET OUT IF ZERO
3223 074504 122701 000060  CMPB     #'0,R1        ;;MAKE SURE THIS CHARACTER
3224 074510 003032          BGT     5$          ;;IS A DIGIT BETWEEN 0 & 9
3225 074512 122701 000071  CMPB     #'9,R1
3226 074516 002427          BLT     5$
3227 074520 032716 170000  BIT      #'C7777,(SP)  ;;DON'T LET NUMBER GET TO BIG
3228 074524 001024          BNE     5$          ;;BR IF NUMBER WOULD OVERFLOW
3229 074526 006316          ASL     (SP)          ;;*2
3230 074530 011646          MOV     (SP),-(SP)    ;;SAVE FOR LATER
3231 074532 006316          ASL     (SP)          ;;*4
3232 074534 006316          ASL     (SP)          ;;*8
3233 074536 062616          ADD     (SP)+,(SP)    ;;*10
3234 074540 102416          BVS     5$          ;;OVERFLOW ISN'T ALLOWED
3235 074542 162701 000060  SUB     #'0,R1        ;;STRIP AWAY THE ASCII JUNK
3236 074546 060116          ADD     R1,(SP)      ;;ADD IN THIS DIGIT
3237 074550 102412          BVS     5$          ;;OVERFLOW ISN'T ALLOWED
3238 074552 000752          BR      2$          ;;LOOP
3239 074554 005702          3$:  TST     R2          ;;CHECK IF NUMBER IS NEG
3240 074556 001401          BEQ     4$          ;;BR IF NO
3241 074560 005416          NEG     (SP)          ;;YES--NEGATE THE NUMBER
3242 074562 012666 000012  4$:  MOV     (SP)+,12(SP) ;;SAVE THE RESULT
3243 074566          POP     R2,R1,R0
3244 074574 000002          RTI          ;;RETURN
3245
3246 074576 005726          5$:  TST     (SP)+      ;;CLEAN PARTIAL NUMBER FROM STACK
3247 074600 105010          CLRB   (R0)          ;;SET A TERMINATOR
3248 074602          TYPE          ;;TYPE THE INPUT UP TO BAD CHAR.
3249 074604 000000          6$:  .WORD   0          ;;POINTER GOES HERE
3250 074606          TYPE   MSG062      ;;INPUT MUSST BE A
3251 074612          TYPE   MSG065      ;;DECIMAL
3252 074616          TYPE   MSG064      ;;NUMBER
3253 074622 000714          BR      1$          ;;TRY AGAIN

```

```

3255                               .SBTTL  ROUTINE SAVE AND RESTORE R0-R5
3256
3257                               ;*****
3258                               ;*SAVE R0-R5
3259                               ;*CALL:
3260                               ;*   SAVREG
3261                               ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
3262                               ;*
3263                               ;*TOP---(+16)
3264                               ;* +2---(+18)
3265                               ;* +4---R5
3266                               ;* +6---R4
3267                               ;* +8---R3
3268                               ;*+10---R2
3269                               ;*+12---R1
3270                               ;*+14---R0
3271
3272 074624   $SAVREG:
3273 074624   PUSH   R0,R1,R2,R3,R4,R5
3274 074640   016646 000022   MOV    22(SP),-(SP)   ;;SAVE PS OF MAIN FLOW
3275 074644   016646 000022   MOV    22(SP),-(SP)   ;;SAVE PC OF MAIN FLOW
3276 074650   016646 000022   MOV    22(SP),-(SP)   ;;SAVE PS OF CALL
3277 074654   016646 000022   MOV    22(SP),-(SP)   ;;SAVE PC OF CALL
3278 074660   000002   RTI
3279
3280                               ;*RESTORE R0-R5
3281                               ;*CALL:
3282                               ;*   RESREG
3283 074662   $RESREG:
3284 074662   012666 000022   MOV    (SP)+,22(SP)   ;;RESTORE PC OF CALL
3285 074666   012666 000022   MOV    (SP)+,22(SP)   ;;RESTORE PS OF CALL
3286 074672   012666 000022   MOV    (SP)+,22(SP)   ;;RESTORE PC OF MAIN FLOW
3287 074676   012666 000022   MOV    (SP)+,22(SP)   ;;RESTORE PS OF MAIN FLOW
3288 074702   POP    R5,R4,R3,R2,R1,R0
3289 074716   000002   RTI
  
```

```

3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302 074720
3303 074726 013700 002716
3304 074732 013701 002714
3305 074736 012702 000007
3306 074742 006300
3307 074744 006101
3308 074746 077203
3309 074750 063700 002716
3310 074754 005501
3311 074756 063701 002714
3312 074762 062700 001057
3313 074766 005501
3314 074770 062701 047401
3315 074774 010037 002716
3316 075000 010137 002714
3317 075004
3318 075012 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
  
```


TABLES

```

3363          .SBTTL  TABLES
3364
3365          .SBTTL  APT MAILBOX-ETABLE
3366 075134    $MAIL:
3367 075134    $MSGTY: .WORD 0      ;;MESSAGE TYPE CODE
3368 075136    $FATAL: .WORD 0      ;;FATAL ERROR NUMBER (ERROR PC)
3369 075140    $TESTN: .WORD 0      ;;TEST NUMBER (DISPLAY REGISTER)
3370 075142    $PASS:  .WORD 0      ;;PASS COUNT
3371 075144    $DEVCT: .WORD 0      ;;DEVICE COUNT
3372 075146    $UNIT:  .WORD 0      ;;I/O UNIT NUMBER
3373 075150    $MSGAD: .WORD 0      ;;MESSAGE ADDRESS
3374 075152    $MSGLG: .WORD 0      ;;MESSAGE LENGTH
3375 075154    $ETABLE:
3376 075154      $ENV:  .BYTE 0      ;;ENVIRONMENT BYTE      ;SET TO A 1 FOR APT AUTO MODE
3377 075155      $ENVM: .BYTE 0      ;;ENVIRONMENT MODE
3378
3379 075156    $SWREG: .WORD 1      ;;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
3380 075160    $USWR:  .WORD 0      ;;APT SWITCH REGISTER
3381 075162    $CPUOP: .WORD 30400  ;;USED TO LIMIT THE NUMBER OF PASSES
3382
3383          ;;CPU TYPE,OPTIONS
3384          ;;BITS 15-11=CPU TYPE
3385          ;;          11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
3386          ;;          11/70=06,PDQ=07,Q=10
3387          ;;BIT 10=REAL TIME CLOCK
3388          ;;BIT 9=FLOATING POINT PROCESSOR
3389          ;;BIT 8=MEMORY MANAGEMENT
3390          ;;HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
3391          ;;MEM. TYPE,BLK#1
3392          MEM.TYPE BYTE  -- (HIGH BYTE)
3393          900 NSEC CORE=001
3394          300 NSEC BIPOLAR=002
3395          500 NSEC MOS=003
3396          MK11 ERROR CORRECTING MOS=004
3397 075166    $MADR1: .WORD 177776 ;;HIGH ADDRESS,BLK#1
3398          ;;MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
3399 075170      $MAMS2: .BYTE 0      ;;HIGH ADDRESS,M.S. BYTE
3400 075171      $MTYP2: .BYTE 0      ;;MEM. TYPE,BLK#2
3401 075172      $MADR2: .WORD 0      ;;MEM.LAST ADDRESS,BLK#2
3402 075174      $MAMS3: .BYTE 0      ;;HIGH ADDRESS,M.S.BYTE
3403 075175      $MTYP3: .BYTE 0      ;;MEM. TYPE,BLK#3
3404 075176      $MADR3: .WORD 0      ;;MEM.LAST ADDRESS,BLK#3
3405 075200      $MAMS4: .BYTE 0      ;;HIGH ADDRESS,M.S.BYTE
3406 075201      $MTYP4: .BYTE 0      ;;MEM. TYPE,BLK#4
3407 075202      $MADR4: .WORD 0      ;;MEM.LAST ADDRESS,BLK#4
3408 075204      $VECT1: .WORD 0      ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
3409 075206      $VECT2: .WORD 0      ;;INTERRUPT VECTOR#2BUS PRIORITY#2
3410 075210      $BASE:  .WORD 0      ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
3411 075212      $DEVM:  .WORD 0      ;;DEVICE MAP
3412
3413          ;IF EITHER OF THESE 2 WORDS ARE NON ZERO
3414          ;THEY WILL BE LOADED INTO ALL CSR'S & THE PROGRAM WILL STOP
3415          $CDW1:  .WORD 0      ;CSR
3416          $CDW2:  .WORD 0      ;CSR+2
3417
3418          ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
3419          ;ARE TO BE RUN FOR PARTICULAR MEMORIES
3420
3421          ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.

```

```
3420 :BITO SET WILL RUN THE FIRST ENTRY IN THE TABLE, BITO SET
3421 :IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
3422 :
3423 :NOTE** NULL TESTS DO NOT TAKE ANY TIME
3424 :
3425 075220 177777 $DDW0: .WORD 177777 :MK11 CSR TESTS TABLE = MKCSRT:
3426 075222 177777 $DDW1: .WORD 177777 :MK11 CSR TESTS TABLE = MKCSRT:
3427 075224 102777 $DDW2: .WORD 102777 :MK11 PATTERNS TABLE = MKPAT:
3428 075226 177777 $DDW3: .WORD 177777 :MK11 PATTERNS TABLE = MKPAT:
3429 075230 103777 $DDW4: .WORD 103777 :MJ11 PATTERNS TABLE = MJPAT:
3430 075232 177777 $DDW5: .WORD 177777 :MJ11 PATTERNS TABLE = MJPAT:
3431 075234 000001 $DDW6: .WORD 1 :NUMBER OF CPU'S
3432 075236 $ETEND:
3433 :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
3434 :INTERFACE SPEC.
3435
3436 075236 $APTHD:
3437 075236 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
3438 075240 075134 $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
3439 075242 000010 $TSTM: .WORD 8. ;;RUN TIM OF LONGEST TEST
3440 075244 000144 $PASTM: .WORD 100. ;;RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
3441 075246 000310 $UNITM: .WORD 200. ;;ADDITIONAL RUN TIME OF A PASS FOR EACH ADDITIONAL 128K
3442 075250 000041 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE (WORDS)
```

.SBTTL ROUTINE TRAP DECODER

3444
 3445
 3446
 3447
 3448
 3449
 3450
 3451
 3452
 3453
 3454
 3455
 3456
 3457
 3458
 3459
 3460
 3461
 3462
 3463
 3464
 3465
 3466
 3467
 3468

075252 010046
 075254 016600 000002
 075260 005740
 075262 111000
 075264 006300
 075266 016000 075314
 075272 000200
 075274 011646
 075276 016666 000004 000002
 075304 000002
 075306
 075312 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
    
```

3471
3472
3473
3474
3475
3476
3477
3478 075314 075274
3479 075316 056752
3480 075320 072374
3481 075322 072350
3482 075324 075306
3483 075326 072576
3484 075330 075306
3485
3486 075332 073160
3487 075334 073022
3488
3489 075336 073602
3490 075340 073722
3491 075342 074266
3492 075344 074436
3493
3494 075346 074624
3495 075350 074662
3496
3497 075352 037020
3498 075354 037030
3499 075356 037040
3500
3501 075360 041016
3502
3503 075362 037050
3504 075364 037066
3505
3506 075366 037076
3507 075370 037220
3508
3509 075372 064574
3510 075374 064622
3511 075376 064650
3512 075400 064700
3513 075402 064762
3514 075404 065004
3515 075406 065034
3516 075410 065054
3517 075412 065076
3518 075414 065116
3519 075416 065140
3520 075420 065162
3521 075422 065202
3522 075424 065220
3523 075426 065236
3524 075430 065256
3525 075432 065274
3526 075434 065312
3527 075436 054570

.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 TIMEOUT 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:\$TYPON :CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS :CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
\$NOTRAP:\$TYPBN :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
3528	075440	065502	\$PER26	:CALL=PERR26 TRAP+52(104452) PROGRAM DETECTED ERROR
3529	075442	065522	\$PER27	:CALL=PERR27 TRAP+53(104453) PROGRAM DETECTED ERROR
3530	075444	054762	\$PER30	:CALL=PERR30 TRAP+54(104454) PROGRAM DETECTED ERROR
3531	075446	065720	\$PER31	:CALL=PERR31 TRAP+55(104455) PROGRAM DETECTED ERROR
3532	075450	066016	\$PER32	:CALL=PERR32 TRAP+56(104456) PROGRAM DETECTED ERROR
3533	075452	066064	\$PER33	:CALL=PERR33 TRAP+57(104457) PROGRAM DETECTED ERROR
3534	075454	066144	\$PER34	:CALL=PERR34 TRAP+60(104460) PROGRAM DETECTED ERROR
3535	075456	066176	\$PER35	:CALL=PERR35 TRAP+61(104461) PROGRAM DETECTED ERROR
3536	075460	075306	\$NOTRAP	:CALL=PERR36 TRAP+62(104462) PROGRAM DETECTED ERROR
3537	075462	075306	\$NOTRAP	:CALL=PERR37 TRAP+63(104463) PROGRAM DETECTED ERROR
3538	075464	075306	\$NOTRAP	:CALL=PERR40 TRAP+64(104464) PROGRAM DETECTED ERROR
3539	075466	075306	\$NOTRAP	:CALL=PERR41 TRAP+65(104465) PROGRAM DETECTED ERROR
3540	075470	075306	\$NOTRAP	:CALL=PERR42 TRAP+66(104466) PROGRAM DETECTED ERROR
3541	075472	075306	\$NOTRAP	:CALL=PERR43 TRAP+67(104467) PROGRAM DETECTED ERROR
3542				
3543	075474	037466	\$ECCDIS	:CALL=ECCDIS TRAP+70(104470) DISABLE ECC ON ALL CSR'S
3544	075476	037522	\$ECC1DIS	:CALL=ECC1DIS TRAP+71(104471) DISABLE ECC ON 1 SELECTED CSR
3545	075500	037554	\$ECCINIT	:CALL=ECCINIT TRAP+72(104472) INITIALIZE ALL MK11 CSR'S
3546	075502	037574	\$ECC1INIT	:CALL=ECC1INIT TRAP+73(104473) INITIALIZE 1 SELECTED MK11 CSR
3547	075504	037700	\$CBCSR	:CALL=CBCSR TRAP+74(104474) WRITE GENERATED CHECKBITS IN ALL CSR'S
3548	075506	037742	\$CB1CSR	:CALL=CB1CSR TRAP+75(104475) WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
3549	075510	040002	\$WASSBE	:CALL=WASSBE TRAP+76(104476) WAS THERE A SBE ON ANY CSR?
3550	075512	040116	\$WAS1SBE	:CALL=WAS1SBE TRAP+77(104477) WAS THERE A SBE ON 1 SELECTED CSR?
3551	075514	040146	\$WASDBE	:CALL=WASDBE TRAP+100(104500) WAS THERE A DBE ON ANY CSR?
3552	075516	040262	\$WAS1DBE	:CALL=WAS1DBE TRAP+101(104501) WAS THERE A DBE ON 1 SELECTED CSR?
3553	075520	040322	\$CLRCSR	:CALL=CLRCSR TRAP+102(104502) CLEAR ALL CSR'S
3554	075522	040340	\$CLR1CSR	:CALL=CLR1CSR TRAP+103(104503) CLEAR 1 SELECTED CSR
3555	075524	040354	\$CHKDIS	:CALL=CHKDIS TRAP+104(104504) DISABLE ECC & WRITE CKBITS FROM ALL CSR'S
3556	075526	040410	\$CHK1DIS	:CALL=CHK1DIS TRAP+105(104505) DISABLE ECC & WRITE CKBITS FROM 1 CSR
3557	075530	037612	\$ENASBE	:CALL=ENASBE TRAP+106(104506) ENABLE TRAPS ON SBE'S FROM ALL CSR'S
3558	075532	037646	\$ENA1SBE	:CALL=ENA1SBE TRAP+107(104507) ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
3559	075534	037310	\$TSTRD	:CALL=TSTREAD TRAP+110(104510) TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES
3560	075536	040510	\$INVALID	:CALL=INVALID TRAP+111(104511) INVALIDATE BACKGROUND PATTERN ON BANK
3561	075540	075306	\$NOTRAP	
3562	075542	075306	\$NOTRAP	
3563	075544	075306	\$NOTRAP	
3564	075546	075306	\$NOTRAP	
3565	075550	075306	\$NOTRAP	
3566	075552	075306	\$NOTRAP	
3567	075554	075306	\$NOTRAP	
3568	075556	075306	\$NOTRAP	

		.SBTTL	ODT (CEMKA) DATA AREA	
3571				
3572	177776	ST =	177776	:STATUS REGISTER
3573	000016	O.BKP =	16	:NUMBER OF BREAKPOINTS-1 MULT. BY 2
3574	000014	O.TVEC =	14	:TRT VECTOR LOCATION
3575	000340	O.STM =	340	:PRIORITY MASK - STATUS REGISTER
3576	000020	O.TBT =	20	:T-BIT MASK - STATUS REGISTER
3577	000003	TRT =	000003	:TRT INSTRUCTION
3578	177562	O.RDB =	177562	:R DATA BUFFER
3579	177560	O.RCSR =	177560	:R C/SR
3580	177566	O.TDB =	177566	:T DATA BUFFER
3581	177564	O.TCSR =	177564	:T C/SR

3585	075560	000000	O.CAD:	0	:	CURRENT ADDRESS
3586	075562	000000	O.DOT:	0	:	ORIGIN ADDRESS
3587	075564	000000	O.XXX:	.WORD 0	:	TEMPORARY STORAGE
3588	075566	000000	O.BW:	.WORD 0	:	=0 - ALL CLOSED
3589					:	=1 - BYTE OPEN,
3590					:	=2 - WORD OPEN
3591	075570	000	O.SEQ:	.BYTE 0	:	CHANGE SEQUENCE INDICATOR
3592	075571	000	O.S:	.BYTE 0	:	SINGLE INSTRUCTION FLAG
3593					:	0 IF NOT ACTIVE
3594					:	-1 IF ACTIVE
3595					:	NO BREAK POINTS MAY BE SET WHILE IN
3596					:	SINGLE INSTRUCTION MODE
3597	075572	000	O.T:	.BYTE 0	:	T-BIT FLAG
3598	075573	000	O.P:	.BYTE 0	:	PROCEED FLAG = -2 IF MANUAL ENTRY
3599					:	-1 IF NO PROCEED ALLOWED
3600					:	0-7 IF PCEED ALLOWED
3601	075574	000	O.CSR1:	.BYTE 0	:	SAVE CELL - R C/SR
3602	075575	000	O.CSR2:	.BYTE 0	:	SAVE CELL - T C/SR
3603	075576	000000	O.FIL:	0	:	FILL COUNTER
3604	075600	000	O.CMFD:	.BYTE 0	:	COMMA FOUND SWITCH, =0 NO COMMA FOUND
3605					:	=1 COMMA FOUND
3606	075601	000	O.SMFD:	.BYTE 0	:	SEMICOLON FOUND SWITCH
3607					:	=0 NO SEMICOLON FOUND
3608					:	=1 SEMICOLON FOUND
3609	075602	000	O.SCRN:	.BYTE 0	:	FLAG; 1=PASS SPACES ON FROM TTY
3610					:	ALSO, IF =1, <LF> IS ECHOED
3611	075603	000	O.MINS:	.BYTE 0	:	MINUS SIGN TYPED (SWITCH)
3612					:	0=NO MINUS TYPED; 1=MINUS SIGN TYPED
3613	075604	000000	O.BIAS:	.WORD 0	:	CURRENT RELOCATION BIAS


```

3642      : INITIALIZE ODT
3643      : USE O.ODT FOR A NORMAL ENTRY
3644      : USE O.ODT+2 TO RESTART ODT - WIPING OUT ALL BREAKPOINTS
3645      : USE O.ODT+4 TO RE-ENTER (I.E. - FAKE A BREAKPOINT)
3646
3647 075774 000421      0.ODT: BR      2$      ;NORMAL ENTRY
3648 075776 000440      BR      3$      ;RESTART
3649 076000 004737 101172 1$: JSR    PC,O.RRST ;RE-ENTER -- SAVE STATUS
3650 076004 013746 000016 MOV    O.TVEC+2,-(SP) ;SET UP LOCAL STATUS
3651 076010 004737 101202 JSR    PC,O.WST
3652 076014 010746      MOV    PC,-(SP)      ;FAKE THE PC
3653 076016 012637 075700 MOV    (SP)+,O.UPC
3654 076022 112737 177777 075573 MOVB  #-1,O.P      ;DISALLOW PROCEED
3655 076030 105037 075571 CLRB  O.S
3656 076034 000137 100060 JMP    O.BK1
3657
3658      ;RUN AT CURRENT STATUS
3659 076040 004537 100500 2$: JSR    5,O.SVTT ;SAVE TTY STATUS
3660 076044 012737 076000 073114 MOV    #1$,@#CONTP+2
3661 076052 004037 100360 JSR    O,O.SVR ;SAVE REGISTERS (MAINLY SP)
3662 076056 012706 075662 MOV    #O.UR0,SP ;SET UP STACK
3663 076062 012704 101300 MOV    #O.ID,R4 ;TYPE ID
3664 076066 012703 101315 MOV    #O.IDND,R3
3665 076072 004537 101126 JSR    5,O.TYPE
3666 076076 000414      BR      4$
3667 076100 004037 100360 3$: JSR    O,O.SVR ;SAVE REGISTERS
3668 076104 004537 100602 JSR    5,O.REM ;REMOVE ALL BREAKPOINTS
3669 076110 113704 075704 MOVB  O.PRI,R4 ;GET ODT PRIORITY
3670 076114 106004      RORB  R4 ;SHIFT
3671 076116 106004      RORB  R4 ;INTO
3672 076120 106004      RORB  R4 ;POSITION
3673 076122 010446      MOV    R4,-(SP) ;STORE IN STATUS
3674 076124 004737 101202 JSR    PC,O.WST
3675 076130 105037 075571 4$: CLRB  O.S ;DISABLE SINGLE INSTRUCTION FOR NOW
3676 076134 112737 177777 075573 MOVB  #-1,O.P ;DISALLOW PROCEED
3677 076142 012737 000340 000016 MOV    #O.STM,O.TVEC+2 ;STATUS WORD TO TRT VECTOR+2
3678 076150 012737 100042 000014 MOV    #O.BRK,O.TVEC ;PC TO TRT VECTOR
3679 076156 000137 077430 JMP    O.RALL ;CLEAR BRK PT TABLES
3680
3681      :O.CTLC
3682      : ^C PROCESSING. RETURN TO KEYBOARD MONITOR
3683      :
3684      :
3685 076162 013706 075676 0.OTLC: MOV    O.USP,SP ;RESTORE USER STACK
3686 076166 012704 076206 MOV    #1$,R4
3687 076172 012703 076207 MOV    #2$,R3
3688 076176 004537 101126 JSR    R5,O.TYPE ;ECHO '^C'
3689 076202 000137 043416 JMP    BOOT
3690 076206      136 1$: .BYTE '^'
3691 076207      103 2$: .BYTE 'C'
    
```

```

3694
3695 076210 013706 075676
3696 076214 012704 076240
3697 076220 012703 076241
3698 076224 004537 101126
3699 076230 004737 043774
3700 076234 000137 075774
3701 076240      136
3702 076241      106
3703
3704
3705 076242 012704 076266
3706 076246 012703 076267
3707 076252 004537 101126
3708 076256 012705 073116
3709 076262 000137 077624
3710 076266      136
3711 076267      105
3712
3713
3714
3715
3716 076270 004537 101006
3717 076274 012704 101345
3718 076300 120024
3719 076302 001414
3720 076304 022704 101365
3721 076310 101373
3722 076312 042700 177770
3723 076316 010004
3724 076320 006304
3725 076322 062704 075662
3726 076326 005202
3727 076330 000137 076554
3728 076334 162704 101336
3729 076340 000767
    
```

```

;CONTROL F PROCESSING
O.CTLF: MOV    O.USP,SP      ;RESTORE USER STACK
        MOV    #1$,R4
        MOV    #2$,R3
        JSR    R5,O.TYPE    ;ECHO ^F
        CALL   FIELDSERVICE
        JMP    O.ODT
1$:     .BYTE  'A
2$:     .BYTE  'F

;CONTROL E PROCESSING
O.CTLE: MOV    #1$,R4
        MOV    #2$,R3
        JSR    R5,O.TYPE    ;ECHO ^E
        MOV    #CONTP+4,R5
        JMP    O.GOGO
1$:     .BYTE  'A
2$:     .BYTE  'E

; SPECIAL NAME HANDLER
; DEPENDS UPON THE EXPLICIT ORDER OF THE TWO TABLES O.TL AND O.URO
O.REGT: JSR    5,O.GET      ;SPECIAL NAME, GET ONE MORE CHARACTER
        MOV    #O.TL,R4    ;TABLE START ADDRESS
1$:     CMPB   R0,(R4)+     ;IS THIS THE CORRECT CHARACTER?
        BEQ    3$          ;JUMP IF YES
        CMP    #O.TL+O.LG,R4 ;IS THE SEARCH DONE?
        BHI   1$          ;BRANCH IF NOT
        BIC   #177770,R0   ;MASK OFF OCTAL
        MOV    R0,R4
2$:     ASL    R4
        ADD   #O.URO,R4    ;GENERATE ADDRESS
        INC   R2          ;SET FOUND FLAG
        JMP   O.SCAN      ;GO FIND NEXT CHARACTER
3$:     SUB    #O.TL-7,R4  ;GO FIND NEXT CHARACTER
        BR    2$
    
```

```

3732      ; 'BACKARROW' HANDLER - OPEN INDEXED ON THE PC (BACK ARROW)
3733      .ENABL  LSB
3734 076342 004537 076416 0.ORPC: JSR 5,2$      ;TEST WORD MODE AND CLOSE
3735 076346 061202      ADD @R2,R2      ;COMPUTE
3736 076350 005202      INC R2
3737 076352 005202      INC R2      ; NEW ADDRESS
3738 076354 010237 075560 1$: MOV R2,0.CAD  ;UPDATE CAD
3739 076360 000137 077226      JMP 0.OP2A      ;GO FINISH UP
3740 076364 004537 076416 0.ORAB: JSR 5,2$      ;TEST WORD MODE AND CLOSE
3741 076370 011202      MOV @R2,R2      ;GET ABSOLUTE ADDRESS
3742 076372 000770      BR 1$
3743 076374 004537 076416 0.ORRB: JSR 5,2$      ;TEST AND CLOSE
3744 076400 011201      MOV @R2,R1      ;COMPUTE NEW ADDRESS
3745 076402 110101      MOVB R1,R1      ;EXTEND THE SIGN
3746 076404 006301      ASL R1          ;R2=2(@R2)
3747 076406 005201      INC R1          ; +2
3748 076410 005201      INC R1
3749 076412 060102      ADD R1,R2      ; +PC
3750 076414 000757      BR 1$
3751 076416 004737 101214 2$: JSR PC,0.CLSE  ;CLOSE CURRENT CELL
3752 076422 022737 000002 075566      CMP #2,0.BW    ;ONLY WORD MODE ALLOWED
3753 076430 001003      BNE 3$        ;BRANCH IF ERROR
3754 076432 013702 075560      MOV 0.CAD,R2  ;CURRENT ADDRESS IN R2
3755 076436 000205      RTS R5
3756 076440 005726      3$: TST (SP)+
3757 076442 000137 076510 0.TCL2: JMP 0.ERR      ;POP A WORD AND SHOW THE ERROR
3758      .DSABL  LSB
3759
3760      ;
3761 076446 105737 075601 0.SNGL: TSTB 0.SMFD  ;DONT REACT IF ; NOT TYPED
3762 076452 001773      BEQ 0.TCL2
3763 076454 005702      TST R2
3764 076456 001004      BNE 1$        ;SEE IF TURN ON OR TURN OFF
3765 076460 105037 075571      CLRB 0.S      ;BRANCH IF TURNING IT ON
3766 076464 000137 076520      JMP 0.DCD     ;CLEAR THE FLAG
3767 076470 112737 177777 075571 1$: MOVB #-1,0.S   ;CONTINUE THE SCAN
3768 076476 000137 076520      JMP 0.DCD     ;SET THE FLAG
    
```

```

3771
3772
3773
3774 076502 105237 075603
3775 076506 000420
3776
3777
3778
3779
3780
3781 076510 012700 000077
3782 076514 004537 100730
3783 076520 005037 075566
3784 076524 004537 101254
3785 076530 105037 075601
3786 076534 105037 075600
3787 076540 105037 075603
3788 076544 005003
3789 076546 005005
3790 076550 005004
3791 076552 005002
3792 076554 004537 101006
3793 076560 022700 000060
3794 076564 101013
3795 076566 022700 000067
3796 076572 103410
3797 076574 042700 177770
3798 076600 006304
3799 076602 006304
3800 076604 006304
3801 076606 060004
3802 076610 005202
3803 076612 000760
3804 076614 005001
3805 076616 120061 101321
3806 076622 001405
3807 076624 005201
3808 076626 020127 000024
3809 076632 103326
3810 076634 000770
3811 076636 105737 075603
3812 076642 001401
3813 076644 005404
3814 076646 105737 075600
3815 076652 001402
3816 076654 063704 075604
3817 076660 105037 075603
3818 076664 006301
3819 076666 000171 076672

```

```

:MINUS PROCESSING
O.MIN: INCB O.MINS ;SET MINUS FOUND SWITCH ON
BR O.DCD1

; COMMAND DECODER - ODT11X
; ALL REGISTERS MAY BE USED (R0-R5),
O.ERR: MOV #'?,R0 ; ? TO BE TYPED
JSR 5,O.FTYP ; OUTPUT ?
O.DCD: CLR O.BW ;CLOSE ALL
JSR 5,O.CRLS ;TYPE <CR><LF>*
O.DCD3: CLRB O.SMFD ;SEMICOLON FOUND FLAG
CLRB O.CMFD ;COMMA FOUND FLAG
CLRB O.MINS ;MINUS SIGN FOUND FLAG
CLR R3 ;R3 IS A SAVE REGISTER FOR R2
CLR R5 ;R5 IS A SAVE REGISTER FOR R4
O.DCD1: CLR R4 ; R4 CONTAINS THE CONVERTED OCTAL
CLR R2 ; R2 IS THE NUMBER FOUND FLAG
O.SCAN: JSR 5,O.GET ;GET A CHAR, RETURN IN R0
CMP #'0,R0 ;COMPARE WITH ASCII 0
BHI 5$ ;CHECK LEGALITY IF NON-NUMERIC
CMP #'7,R0 ;COMPARE WITH ASCII 7
BLO 5$ ;CHECK LEGALITY IF NOT OCTAL
BIC #177770,R0 ;CONVERT TO BCD
ASL R4 ; MAKE ROOM
ASL R4 ; IN
ASL R4 ; R4
ADD R0,R4 ;PACK THREE BITS IN R4
INC R2 ;R2 HAS NUMERIC FLAG
BR O.SCAN ; AND TRY AGAIN
5$: CLR R1 ;CLEAR INDEX
1$: CMPB R0,O.LGCH(R1) ;DO THE CODES MATCH?
BEQ 2$ ;JUMP IF YES
INC R1 ; SET INDEX FOR NEXT SEARCH
CMP R1,#O.CLGT ;IS THE SEARCH DONE?
BHI O.ERR ; OOPS!
BR 1$ ;RE-LOOP
2$: TSTB O.MINS ;IF MINUS WAS NOT TYPED
BEQ 4$ ;DO NOT NEGATE K
NEG R4 ;OTHERWISE, TAKE 2'S COMPLEMENT.
4$: TSTB O.CMFD ;IF A COMMA NOT TYPED,
BEQ 3$ ;SKIP NEXT INSTRUCTION.
ADD O.BIAS,R4 ;OTHERWISE, ADD RELOC. BIAS TO (R4)
3$: CLRB O.MINS ;REINITIALIZE MINUS-TYPED SWITCH FOR NXT SCAN
ASL R1 ;MULTIPLY BY TWO
JMP @6$(R1) ;GO TO PROPER ROUTINE

```

3822	076672	076742	6\$:	O.SEMI	:	:	SEMICOLON
3823	076674	076760		O.WRD	:	/	OPEN WORD
3824	076676	076772		O.BYT	:	\	OPEN BYTE -BACK SLASH
3825	076700	077140		O.CRET	:		CARRIAGE RETURN CLOSE
3826	076702	076270		O.REGT	:	\$	REGISTER OPS
3827	076704	077576		O.GO	:	G	GO TO ADDRESS K
3828	076706	077160		O.OP1	:	<LF>	MODIFY, CLOSE, OPEN NEXT
3829	076710	076342		O.ORPC	:	'BACKARROW'	OPEN RELATED, INDEX - PC (BACK ARROW)
3830	076712	077154		O.OLD	:	<	RETURN TO OLD SEQUENCE AND OPEN
3831	076714	077302		O.BACK	:	^	OPEN PREVIOUS (UP ARROW)
3832	076716	077466		O.OFST	:	O	OFFSET
3833	076720	077324		O.BKPT	:	B	BREAKPOINTS
3834	076722	077732		O.PROC	:	P	PROCEED
3835	076724	076364		O.ORAB	:	@	OPEN RELATED, ABSOLUTE
3836	076726	076374		O.ORRB	:	>	OPEN RELATED, REL. BRANCH
3837	076730	076446		O.SNGL	:	S	SINGLE INSTRUCTION MODE
3838	076732	076502		O.MIN	:	-	MINUS, NEGATES NUMBER TYPED IN
3839	076734	076162		O.CTLC	:	^C	EXIT TO MONITOR
3840	076736	076210		O.CTLF	:	^F	EXIT TO FIELD SERVICE MODE
3841	076740	076242		O.CTLE	:	^E	PROCEED FROM ^D

```

3844
3845      : SEMI-COLON PROCESSOR
3846
3847 076742 010203      0.SEMI: MOV      R2,R3      ;A SEMI-COLON HAS BEEN RECEIVED
3848 076744 010405      MOV      R4,R5      ;NUMERIC FLAG TO R3, CONTENTS TO R5
3849 076746 105237 075601      INCB     0.SMFD      ;SET SEMICOLON FOUND FLAG
3850 076752 105037 075600      CLRFB   0.CMFD      ;RESET COMMA FOUND FLAG
3851 076756 000674      BR       0.DCD1     ;GO BACK FOR MORE
3852
3853      : PROCESS / AND \ - OPEN WORD OR BYTE
3854
3855      :INPUT - IF R2 IS NON-ZERO A NEW RFXP HAS BEEN TYPED IN
3856      :INPUT - -ADDRESS OF WORD TO BE OPENED IS IN R4
3857      .ENABL  LSB
3858 076760 012737 000002 075566 0.WRD:  MOV      #2,0.BW      ;OPEN WORD
3859 076766 000404      BR       11$
3860 076770 006104      1$:     ROL      R4      ;GET THE ADDRESS BACK
3861 076772 012737 000001 075566 0.BYT:  MOV      #1,0.BW      ;OPEN BYTE
3862 077000 005702      11$:   TST      R2      ;GET VALUE IF R2 IS NON-ZERO
3863 077002 001011      BNE     14$      ;BRANCH IF NUMBER INPUT
3864 077004 105737 075600      TSTB   0.CMFD      ;TEST FOR ','AND';'
3865 077010 001402      BEQ     12$
3866 077012 000137 076510      13$:   JMP      0.ERR      ;ERROR IF PRESENT WITHOUT NUMBER.
3867 077016 105737 075601      12$:   TSTB   0.SMFD
3868 077022 001373      BNE     13$
3869 077024 000404      BR      0.WRD1     ;NO NUMBER - REOPEN PREVIOUS LOCATION
3870 077026 010437 075562      14$:   MOV      R4,0.DOT ;PUT VALUE IN DOT
3871 077032 010437 075560      MOV      R4,0.CAD  ;ALSO IN CAD
3872 077036 022737 000001 075566 0.WRD1: CMP      #1,0.BW      ;CHECK BYTE MODE
3873 077044 001407      BEQ     22$      ;JUMP IF BYTE
3874 077046 013704 075560      MOV      0.CAD,R4
3875 077052 006204      ASR     R4      ;MOVE ONE BIT TO CARRY
3876 077054 103745      BCS     1$      ;JUMP IF ODD ADDRESS
3877 077056 017700 176476      MOV      @0.CAD,R0 ;GET CONTENTS OF WORD
3878 077062 000402      BR      23$
3879 077064 117700 176470      22$:   MOVB   @0.CAD,R0   ;GET CONTENTS OF BYTE
3880 077070 004537 100634      23$:   JSR     5,0.CADV   ;GO GET AND TYPE OUT @CAD
3881 077074 022737 000001 075566      CMP      #1,0.BW      ;CHECK IF BYTE MODE.
3882 077102 001212      BNE     0.DCD3     ;IF NOT WE'RE DONE. ELSE:
3883 077104 112700 000075      MOVB   #'=,R0      ;TYP '=' AND THEN THE ASCII BYTE
3884 077110 004537 100730      JSR     5,0.FTYP
3885 077114 117700 176440      MOVB   @0.CAD,R0
3886 077120 004537 100730      JSR     5,0.FTYP
3887 077124 112700 000040      MOVB   #' ,R0
3888 077130 004537 100730      JSR     5,0.FTYP
3889 077134 000137 076530      JMP     0.DCD3     ;GO BACK TO DECODER
3890      .DSABL  LSB
    
```

```

3893          ; PROCESS CARRIAGE RETURN
3894 077140 004737 101214  O.CRET: JSR   PC,O.CLSE  ;CLOSE LOCATION
3895 077144 000137 076520  O.DCDA: JMP   O.DCD      ;RETURN TO DECODER
3896
3897 077150 000137 076510  O.ERR3: JMP   O.ERR      ; INTERMEDIATE HELP
3898
3899          ; PROCESS <LF>, OPEN NEXT WORD
3900
3901 077154 105237 075570  O.OLD:  INCB  O.SEQ      ;SET FLAG TO LATER RESTORE CAD
3902 077160 005737 075566  O.OP1:  TST   O.BW      ;<LF> RECEIVED
3903 077164 001771          O.ERR2: BEQ   O.ERR3    ;ERROR IF NOTHING IS OPEN
3904 077166 004737 101214          JSR   PC,O.CLSE      ;CLOSE PRESENT CELL
3905 077172 105737 075570          TSTB  O.SEQ      ;SHOULD CAD BE RESTORED?
3906 077176 001405          BEQ   1$          ;BRANCH IF NOT
3907 077200 013737 075562 075560  MOV   O.DOT,O.CAD    ;RESTORE PREVIOUS SEQUENCE
3908 077206 105037 075570          CLRB  O.SEQ      ;RESET FLAG; NO LONGER NEEDED
3909 077212 063737 075566 075560  1$:   ADD   O.BW,O.CAD ;GENERATE NEW ADDRESS
3910 077220 013737 075560 075562  O.OP2: MOV   O.CAD,O.DOT ;INITIALIZE DOT
3911 077226 004537 101246          O.OP2A: JSR   5,O.CRLF  ;<CR><LF>
3912 077232 013746 075566          MOV   O.BW,-(SP)    ;SAVE BW
3913 077236 012737 000002 075566  MOV   #2,O.BW      ;SET TO TYPE FULL WORD ADDRESS
3914 077244 013700 075560          MOV   O.CAD,R0     ;NUMBER TO TYPE
3915 077250 004537 101272          JSR   5,O.RORA     ; CHECK FORMAT
3916 077254 011637 075566          MOV   @SP,O.BW     ;RESTORE BW
3917 077260 012700 027534          MOV   #27534,R0   ;PUT '/' IN R0
3918 077264 022726 000001          CMP   #1,(SP)+    ;IS IT BYTE MODE?
3919 077270 001401          BEQ   1$          ;JUMP IF YES
3920 077272 000300          SWAB  R0          ;TYPE A /
3921 077274 004537 100730  1$:   JSR   5,O.FTYP   ;TYPE THE LOW BYTE OF R0
3922 077300 000656          BR   O.WRD1      ;GO PROCESS IT
3923
    
```

```

3926
3927           ; PROCESS ^, OPEN PREVIOUS WORD
3928
3929 077302 005737 075566  O.BACK: TST      O.BW      ; ^ RECEIVED
3930 077306 001726          BEQ      O.ERR2    ; ERROR IF NOTHING OPEN
3931 077310 004737 101214  JSR      PC,O.CLSE
3932 077314 163737 075566 075560  SUB      O.BW,O.CAD ; GENERATE NEW ADDRESS
3933 077322 000736          BR       O.OP2    ; GO DO THE REST
3934
3935           ; B HANDLER - SET AND REMOVE BREAKPOINTS
3936
3937 077324 012700 101370  O.BKPT: MOV      #O.TRTC,R0
3938 077330 006304          ASL      R4        ; MULTIPLY NUMBER BY TWO
3939 077332 005703          TST      R3
3940 077334 001423          BEQ      3$        ; IF R3 IS ZERO GO REMOVE BREAKPOINT
3941 077336 006205          ASR      R5        ; GET ONE BIT TO CARRY
3942 077340 103514          BCS      O.ERR1    ; BADNESS IF ODD ADDRESS
3943 077342 006305          ASL      R5        ; RESTORE ONE BIT
3944 077344 062704 075706  ADD      #O.ADR1,R4
3945 077350 005702          TST      R2
3946 077352 001007          BNE      1$        ; JUMP IF SPECIFIC CELL
3947 077354 020014 2$:    CMP      R0,@R4    ; IS THIS CELL FREE?
3948 077356 001405          BEQ      1$        ; JUMP IF YES
3949 077360 020427 075724  CMP      R4,#O.BKP+O.ADR1 ; ARE WE AT THE END OF OUR ROPE
3950 077364 103102          BHI      O.ERR1    ; YES, THERE IS NOTHING FREE
3951 077366 005724          TST      (R4)+    ; INCREMENT BY TWO
3952 077370 000771          BR       2$
3953 077372 020427 075724 1$:    CMP      R4,#O.BKP+O.ADR1
3954 077376 101075          BHI      O.ERR1    ; ERROR IF TOO LARGE
3955 077400 010514          MOV      R5,@R4    ; SET BREAKPOINT
3956 077402 000660          BR       O.DCDA    ; RETURN
3957 077404 005702 3$:    TST      R2
3958 077406 001410          BEQ      O.RALL    ; GO REMOVE ALL
3959 077410 020427 000016  CMP      R4,#O.BKP
3960 077414 101066          BHI      O.ERR1    ; JUMP IF NUMBER TOO LARGE
3961 077416 010064 075706  MOV      R0,O.ADR1(R4) ; CLEAR BREAKPOINT
3962 077422 005064 075730  CLR      O.CT(R4)    ; CLEAR COUNT ALSO
3963 077426 000646          BR       O.DCDA
3964 077430 005004  O.RALL: CLR      R4
3965 077432 012700 101370  MOV      #O.TRTC,R0
3966 077436 020427 000020 1$:    CMP      R4,#O.BKP+2 ; ALL DONE?
3967 077442 101240          BHI      O.DCDA    ; JUMP IF YES
3968 077444 010064 075706  MOV      R0,O.ADR1(R4) ; RESET BKPT
3969 077450 012764 000003 075752  MOV      #TRT,O.UIN(R4) ; RESET CONTENTS OF TABLE
3970 077456 005064 075730  CLR      O.CT(R4)    ; CLEAR COUNT
3971 077462 005724          TST      (R4)+    ; INCREMENT BY TWO
3972 077464 000764          BR       1$
3973
    
```

```

3976
3977          ; PROCESS 0, COMPUTE OFFSET
3978
3979 077466 022737 000002 075566 0.OFST: CMP      #2,O.BW      ;CHECK WORD MODE
3980 077474 001036          BNE      0.ERR1      ;ERROR IF NOT CORRECT MODE
3981 077476 012700 000040          MOV      #' ,R0      ;TYPE ONE BLANK
3982 077502 004537 100730          JSR      5,O.FTYP      ; AS A SEPARATOR
3983 077506 005703          TST      R3          ;WAS SEMI-COLON TYPED?
3984 077510 001430          BEQ      0.ERR1      ;NO, CALL IT AN ERROR
3985 077512 163705 075560          SUB      0.CAD,R5      ;COMPUTE
3986 077516 005305          DEC      R5
3987 077520 005305          DEC      R5          ; 16 BIT OFFSET
3988 077522 010500          MOV      R5,R0
3989 077524 004537 100634          JSR      5,O.CADV      ;NUMBER IN R0 - WORD MODE
3990 077530 010500          MOV      R5,R0
3991 077532 006200          ASR      R0          ;DIVIDE BY TWO
3992 077534 103414          BCS      1$          ;ERROR IF ODD
3993 077536 022700 177600          CMP      #-200,R0      ;COMPARE WITH -200
3994 077542 003011          BGT      1$          ;DO NOT TYPE IF OUT OF RANGE
3995 077544 022700 000177          CMP      #177,R0      ;COMPARE WITH +177
3996 077550 002406          BLT      1$          ;DO NOT TYPE IF OUT OF RANGE
3997 077552 005337 075566          DEC      0.BW          ;SET TEMPORARY BYTE MODE
3998 077556 004537 100634          JSR      5,O.CADV      ;NUMBER IN R0 - BYTE MODE
3999 077562 005237 075566          INC      0.BW          ;RESTORE WORD MODE
4000 077566 000137 076530          1$: JMP      0.DCD3      ;ALL DONE
4001
4002 077572 000137 076510          0.ERR1: JMP      0.ERR      ;INTERMEDIATE HELP
    
```

```

4005          ; PROCESS G - GO
4006
4007 077576 105737 075601      0.GO:  TSTB  0.SMFD      ;WAS ':' TYPED?
4008 077602 001773              BEQ    0.ERR1    ;BR IF NOT TYPED
4009 077604 005703              TST    R3        ;WAS K; TYPED?
4010 077606 001771              BEQ    0.ERR1    ; TYPE ?<CR,LF> IF NOT
4011 077610 112737 000021 075573  MOVB  #0.BKP+3,0.P ;CLEAR PROCEED
4012 077616 006205              ASR    R5        ;CHECK LOW ORDER BIT
4013 077620 103764              BCS   0.ERR1    ;ERROR IF ODD NUMBER
4014 077622 006305              ASL    R5        ;RESTORE WORD
4015 077624 010537 075700      0.GOGO: MOV  R5,0.UPC   ;SET UP NEW PC
4016 077630 012746 000340      MOV  #0.SIM,-(SP) ;SET HIGH PRIORITY
4017 077634 004737 101202      JSR  PC,0.WST
4018 077640 004537 100534      JSR  5,0.RSTT   ;RESTORE TELETYPE
4019 077644 105037 075572      0.TBIT: CLRB  0.T    ;CLEAR
4020 077650 052737 000020 075702  BIS  #0.TBT,0.UST ; BOTH T-BIT FLAGS
4021 077656 105737 075571      TSTB 0.S        ;SEE IF WE NEED A T BIT
4022 077662 001005              BNE   0.G02     ;IF NOT GO NOW
4023 077664 042737 000020 075702  BIC  #0.TBT,0.UST ;CLEAR THE T BIT
4024 077672 004537 100450      JSR  5,0.RSB    ;RESTORE BREAKPOINTS
4025 077676 004037 100416      0.G02: JSR  0,0.RSR  ;RESTORE REGISTERS
4026 077702 013746 075702      MOV  0.UST,-(SP) ; AND STATUS
4027 077706 013746 075700      MOV  0.UPC,-(SP) ; AND PC
4028 077712 000240              NOP
4029 077714 013746 075702      MOV  0.UST,-(SP) ; CHANGE TO HALT FOR DEBUGGING
4030 077720 042716 177420      BIC  #0.TBT!177400,(SP); MOV IN STATUS FIRST W/O T BIT
4031 077724 004737 101202      JSR  PC,0.WST   ; SO INTERRUPTS CAN HAPPEN BEFORE
4032 077730 000006              RTT           ; RTT TURNS ON THE T BIT.
    
```

```

4035      : PROCESS P - PROCEED
4036      : ONLY ALLOWED AFTER A BREAKPOINT
4037
4038 077732 105737 075601      0.PROC: TSTB      0.SMFD      ;WAS ':' TYPED?
4039 077736 001715              BEQ      0.ERR1      ;BR IF NOT TYPED
4040 077740 113700 075573      MOVB     0.P,R0
4041 077744 105700              TSTB     R0          ;CHECK LEGALITY OF PROCEED
4042 077746 002711              BLT      0.ERR1      ;NOT LEGAL
4043 077750 005702              TST      R2          ;CHECK FOR ILLEGAL COUNT
4044 077752 001307              BNE      0.ERR1      ;JUMP IF ILLEGAL
4045 077754 005703              TST      R3          ;WAS COUNT SPECIFIED?
4046 077756 001402              BEQ      0.PR1
4047 077760 010560 075730      MOV      R5,0.CT(R0) ;YES, PUT AWAY COUNT
4048 077764 012746 000340      0.PR1: MOV      #0.STM,-(SP) ;FORCE HIGH PRIORITY
4049 077770 004737 101202      JSR      PC,0.WST
4050 077774 004537 100534      JSR      5,0.RSTT    ;RESTORE TTY
4051 100000 123727 075573 000016 0.C1: CMPB     0.P,#0.BKP ;SEE IF A REAL ONE OR A FAKE
4052 100006 003316              BGT      0.TBIT      ;BRANCH IF FAKE
4053 100010 105737 075571      TSTB     0.S         ;SEE IF SINGLE INSTRUCTION MODE
4054 100014 001313              BNE      0.TBIT      ;IF SO EXIT NOW
4055 100016 012746 000340      MOV      #0.STM,-(SP) ;SET HIGH PRIORITY
4056 100022 004737 101202      JSR      PC,0.WST
4057 100026 105237 075572      INCB     0.T         ;SET T-BIT FLAG
4058 100032 052737 000020 075702  BIS      #0.TBT,0.UST ;SET T-BIT
4059 100040 000716              BR       0.G02
    
```

```

4062          : BREAKPOINT HANDLER
4063 100042 012637 075700 0.BRK: MOV (SP)+,0.UPC ;PRIORITY IS 7 UPON ENTRY
4064 100046 012637 075702      MOV (SP)+,0.UST ;SAVE STATUS AND PC
4065 100052 112737 000021 075573 MOVB #0.BKP+3,0.P ;TELL ;P THAT WE CAN CONTINUE
4066 100060 004037 100360 0.BK1: JSR 0,0.SVR ;SAVE VARIOUS REGISTERS
4067 100064 105737 075572      TSTB 0.T ;CHECK FOR T-BIT SET
4068 100070 001265      BNE 0.TBIT ;JUMP IF SET
4069 100072 004537 100602      JSR 5,0.REM ;REMOVE BREAKPOINTS
4070 100076 105737 075704      TSTB 0.PRI ;CHECK IF PRIORITY
4071 100102 100003      BPL 2$ ; IS AS SAME AS USER PGM
4072 100104 113705 075702      MOVB 0.UST,R5 ;PICK UP USER UST IF SO
4073 100110 000407      BR 3$
4074 100112 113705 075704 2$: MOVB 0.PRI,R5 ;OTHERWISE PICK UP ACTUAL PRIORITY
4075 100116 000257      CCC ;CLEAR CARRY
4076 100120 106005      RORB R5 ;SHIFT LOW ORDER BITS
4077 100122 106005      RORB R5 ; INTO
4078 100124 106005      RORB R5 ; HIGH ORDER
4079 100126 106005      RORB R5 ; POSITION
4080 100130 042705 177420 3$: BIC #0.TBT!177400,R5 ;CLEAR POSSIBLE T BIT (S/I MODE)
4081 100134 010546      MOV R5,-(SP) ;PUT THE STATUS AWAY WHERE IT BELONGS
4082 100136 004737 101202      JSR PC,0.WST
4083 100142 013705 075700      MOV 0.UPC,R5 ;GET PC, IT POINTS TO THE TRT
4084 100146 105737 075571      TSTB 0.S ;SEE IF IT WAS SINGLE INSTRUCTION FUN
4085 100152 100432      BMI 14$ ;IF SO HANDLE THERE
4086 100154 005745      TST -(R5)
4087 100156 010537 075700      MOV R5,0.UPC
4088 100162 012704 000016      MOV #0.BKP,R4 ;GET A COUNTER
4089 100166 020564 075706 11$: CMP R5,0.ADR1(R4) ;COMPARE WITH LIST
4090 100172 001426      BEQ 12$ ;JUMP IF FOUND
4091 100174 005304      DEC R4
4092 100176 005304      DEC R4
4093 100200 002372      BGE 11$ ;RE-LOOP UNTIL FOUND
4094 100202 004537 100500      JSR 5,0.SVTT ;SAVE TELETYPE STATUS
4095 100206 004537 101246      JSR 5,0.CRLF
4096 100212 012704 101366      MOV #0.BD,R4 ;ERROR, NOTHING FOUND
4097 100216 012703 101367      MOV #0.BD+1,R3
4098 100222 004537 101126      JSR 5,0.TYPE ;OUTPUT 'BE' FOR BAD ENTRY
4099 100226 010500      MOV R5,R0
4100 100230 062737 000002 075700 ADD #2,0.UPC ;POP OVER THE ADJUSTMENT ABOVE
4101 100236 000444      BR 13$ ; OR CONTINUE
    
```

```

4104 100240 112704 000020      14$:  MOVB  #0,BKP+2,R4      ;SET BREAK POINT HIGH + 1
4105 100244 010564 075706      MOV  R5,O.ADR1(R4)      ;STORE NEXT PC VALUE FOR TYPE OUT
4106 100250 110437 075573      12$:  MOVB  R4,O.P          ;ALLOW PROCEED
4107 100254 005364 075730      DEC  O.CT(R4)
4108 100260 003247              BGT  O.C1              ;JUMP IF REPEAT
4109 100262 012764 000001 075730  MOV  #1,O.CT(R4)      ;RESET COUNT TO 1
4110 100270 004537 100500      JSR  5,O.SVTT         ;SAVE TELETYPE STATUS, R4 IS SAFE
4111 100274 012700 000102      MOV  #'B,R0
4112 100300 004537 100730      JSR  5,O.FTYP         ;TYPE 'B'
4113 100304 113700 075573      MOVB O.P,R0          ;CONVERT BREAKPOINT NUMBER TO ASCII
4114 100310 062700 000140      ADD  #140,R0
4115 100314 006200              ASR  R0
4116 100316 004537 100730      JSR  5,O.FTYP
4117 100322 012700 000073      MOV  #':,R0
4118 100326 004537 100730      JSR  5,O.FTYP         ; TYPE
4119 100332 012737 000002 075566  MOV  #2,O.BW         ; SET WORD MODE
4120 100340 113704 075573      MOVB O.P,R4
4121 100344 016400 075706      MOV  O.ADR1(R4),R0   ;GET ADDRESS OF BREAK
4122 100350 004537 101272      13$:  JSR  5,O.RORA       ; CHECK FORMAT
4123 100354 000137 076520      JMP  O.DCD           ;GO TO DECODER
    
```

```

4126          ; SAVE REGISTERS R0-R6
4127          ;   INTERNAL STACK
4128
4129 100360 012637 075564  O.SVR: MOV   (SP)+,O.XXX ;PICK REGISTER FROM STACK AND SAVE
4130 100364 010637 075676      MOV   SP,O.USP   ;SAVE USER STACK ADDRESS
4131 100370 012706 075676      MOV   #O.USP,SP  ;SET TO INTERNAL STACK
4132 100374 010546           MOV   R5,-(SP)   ;SAVE
4133 100376 010446           MOV   R4,-(SP)   ; REGISTERS
4134 100400 010346           MOV   R3,-(SP)   ; 1
4135 100402 010246           MOV   R2,-(SP)   ; THRU
4136 100404 010146           MOV   R1,-(SP)   ; 5
4137 100406 013746 075564      MOV   O.XXX,-(SP) ;PUT SAVED REGISTER ON STACK
4138 100412 005746           TST   -(SP)
4139 100414 000200           RTS   R0
4140
4141          ; RESTORE REGISTERS R0-R6
4142
4143 100416 005726           O.RSR: TST   (SP)+ ;POP THE EXTRA CELL
4144 100420 012637 075564      MOV   (SP)+,O.XXX ;GET R0 FROM STACK
4145 100424 012601           MOV   (SP)+,R1   ;RESTORE
4146 100426 012602           MOV   (SP)+,R2   ; REGISTERS
4147 100430 012603           MOV   (SP)+,R3   ; 1
4148 100432 012604           MOV   (SP)+,R4   ; THRU
4149 100434 012605           MOV   (SP)+,R5   ; 5
4150 100436 013706 075676      MOV   O.USP,SP   ;RESTORE USER STACK
4151 100442 013746 075564      MOV   O.XXX,-(SP) ;PUT R0 ON USER STACK
4152 100446 000200           RTS   R0
    
```

```

4155 ; RESTORE BREAKPOINTS 0-7
4156
4157 100450 012704 000016 0.RSB: MOV #0.BKP,R4 ;RESTORE ALL BREAKPOINTS
4158 100454 017464 075706 075752 1$: MOV @0.ADR1(R4),0.UIN(R4);SAVE CONTENTS
4159 100462 013774 101370 075706 MOV 0.TRTC,@0.ADR1(R4);REPLACE WITH TRAP
4160 100470 005304 DEC R4
4161 100472 005304 DEC R4
4162 100474 002367 BGE 1$ ;RE-LOOP UNTIL DONE
4163 100476 000205 RTS R5 ; THEN QUIT
4164
4165 ; SAVE TELETYPE STATUS
4166
4167 100500 113737 177560 075574 0.SVTT: MOVB 0.RCSR,0.CSR1 ;SAVE R C/SR
4168 100506 113737 177564 075575 MOVB 0.TCSR,0.CSR2 ;SAVE T C/SR
4169 100514 105037 177560 CLRB 0.RCSR ;CLEAR ENABLE AND MAINTENANCE
4170 100520 105037 177564 CLRB 0.TCSR ; BITS IN BOTH C/SR
4171 100524 105737 177564 1$: TSTB 0.TCSR ;LOOP UNTIL READY BIT COMES ON
4172 100530 100375 BPL 1$ ;BR IF BIT NOT ON
4173 100532 000205 RTS R5
4174
4175 ; RESTORE TELETYPE STATUS
4176 100534 004537 101246 0.RSTT: JSR 5.0.CRLF
4177 100540 105737 177564 TSTB 0.TCSR ;WAIT READY
4178 100544 100375 BPL -4 ; ON PRINTER
4179 100546 032737 004000 177560 BIT #4000,0.RCSR ;CHECK BUSY FLAG
4180 100554 001403 BEQ 1$ ;SKIP READY LOOP IF NOT BUSY
4181 100556 105737 177560 TSTB 0.RCSR ;WAIT READY
4182 100562 100375 BPL -4 ; ON READER
4183 100564 113737 075574 177560 1$: MOVB 0.CSR1,0.RCSR ;RESTORE
4184 100572 113737 075575 177564 MOVB 0.CSR2,0.TCSR ; THE STATUS REGISTERS
4185 100600 000205 RTS R5
4186
4187 ; REMOVE BREAKPOINTS 0-7
4188 ; IN THE OPPOSITE ORDER OF SETTING
4189
4190 100602 105737 075571 0.REM: TSTB 0.S ;SEE IF SINGLE INSTRUCTION IS GOING
4191 100606 001011 BNE 2$ ;EXIT IF SO
4192 100610 005004 CLR R4 ;REMOVE ALL BREAKPOINTS
4193 100612 016474 075752 075706 1$: MOV 0.UIN(R4),@0.ADR1(R4) ;CLEAR BREAKPOINT
4194 100620 005204 INC R4
4195 100622 005204 INC R4
4196 100624 020427 000016 CMP R4,#0.BKP
4197 100630 003770 BLE 1$ ;RE-LOOP UNTIL DONE
4198 100632 000205 2$: RTS R5 ;THEN QUIT
    
```

ODT (CEMKA) DATA AREA

```

4201 ; TYPE OUT CONTENTS OF WORD OR BYTE WITH ONE TRAILING SPACE
4202 ; WORD IS IN R0
4203
4204 100634 012703 000006 O.CADV: MOV #6,R3 ;# OF DIGITS
4205 100640 012704 177776 MOV #-2,R4 ;# OF BITS FIRST-3
4206 100644 022737 000001 075566 CMP #1,O.BW ;SEE IF WORD MODE
4207 100652 001004 BNE 3$ ;BRANCH IF SO
4208 100654 162703 000003 SUB #3,R3 ;ONLY DO 3 DIGITS
4209 100660 005204 INC R4 ;DO 2 BITS FIRST
4210 100662 000300 SWAB R0 ;AND TURN R0 AROUND
4211 100664 010046 3$: MOV R0,-(SP) ;SAVE R0
4212 100666 062704 000003 2$: ADD #3,R4 ;COMPUTE THE NUMBER OF BITS TO DO
4213 100672 005000 CLR R0
4214 100674 006116 1$: ROL (SP) ;GET A BIT
4215 100676 006100 ROL R0 ;STORE IT AWAY
4216 100700 005304 DEC R4 ;DECREMENT COUNTER
4217 100702 003374 BGT 1$ ;LOOP IF MORE BITS NEEDED
4218 100704 062700 000060 ADD #'0,R0 ;CONVERT TO ASCII
4219 100710 004537 100730 JSR R5,O.FTYP ;TYPE IT
4220 100714 005303 DEC R3 ;SEE IF MORE DIGITS TO DO
4221 100716 003363 BGT 2$ ;LOOP IF SO
4222 100720 112700 000040 MOVB #' ,R0 ;SET UP FOR TRAILING SPACE
4223 100724 005726 TST (SP)+ ;GET RID OF JUNK
4224 100726 000400 BR O.FTYP
4225
4226
4227 ; TYPE ONLY ONE CHARACTER (CONTAINED IN R0)
4228
4229 100730 105737 177564 O.FTYP: TSTB O.TCSR
4230 100734 100375 BPL O.FTYP
4231 100736 042700 177400 BIC #177400,R0 ;CLEAR HIGH BYTE,SHOULD NOT
4232 ;CONTAIN INPORTANT INFO.
4233 100742 001416 BEQ 3$
4234 100744 022700 000176 CMP #176,R0 ;PRINT ? FOR 177:16-37
4235 100750 103411 BLO 2$ ; 1-10 AND 200-377.
4236 100752 022700 000037 CMP #37,R0
4237 100756 103410 BLO 3$
4238 100760 022700 000015 CMP #15,R0
4239 100764 103403 BLO 2$
4240 100766 022700 000010 CMP #10,R0
4241 100772 103402 BLO 3$
4242 100774 012700 000077 2$: MOV #'?,R0
4243 101000 110037 177566 3$: MOVB R0,O.TDB
4244 101004 000205 O.TYP1: RTS R5

```

```

4247      ; GENERAL CHARACTER INPUT ROUTINE -- ODT11X
4248      ; CHARACTER INPUT GOES TO R0
4249
4250 101006 105737 177560      0.GET: TSTB  O.RCSR      ;WAIT FOR
4251 101012 100375              BPL  O.GET          ; INPUT FROM KBD
4252 101014 113700 177562      MOVB  O.RDB,R0      ;GET CHARACTER - STRIP OFF PARITY
4253 101020 042700 177600      BIC   #177600,R0    ;STRIP OFF PARITY FROM CHARACTER
4254 101024 122700 000003      CMPB  #3,R0         ;IS IT ^C?
4255 101030 001435              BEQ   3$            ;IF SO, DO NOT ECHO
4256 101032 122700 000006      CMPB  #6,R0         ;IS IT ^F?
4257 101036 001432              BEQ   3$            ;IF SO, DO NOT ECHO
4258 101040 122700 000005      CMPB  #5,R0         ;IS IT ^E?
4259 101044 001427              BEQ   3$            ;IF SO, DO NOT ECHO
4260 101046 105737 075602      TSTB  O.SCRN        ;SHOULD WE ECHO <LF>?
4261 101052 001003              BNE   2$            ;BR IF YES
4262 101054 120027 000012      CMPB  R0,#012       ;SEE IF A <LF>
4263 101060 001421              BEQ   3$            ;IF SO SAVE THE PAPER
4264 101062 120027 000173      2$:  CMPB  R0,#173
4265 101066 002005              BGE   1$
4266 101070 122700 000140      CMPB  #140,R0       ;TEST FOR LOWER CASE
4267 101074 002002              BGE   1$
4268 101076 162700 000040      SUB   #40,R0        ;CHAR IS LC-CONVERT TO UPPER CASE
4269 101102 004537 100730      1$:  JSR   5,O.FTYP   ;ECHO CHARACTER
4270 101106 001737              BEQ   0.GET         ;IGNORE NULLS
4271 101110 105737 075602      TSTB  O.SCRN        ;SHOULD WE PASS ON SPACES?
4272 101114 001003              BNE   3$            ;BR IF YES
4273 101116 122700 000040      CMPB  #40,R0        ;CHECK FOR SPACES
4274 101122 001731              BEQ   0.GET         ;IGNORE SPACES
4275 101124 000205              3$:  RTS   R5
    
```

```

4278      ; GENERAL CHARACTER OUTPUT ROUTINE - ODT11X
4279      ; ADDRESS OF FIRST BYTE IN R4,
4280      ; ADDRESS OF LAST BYTE IN R3, (R3)>(R4)
4281      ; EXPECTS LOCS 56,57 TO BE INITIALIZED BY MONITOR FOR FILL
4282      ; CHARACTERISTICS OF TERMINAL
4283      ; 56=CHAR TO BE FILLED AFTER
4284      ; 57=# OF NULLS TO FILL WITH
4285
4286 101126 020304      O.TYPE: CMP      R3,R4      ;CHECK FOR COMPLETION
4287 101130 103725      BLO      O.TYP1      ; EXIT WHEN DONE
4288 101132 112400      MOVB     (R4)+,R0      ;GET A CHARACTER
4289 101134 004537 100730 JSR      5,O.FTYP      ;TYPE ONE CHARACTER
4290 101140 120037 003002 CMPB     R0,@#$FILLC    ;COMPARE CHAR AGAINST FILL REQUIREMENT
4291 101144 001370      BNE      O.TYPE      ;NO FILL NEEDED
4292 101146 113737 002451 075576 MOVB     @#$FILLS,O.FIL ;FILL COUNT INTO TEMP
4293 101154 005000      CLR      R0          ;FILL WITH NULLS
4294 101156 004537 100730 2$: JSR      R5,O.FTYP      ;TYPE NULLS
4295 101162 105337 075576 DECB     O.FIL        ;DECREASE COUNT
4296 101166 003373      BGT      2$          ;BRANCH IF NOT DONE
4297 101170 000756      BR       O.TYPE      ;LOOP UNTIL DONE
    
```

```

4300          :SUBROUTINE TO READ THE PS INDEPENDENT OF MACHINE
4301
4302 101172 013737 177776 075702 0.RRST: MOV     ST,O.UST      :STORE THE STATUS IN USER
4303                                     :STATUS AREA.FOR AN LSI
4304                                     :THIS IS CHANGED TO
4305                                     :MFPS O.UST
4306 101200 000207          RTS     PC
4307
4308          :SUBROUTINE TO WRITE PS INDEPENDENT OF MACHINE
4309          :CALL ROUTINE WITH PS VALUE
4310          :ON THE STACK
4311
4312 101202 016637 000002 177776 0.WST:  MOV     2(SP),ST      :STORE NEW PS VALUE
4313                                     :THIS INSTRUCTION IS CHANGED FOR LSI
4314                                     :TO MTPS 2(SP)
4315 101210 012616          MOV     (SP)+,(SP)      :PUT RETURN PC OVER SUB. ARGUMENT
4316 101212 000207          RTS     PC              :TO RETURN WITHOUT IT ON THE STACK
4317          : CLOSE WORD OR BYTE AND EXIT,
4318          : UPON ENTERING, R2 HAS NUMERIC FLAG, R4 HAS CONTENTS
4319
4320          .ENABL  LSB
4321 101214 005702 0.CLSE:  TST     R2              :IF NO NUMBER WAS TYPED THERE IS
4322 101216 001412          BEQ     1$              :NO CHANGE TO THE OPEN CELL
4323 101220 022737 000001 075566  CMP     #1,O.BW
4324 101226 001404          BEQ     2$              :JUMP IF BYTE MODE
4325 101230 101005          BHI     1$              :JUMP IF ALREADY CLOSED
4326 101232 010477 174322          MOV     R4,@O.CAD      :STORE WORD
4327 101236 000402          BR      1$
4328 101240 110477 174314 2$:  MOVB   R4,@O.CAD      :STORE BYTE
4329 101244 000207 1$:  RTS     PC
4330 101246 012703 101317 0.CRLF: MOV     #O.CR+1,R3      :LWA <CR,LF>
4331 101252 000402          BR      3$
4332 101254 012703 101320 0.CRLS: MOV     #O.CR+2,R3      :LWA <CR,LF>*
4333 101260 012704 101316 3$:  MOV     #O.CR,R4        :FWA
4334 101264 004537 101126          JSR     5,O.TYPE      :TYPE SOMETHING
4335 101270 000205          RTS     R5
4336          .DSABL  LSB
    
```

```

4339      ;SUBROUTINE O.RORA
4340      ;FUNCTION:
4341      ;ADDRESS WILL BE PRINTED IN ABSOLUTE FORM
4342      ;INPUT: THE ADDRESS TO BE PRINTED IS IN RO.
4343      ;DATA SAVED: RO CONTAINING THE ADDRESS TO BE
4344      ;PRINTED, AND LOCATION O.CAD CONTAINING
4345      ;THE CURRENT ADDRESS WERE SAVED AND RESTORED.
4346      ;CALLED: JSR 5,O.RORA
4347
4348 101272 004537 100634 O.RORA: JSR 5,O.CADV ;PRINT ABSOLUTE ADDRESS
4349 101276 000205 RTS R5
4350 101300 012 O.ID: .BYTE 012
4351 101301 015 .BYTE 015
4352 101302 040 .BYTE 40
4353 101303 117 104 124 .ASCII /ODT (CEMKA)/
      101306 040 050 103
      101311 105 115 113
      101314 101 051
4354 101315 O.IDND=-1
4355 101316 015 O.CR: .BYTE 015 ; <CR>
4356 101317 012 .BYTE 012 ; <LF>
4357 101320 052 .BYTE '*' ; *
4358
4359 101321 073 O.LGCH: .BYTE '/' ; /
4360 101322 057 .BYTE '\ ' ; \ (BACK SLASH)
4361 101323 134 .BYTE 015 ; CARRIAGE RETURN
4362 101324 015 .BYTE '$' ; $
4363 101325 044 .BYTE 'G' ; G
4364 101326 107 .BYTE 012 ; <LF>
4365 101327 012 .BYTE '<' ; (BACK ARROW)
4366 101330 137 .BYTE '^' ; (UP ARROW)
4367 101331 074 .BYTE 'O' ; O
4368 101332 136 .BYTE 'B' ; B
4369 101333 117 .BYTE 'P' ; P
4370 101334 102 .BYTE '@' ; @
4371 101335 120 .BYTE '>' ; >
4372 101336 100 .BYTE '-' ; -
4373 101337 076 .BYTE 003 ; CTRL C
4374 101340 123 .BYTE 006 ; CTRL F
4375 101341 055 .BYTE 005 ; CTRL E
4376 101342 003
4377 101343 006
4378 101344 005
4379 000024 O.CLGT = .-O.LGCH ;TABLE LENGTH
4380
4381 101345 123 O.TL: .BYTE 'S' ;DO 1
4382 101346 120 .BYTE 'P' ;NOT 2
4383 101347 115 .BYTE 'M' ;CHANGE 3
4384 101350 000 .BYTE 0 ;THE 4
4385 101351 000 .BYTE 0 ;ORDER 5
4386 101352 103 .BYTE 'C' ; 6
4387 101353 106 .BYTE 'F' ; 7
4388 101354 122 .BYTE 'R' ; 10
4389 101355 000 .BYTE 0 ; 11
4390 101356 000 .BYTE 0 ; 12
4391 101357 000 .BYTE 0 ; 13
4392 101360 000 .BYTE 0 ; 14
    
```

4393 101361 000
4394 101362 000
4395 101363 000
4396 101364 102
4397 000020
4398
4399 101366 042502
4400 101370 000003

.BYTE 0 : 15
.BYTE 0 : 16
.BYTE 0 : 17
.BYTE 'B : 20
O.LG = -.O.TL
O.BD: .EVEN
O.TRTC: TRT .WORD 'BE

;TRACE TRAP PROTOTYPE

4403
 4404
 4405
 4406
 4407
 4408
 4409
 4410
 4411
 4412
 4413
 4414
 4415
 4416
 4417 101372
 4418 101372 104530
 4419 101374 110307
 4420 101376 102504
 4421 101400 103103
 4422
 4423 101402 103220
 4424 101404 107366
 4425 101406 102252
 4426 101410 102743
 4427
 4428 101412 103256
 4429 101414 107446
 4430 101416 102302
 4431 101420 103071
 4432
 4433 101422 103311
 4434 101424 107446
 4435 101426 102312
 4436 101430 103071
 4437
 4438 101432 103357
 4439 101434 107507
 4440 101436 102322
 4441 101440 102743
 4442
 4443 101442 103434
 4444 101444 107507
 4445 101446 102322
 4446 101450 102743
 4447
 4448 101452 103461
 4449 101454 107507
 4450 101456 102322
 4451 101460 102743
 4452
 4453 101462 103633
 4454 101464 107366
 4455 101466 102252
 4456 101470 102743

.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

\$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
 EM10
 DH1
 DT1
 DF2

4459			:ERROR	11
4460	101472	103677	EM11	
4461	101474	107664	DH7	
4462	101476	102362	DT7	
4463	101500	102767	DF3	
4464			:ERROR	12
4465	101502	103677	EM11	
4466	101504	107664	DH7	
4467	101506	102362	DT7	
4468	101510	103004	DF4	
4469			:ERROR	13
4470	101512	103721	EM12	
4471	101514	110145	DH10	
4472	101516	102444	DT10	
4473	101520	102743	DF2	
4474			:ERROR	14
4475	101522	103677	EM11	
4476	101524	107664	DH7	
4477	101526	102362	DT7	
4478	101530	103021	DF5	
4479			:ERROR	15
4480	101532	103677	EM11	
4481	101534	107664	DH7	
4482	101536	102362	DT7	
4483	101540	103036	DF6	
4484			:ERROR	16
4485	101542	103745	EM13	
4486	101544	110307	DH13	
4487	101546	102504	DT13	
4488	101550	103103	DF11	
4489			:ERROR	17
4490	101552	103777	EM14	
4491	101554	110307	DH13	
4492	101556	102504	DT13	
4493	101560	103103	DF11	
4494			:ERROR	20
4495	101562	104043	EM15	
4496	101564	110307	DH13	
4497	101566	102504	DT13	
4498	101570	103103	DF11	
4499			:ERROR	21
4500	101572	104111	EM16	
4501	101574	110243	DH11	
4502	101576	102466	DT11	
4503	101600	102743	DF2	
4504			:ERROR	22
4505	101602	104137	EM17	
4506	101604	107664	DH7	
4507	101606	102362	DT7	
4508	101610	103021	DF5	
4509			:ERROR	23
4510	101612	104177	EM18	
4511	101614	110006	DH8	
4512	101616	102416	DT8	
4513	101620	103076	DF10	

4516			:ERROR	24	
4517	101622	104256	EM19		
4518	101624	110307	DH13		
4519	101626	102504	DT13		
4520	101630	103103	DF11		
4521			:ERROR	25	
4522	101632	104333	EM20		
4523	101634	110307	DH13		
4524	101636	102504	DT13		
4525	101640	103103	DF11		
4526			:ERROR	26	
4527	101642	000000	0		:NO MESSAGE
4528	101644	110272	DH12		
4529	101646	102476	DT12		
4530	101650	102743	DF2		
4531			:ERROR	27	
4532	101652	104415	EM21		
4533	101654	110243	DH11		
4534	101656	102466	DT11		
4535	101660	102743	DF2		
4536			:ERROR	30	
4537	101662	104454	EM22		
4538	101664	110307	DH13		
4539	101666	102504	DT13		
4540	101670	103103	DF11		
4541			:ERROR	31	
4542	101672	000000	0		:NO MESSAGE
4543	101674	110415	DH14		
4544	101676	102530	DT14		
4545	101700	102743	DF2		
4546			:ERROR	32	
4547	101702	104501	EM23		
4548	101704	107507	DH5		
4549	101706	102322	DT5		
4550	101710	102743	DF2		
4551			:ERROR	33	
4552	101712	106025	EM38		
4553	101714	110672	DH17		
4554	101716	102264	DT2		
4555	101720	103116	DF13		
4556			:ERROR	34	
4557	101722	104607	EM25		
4558	101724	110533	DH15		
4559	101726	102556	DT16		
4560	101730	103053	DF7		
4561			:ERROR	35	
4562	101732	104634	EM26		
4563	101734	110655	DH16		
4564	101736	102610	DT17		
4565	101740	102767	DF3		
4566			:ERROR	36	
4567	101742	103521	EM8		
4568	101744	110672	DH17		
4569	101746	102616	DT18		
4570	101750	103116	DF13		

TABLE	ERROR POINTER		
4573			:ERROR 37
4574	101752	103556	EM9
4575	101754	110106	DH9
4576	101756	102432	DT9
4577	101760	102743	DF2
4578			:ERROR 40
4579	101762	104705	EM27
4580	101764	110655	DH16
4581	101766	102610	DT17
4582	101770	103067	DF8
4583			:ERROR 41
4584	101772	104774	EM28
4585	101774	110672	DH17
4586	101776	102634	DT19
4587	102000	103116	DF13
4588			:ERROR 42
4589	102002	105624	EM35
4590	102004	107664	DH7
4591	102006	102362	DT7
4592	102010	102767	DF3
4593			:ERROR 43
4594	102012	105027	EM29
4595	102014	107664	DH7
4596	102016	102362	DT7
4597	102020	102767	DF3
4598			:ERROR 44
4599	102022	105111	EM30
4600	102024	107664	DH7
4601	102026	102362	DT7
4602	102030	103021	DF5
4603			:ERROR 45
4604	102032	105230	EM31
4605	102034	107664	DH7
4606	102036	102362	DT7
4607	102040	102767	DF3
4608			:ERROR 46
4609	102042	105330	EM32
4610	102044	107664	DH7
4611	102046	102362	DT7
4612	102050	102767	DF3
4613			:ERROR 47
4614	102052	105435	EM33
4615	102054	107664	DH7
4616	102056	102362	DT7
4617	102060	102767	DF3
4618			:ERROR 50
4619	102062	105543	EM34
4620	102064	111023	DH19
4621	102066	102700	DT22
4622	102070	103114	DF12
4623			:ERROR 51
4624	102072	105711	EM36
4625	102074	107615	DH6
4626	102076	102346	DT6
4627	102100	102743	DF2

4630			:ERROR	52
4631	102102	105760	EM37	
4632	102104	110672	DH17	
4633	102106	102616	DT18	
4634	102110	103116	DF13	
4635			:ERROR	53
4636	102112	106133	EM39	
4637	102114	110672	DH17	
4638	102116	102616	DT18	
4639	102120	103116	DF13	
4640			:ERROR	54
4641	102122	106220	EM40	
4642	102124	107423	DH2	
4643	102126	102652	DT20	
4644	102130	102743	DF2	
4645			:ERROR	55
4646	102132	106272	EM41	
4647	102134	110746	DH18	
4648	102136	102662	DT21	
4649	102140	103103	DF11	
4650			:ERROR	56
4651	102142	106374	EM42	
4652	102144	110746	DH18	
4653	102146	102662	DT21	
4654	102150	103103	DF11	
4655			:ERROR	57
4656	102152	103134	EM1	
4657	102154	000000	0	
4658	102156	000000	0	
4659	102160	000000	0	
4660			:ERROR	60
4661	102162	106477	EM43	
4662	102164	111037	DH20	
4663	102166	102706	DT23	
4664	102170	102743	DF2	
4665			:ERROR	61
4666	102172	106575	EM44	
4667	102174	111056	DH21	
4668	102176	102714	DT24	
4669	102200	102743	DF2	
4670			:ERROR	62
4671	102202	106672	EM45	
4672	102204	111037	DH20	
4673	102206	102706	DT23	
4674	102210	102743	DF2	
4675			:ERROR	63
4676	102212	106773	EM46	
4677	102214	110672	DH17	
4678	102216	102264	DT2	
4679	102220	103116	DF13	
4680			:ERROR	64
4681	102222	107062	EM47	
4682	102224	110655	DH16	
4683	102226	102610	DT17	
4684	102230	102767	DF3	

4687			:ERROR 65
4688	102232	107170	EM48
4689	102234	111063	DH22
4690	102236	102652	DT20
4691	102240	102743	DF2
4692			:ERROR 66
4693	102242	107267	EM49
4694	102244	111111	DH23
4695	102246	102720	DT25
4696	102250	103125	DF14

ERROR DATA TAGS (DT)

4699						.SBTTL	ERROR DATA TAGS (DT)
4700	102252	002024	002040	002050	DT1:	.WORD	ERRPC,ADDRESS,GOOD,BAD,0
	102260	002056	000000				
4701	102264	002024	002304	002656	DT2:	.WORD	ERRPC,DUMMY,SLFLAG,SLFLAG+2,SLFLAG+4,SLFLAG+6,0
	102272	002660	002662	002664			
	102300	000000					
4702	102302	002024	002042	002074	DT3:	.WORD	ERRPC,PADDRESS,PARCNT,0
	102310	000000					
4703	102312	002024	002040	002072	DT4:	.WORD	ERRPC,ADDRESS,NEMCNT,0
	102320	000000					
4704	102322	002024	177766	177744	DT5:	.WORD	ERRPC,CPUERR,MEMERR,HIADRS,LOADRS,MMR0,MMR1,MMR2,MMR3,0
	102330	177742	177740	177572			
	102336	177574	177576	172516			
	102344	000000					
4705	102346	002024	002522	002466	DT6:	.WORD	ERRPC,APTCORE,MJSIZE,APTMOS,MKSIZE,0
	102354	002524	002470	000000			
4706	102362	002024	002304	002040	DT7:	.WORD	ERRPC,DUMMY,ADDRESS,DUMMY,GOOD,BAD,BADXOR
	102370	002304	002050	002056			
	102376	002064					
4707	102400	002304	002304	002304		.WORD	DUMMY,DUMMY,DUMMY,DUMMY,DUMMY,DUMMY,0
	102406	002304	002304	002304			
	102414	000000					
4708	102416	002040	002050	002056	DT8:	.WORD	ADDRESS,GOOD,BAD,GDSWITCH,BDSWITCH,0
	102424	002476	002500	000000			
4709	102432	002024	177766	177744	DT9:	.WORD	ERRPC,CPUERR,MEMERR,CSRNO,0
	102440	002256	000000				
4710	102444	002306	002310	002312	DT10:	.WORD	DETRO,DETR1,DETR2,DETR3,DETR4,DETR5,DETSP,DETSPW,0
	102452	002314	002316	002320			
	102460	002322	002324	000000			
4711	102466	002024	002144	002146	DT11:	.WORD	ERRPC,CSR,CSR+2,0
	102474	000000					
4712	102476	002144	002146	000000	DT12:	.WORD	CSR,CSR+2,0
4713	102504	002024	002304	002040	DT13:	.WORD	ERRPC,DUMMY,ADDRESS,DUMMY,TSTDAT,TSTDAT+2,CHECK,CSR,CSR+2,0
	102512	002304	002344	002346			
	102520	002404	002144	002146			
	102526	000000					
4714	102530	177744	177750	177746	DT14:	.WORD	MEMERR,MAINT,CONTRL,HIADRS,LOADRS,CPUERR,MMR0,MMR1,MMR2,MMR3,0
	102536	177742	177740	177766			
	102544	177572	177574	177576			
	102552	172516	000000				
4715	102556	002024	002304	002304	DT16:	.WORD	ERRPC,DUMMY,DUMMY,GOOD,GOOD2,GOOD3
	102564	002050	002052	002054			
4716	102572	002056	002060	002062		.WORD	BAD,BAD2,BAD3,DUMMY,DUMMY,DUMMY,0
	102600	002304	002304	002304			
	102606	000000					
4717	102610	002024	002304	000000	DT17:	.WORD	ERRPC,DUMMY,0
4718	102616	002024	002304	002646	DT18:	.WORD	ERRPC,DUMMY,PORTCOUNT,PORTCOUNT+2,PORTCOUNT+4,PORTCOUNT+6,0
	102624	002650	002652	002654			
	102632	000000					
4719	102634	002024	002304	002606	DT19:	.WORD	ERRPC,DUMMY,PORTDIR,PORTDIR+2,PORTDIR+4,PORTDIR+6,0
	102642	002610	002612	002614			
	102650	000000					
4720	102652	002024	002050	002056	DT20:	.WORD	ERRPC,GOOD,BAD,0
	102660	000000					
4721	102662	002024	002304	002040	DT21:	.WORD	ERRPC,DUMMY,ADDRESS,DUMMY,GOOD,BAD,0
	102670	002304	002050	002056			
	102676	000000					

ERROR DATA TAGS (DT)

4722	102700	002024	002304	000000	DT22:	.WORD	ERRPC,DUMMY,0
4723	102706	002024	002056	000000	DT23:	.WORD	ERRPC,BAD,0
4724	102714	002024	000000		DT24:	.WORD	ERRPC,0
4725	102720	002024	002304	002646	DT25:	.WORD	ERRPC,DUMMY,PORTCOUNT,PORTCOUNT+2,PORTCOUNT+4,PORTCOUNT+6,BAD,0
	102726	002650	002652	002654			
	102734	002056	000000				

Address	Hex	Hex	Hex	Hex	Code	Description
4746					.SBTTL	ERROR MESSAGES (EM)
4752	103134	125	116	105	EM1:	.ASCIZ /UNEXPECTED TRAP TO 170 (IIST) - PROGRAM CONTINUING!/ /CAN'T SET 22 BIT MODE IN MMR3/
4753	103220	103	101	116	EM2:	.ASCIZ /PARITY ERROR(S) IN BANK 0/
4754	103256	120	101	122	EM3:	.ASCIZ /NON-EXISTANT MEMORY (HOLES) IN BANK 0/
4755	103311	116	117	116	EM4:	.ASCIZ /ILLEGAL OR RESERVED INSTRUCTION (TRAP TO 10)/
4756	103357	111	114	114	EM5:	.ASCIZ /UNEXPECTED TRAP TO 4/
4757	103434	125	116	105	EM6:	.ASCIZ /MEMORY MANAGEMENT (TRAP TO 250)/
4758	103461	115	105	115	EM7:	.ASCIZ /PORT CONTENTION NOT BALANCED/
4759	103521	120	117	122	EM8:	.ASCIZ /NON-EXISTANT CSR CAUSED INCORRECT ERROR TRAP/
4760						
4761	103556	116	117	116	EM9:	.ASCIZ /INCORRECT DATA FROM DEVICE REGISTER/
4762	103633	111	116	103	EM10:	.ASCIZ /MEMORY DATA ERROR/
4763	103677	115	105	115	EM11:	.ASCIZ /DETAILED ERROR DUMP/
4764	103721	104	105	124	EM12:	.ASCIZ /MISSING EXPECTED SBE FLAG/
4765	103745	115	111	123	EM13:	.ASCIZ /WRITE BYTE FAILED TO CLEAR SBE FLAG/
4766	103777	127	122	111	EM14:	.ASCIZ /FAILED TO GET INTERRUPT WITH DBE FLAG/
4767	104043	106	101	111	EM15:	.ASCIZ /ONLY ONE DBE FLAG SET/
4768	104111	117	116	114	EM16:	.ASCIZ /MEMORY DATA ERROR IN CHECK BITS/
4769	104137	115	105	115	EM17:	.ASCIZ /SYSTEM SIZE REGISTER LOW DOES NOT MATCH MEMORY/
4770	104177	123	131	123	EM18:	.ASCIZ /SBE OR DBE CAUSED PARITY TRAP WHEN INHIBITED/
4771	104256	123	102	105	EM19:	.ASCIZ /SBE OR DBE DID NOT CAUSE PARITY TRAP WHEN ENABLED/
4772	104333	123	102	105	EM20:	.ASCIZ /SBE OR DBE ON MASTER TEST WORD/
4773	104415	123	102	105	EM21:	.ASCIZ /MISSING EXPECTED DBE/
4774	104454	115	111	123	EM22:	.ASCIZ /UNEXPECTED PARITY TRAP/
4775	104501	125	116	105	EM23:	.ASCIZ /RECEIVED DBE FLAG WHEN EXPECTING ONLY SBE FLAG/
4776	104530	122	105	103	EM24:	.ASCIZ /CHECK BIT DATA ERROR/
4777	104607	103	110	105	EM25:	.ASCIZ /ADDRESS PARITY ERROR DID NOT CAUSE ABORT/
4778	104634	101	104	104	EM26:	.ASCIZ /MK11 PROTECTION POINTER FAILURE - DID NOT PROTECT BANK/
4779	104705	115	113	061	EM27:	.ASCIZ /CPU IDENTIFICATION FAILURE/
4780	104774	103	120	125	EM28:	.ASCIZ /CORRECTION FAILURE WITH ECC ENABLED ON FORCED SBE/
4781	105027	103	117	122	EM29:	.ASCIZ /WRITE BYTE (MOVB) WITH ECC ENABLED FAILED TO CLEAR DATA AT/<CRLF>
4782	105111	127	122	111	EM30:	.ASCIZ /FORCED SBE LOCATION/
4783	105204	106	117	122		
4784	105230	101	123	122	EM31:	.ASCIZ /ASRB (R3)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
4785	105330	115	117	126	EM32:	.ASCIZ /MOVB #360,(R2)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
4786	105435	115	117	126	EM33:	.ASCIZ /MOV #177400,(R1) WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
4787	105543	123	102	105	EM34:	.ASCIZ /SBE DETECTED BY MK11 BUT NOT ISOLATED BY PROGRAM/
4788	105624	125	116	105	EM35:	.ASCIZ /UNEXPECTED CORRECTION WITH ECC DISABLE ON FORCED SBE/
4789	105711	101	120	124	EM36:	.ASCIZ /APT SIZE DISAGREES WITH PROGRAM SIZING/
4790	105760	122	125	116	EM37:	.ASCIZ /RUNNING CONTENTION INCREMENT FAILURE/
4791	106025	101	123	122	EM38:	.ASCIZ /ASRB LOCK FAILURE (1 & ONLY 1 CPU SHOULD HAVE LOCKED) (LOCKED = ONES)/
4792	106133	101	123	122	EM39:	.ASCIZ /ASRB LOCK BALANCE FAILURE (# OF LOCKS PER CPU BELOW)/
4793	106220	102	122	101	EM40:	.ASCIZ /BRANCH GOBBLE FAILED CONDITION CODES TEST/
4794	106272	123	124	101	EM41:	.ASCIZ /STALE DATA COULD NOT BE CREATED - CACHE FAILURE NOT MAIN MEMORY!!!/
4795	106374	103	101	103	EM42:	.ASCIZ /CACHE COULD NOT FLUSH STALE DATA - CACHE FAILURE NOT MAIN MEMORY!!!/
4796	106477	123	114	101	EM43:	.ASCIZ /SLAVE DID NOT RESPOND IN TIME - ABORTING MULTIPROCESSOR TESTS/
4797	106575	111	111	123	EM44:	.ASCIZ /IIST DID NOT RESPOND IN TIME - ABORTING MULTIPROCESSOR TESTS/
4798	106672	116	117	124	EM45:	.ASCIZ /NOT ENOUGH GOOD MEMORY FOR SLAVE - ABORTING MULTIPROCESSOR TESTS/
4799	106773	101	114	114	EM46:	.ASCIZ /ALL CPUS NOT SHARING A MEMORY BANK (ACCESSABLE = ONES)/
4800	107062	103	123	122	EM47:	.ASCIZ /CSR CONFIGURATION ERROR - PROBABLY CSR SWITCHES IN BOX ARE SET WRONG!/ /APT EXPECTS DIFFERENT NUMBER OF CPU'S THAN DIAGNOSTIC CAN BOOT/
4801	107170	101	120	124	EM48:	.ASCIZ /PORT CONTENTION BALANCED WITH 3 CPU'S RAISED TO HIGH PRIORITY/
4802	107267	120	117	122	EM49:	.ASCIZ /PORT CONTENTION BALANCED WITH 3 CPU'S RAISED TO HIGH PRIORITY/
4803					.EVEN	

```

4806 .SBTTL ERROR DATA HEADERS (DH)
4807 107366 040 040 120 DH1: .ASCIZ / PC DEV ADD GOOD BAD/
4808 107423 040 040 120 DH2: .ASCIZ / PC GD-CC BD-CC/
4809 107446 040 040 120 DH3: .ASCIZ / PC 1ST ADD # OF ERRORS/
4810 107502 040 040 120 DH4: .ASCIZ / PC/
4811 107507 040 040 120 DH5: .ASCIZ / PC CPUERR MEMERR HIADRS LOADRS/
4812 107555 040 040 040 .ASCIZ / MMRO MMR1 MMR2 MMR3/
4813 107615 040 040 120 DH6: .ASCIZ / PC APTCORE MJSIZE APTMOS MKSIZE/
4814 107664 040 040 120 DH7: .ASCIZ / PC BANK VADD PADD GOOD/
4815 107730 040 040 040 .ASCIZ / BAD XOR MAR BOX MTYPE INT PAT ARRAY/
4816 110006 104 105 126 DH8: .ASCIZ /DEV ADD GOOD BAD GDSWITCH BDSWITCH NOTE: (OFF = 1)/
4817 110106 040 040 120 DH9: .ASCIZ / PC CPUERR MEMERR CSRNUM/
4818 110145 040 040 122 DH10: .ASCIZ / RO R1 R2 R3 R4 R5 SP PSW/
4819 110243 040 040 120 DH11: .ASCIZ / PC CSR CSR+2/
4820 110272 040 103 123 DH12: .ASCIZ / CSR CSR+2/
4821 110307 040 040 120 DH13: .ASCIZ / PC BANK VADD PADD WROTE1 WROTE2/
4822 110364 040 103 110 .ASCIZ / CHKBITS CSR CSR+2/
4823 110415 115 105 115 DH14: .ASCIZ /MEMERR MAINT CONTRL HIADRS LOADRS CPUERR/
4824 110473 040 040 040 .ASCIZ / MMRO MMR1 MMR2 MMR3/
4825 110533 040 040 120 DH15: .ASCIZ / PC BANK PADD GD-WD1 GD-WD2 GD-CHK/
4826 110610 040 102 101 .ASCIZ / BAD-WD1 BAD-WD2 BAD-CHK MAR INT PAT/
4827 110655 040 040 120 DH16: .ASCIZ / PC BANK/
4828 110672 040 040 120 DH17: .ASCIZ / PC BANK MASTER SLAVE1 SLAVE2 SLAVE3/
4829 110746 040 040 120 DH18: .ASCIZ / PC BANK VADD PADD GOOD BAD/
4830 111023 040 040 120 DH19: .ASCIZ / PC ARRAY/
4831 111037 040 040 120 DH20: .ASCIZ / PC SLAVE#/
4832 111056 040 040 120 DH21: .ASCIZ / PC/
4833 111063 040 040 120 DH22: .ASCIZ / PC REAL# APT#/
4834 111111 040 040 120 DH23: .ASCIZ / PC BANK MASTER SLAVE1 SLAVE2 SLAVE3 LOW-CPU/
4835 .EVEN

```

```
4838 .SBTTL MESSAGES
4839 111176 200 103 105 MSG000: .ASCIZ <CRLF>/CEMKABO 1170 MAIN MEMORY DIAGNOSTIC <V14-00>/
4840 111254 200 040 040 MSG001: .ASCIZ <CRLF>/
4841 111336 200 040 040 MSG002: .ASCIZ <CRLF>/
4842 111413 200 040 040 MSG003: .ASCIZ <CRLF>/
4843 111455 040 040 040 .ASCIZ /
4844 111520 200 040 040 MSG004: .ASCIZ <CRLF>/
4845 111561 060 061 062 .ASCIZ /01234567012345670123456701234567/
4846 111626 200 105 122 MSG005: .ASCIZ <CRLF>/ERRORS /
4847 111640 200 125 116 MSG006: .ASCIZ <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>
4848 111675 200 111 116 MSG007: .ASCIZ <CRLF>/INTRLV /
4849 111707 200 103 120 MSG008: .ASCIZ <CRLF>/CPU MAP / ;HEXADECIMAL(1=MASTER;2=SLAVE1;4=SLAVE2;8=SLAVE3)
4850 111721 200 115 105 MSG009: .ASCIZ <CRLF>/MEMTYPE / ;MEMORY TYPE (0=MJ11;1=MK11)
4851 111733 200 120 122 MSG010: .ASCIZ <CRLF>/PROTECT / ;MEMORY PROTECTED
4852 111745 040 040 040 MSG011: .ASCIZ / 0 1 2 3 4 5 6/
4853 112033 064 065 066 MSG012: .ASCIZ /4567012345670123456701234567012345670123456701234567/
4854 112130 130 000 MSG013: .ASCIZ /X/
4855 112132 040 000 MSG014: .ASCIZ / / ;SPACE
4856 112134 000 000 MSG015: .BYTE 0,0 ;FOR SINGLE ASCII CHARACTERS & TERMINATOR
4857 112136 200 102 117 MSG016: .ASCIZ <CRLF>/BOX /
4858 112150 040 040 040 MSG017: .ASCIZ / / ;8 SPACES
4859 112161 040 040 000 MSG018: .ASCIZ / / ;2 SPACES
4860 112164 040 040 040 MSG019: .ASCIZ / / ;3 SPACES
4861 112170 200 106 111 MSG020: .ASCIZ <CRLF>/FIELD SERVICE COMMAND MODE/
4862 112224 200 103 117 MSG021: .ASCIZ <CRLF>/COMMANDS AVAILABLE ARE:/
4863 112254 200 060 040 .ASCIZ <CRLF>/0 = EXIT FIELD SERVICE COMMANDS/
4864 112314 200 061 040 .ASCIZ <CRLF>/1 = READ CSR/
4865 112331 200 062 040 .ASCIZ <CRLF>/2 = LOAD CSR/
4866 112346 200 063 040 .ASCIZ <CRLF>/3 = EXAMINE MEMORY/
4867 112371 200 064 040 .ASCIZ <CRLF>/4 = MODIFY MEMORY/
4868 112413 200 065 040 .ASCIZ <CRLF>/5 = SELECT BANK, MARGIN, & PATTERN/
4869 112455 200 066 040 .ASCIZ <CRLF>/6 = TYPE CONFIGURATION MAP/
4870 112510 200 067 040 .ASCIZ <CRLF>/7 = BATTERY BACKUP - P/F TEST''
4871 112546 200 070 040 .ASCIZ <CRLF>/8 = SOB-A-LONG TEST/
4872 112572 200 071 040 .ASCIZ <CRLF>/9 = SUPER TIGHT SCOPE LOOP/
4873 112625 200 061 060 .ASCIZ <CRLF>/10= ERROR SUMMARY/
4874 112647 200 061 061 .ASCIZ <CRLF>/11= REFRESH TEST/
4875 112670 200 061 062 .ASCIZ <CRLF>/12= SET FILL COUNT/
4876 112713 200 061 063 .ASCIZ <CRLF>/13= ENTER KAMIKAZE MODE/
4877 112743 200 061 064 .ASCIZ <CRLF>/14= EXIT KAMIKAZE MODE/
4878 112772 200 061 065 .ASCIZ <CRLF>/15= TURN CACHE OFF/
4879 113015 200 061 066 .ASCIZ <CRLF>/16= TURN CACHE ON/
4880 113037 200 061 067 .ASCIZ <CRLF>/17= RUN ONLY MULTIPOINT TESTS/
4881 113074 200 061 070 .ASCIZ <CRLF>/18= RESUME RUNNING BOTH SINGLE & MULTIPOINT TESTS/
4882 113155 200 061 071 .ASCIZ <CRLF>/19= TEST ONLY SELECTED BANKS/
4883 113212 200 062 060 .ASCIZ <CRLF>/20= RESUME TESTING ALL BANKS/
4884 113247 015 012 000 .BYTE 15,12,0
4885 113252 200 127 110 MSG022: .ASCIZ <CRLF>/WHICH CSR(0-7)? /
4886 113274 200 103 123 MSG023: .ASCIZ <CRLF>/CSR - 1ST WORD? /
4887 113316 200 103 123 MSG024: .ASCIZ <CRLF>/CSR - 2ND WORD? /
4888 113340 200 124 110 MSG025: .ASCIZ <CRLF>/THIS CSR DOES NOT EXIST/
4889 113371 200 103 117 MSG026: .ASCIZ <CRLF>/COMMAND:/
4890 113403 200 117 114 MSG027: .ASCIZ <CRLF>/OLD CSR WAS/
4891 113420 200 103 123 MSG028: .ASCIZ <CRLF>/CSR IS NOW/
4892 113434 200 105 130 MSG029: .ASCIZ <CRLF>/EXAMINE MEMORY/
4893 113454 200 102 101 MSG030: .ASCIZ <CRLF>/BANK(0-167)? /
4894 113473 200 120 110 MSG031: .ASCIZ <CRLF>/PHYSICAL ADDRESS(0-16777776)? /
```

4895	113533	200	120	101	MSG032:	.ASCIZ	<CRLF>/PARITY ABORT/<32>
4896	113552	200	124	111	MSG033:	.ASCIZ	<CRLF>/TIMEOUT TRAP/<32>
4897	113571	200	102	131	MSG034:	.ASCIZ	<CRLF>/BYPASSING CSR TESTS ON BOX /
4898	113625	000			MSG034:	.BYTE	0
4899	113626	040	103	117		.ASCIZ	/ CONTROLLER /
4900	113642	000			MSG034:	.BYTE	0
4901	113643	200	104	125		.ASCIZ	<CRLF>/DUE TO LACK OF SBE FREE LOCATIONS/<32>
4902	113707	121	126	000	MSG035:	.ASCIZ	/QV/
4903	113712	200	115	117	MSG036:	.ASCIZ	<CRLF>/MODIFY MEMORY/
4904	113731	200	117	114	MSG037:	.ASCIZ	<CRLF>/OLD DATA WAS /
4905	113750	200	104	101	MSG038:	.ASCIZ	<CRLF>/DATA IS NOW /
4906	113766	200	111	116	MSG039:	.ASCIZ	<CRLF>/INPUT NEW DATA? /
4907	114010	200	123	105	MSG040:	.ASCIZ	<CRLF>/SELECT BANK, MARGIN, & PATTERN TEST/
4908	114055	200	102	101	MSG041:	.ASCIZ	<CRLF>/BANK NOT ACCESSABLE/
4909	114102	200	120	101	MSG042:	.ASCIZ	<CRLF>/PATTERN(0-37)? /
4910	114123	200	120	101	MSG043:	.ASCIZ	<CRLF>/PATTERN 0 DATA IS? /
4911	114150	200	115	101	MSG044:	.ASCIZ	<CRLF>/MARGIN(0,2,3,4,5)? /
4912	114175	200	115	101	MSG045:	.ASCIZ	<CRLF>/MARGINS INHIBITED BY SW11! /
4913	114231	200	124	117	MSG046:	.ASCIZ	<CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
4914	114264	200	124	105	MSG047:	.ASCIZ	<CRLF>/TEST COMPLETE/
4915	114303	040	116	117	MSG048:	.ASCIZ	/ NOT AVAILABLE NOW - TRY LATER! /
4916	114343	200	102	101	MSG049:	.ASCIZ	<CRLF>/BANK REQUIRES RELOCATION/
4917	114375	200	102	101	MSG050:	.ASCIZ	<CRLF>/BATTERY BACKUP TEST/
4918	114421	200	127	122		.ASCIZ	<CRLF>/WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND/
4919	114503	200	120	117	MSG051:	.ASCIZ	<CRLF>/POWER RECOVERY/
4920	114523	200	122	105	MSG052:	.ASCIZ	<CRLF>/REMOVE SYSTEM POWER FOR/
4921	114554	040	123	105	MSG053:	.ASCIZ	/ SECONDS MAX! /
4922	114572	200	116	117	MSG054:	.ASCIZ	<CRLF>/NOW STARTING READ TEST OF MEMORY BANKS/
4923	114642	200	123	117	MSG055:	.ASCIZ	<CRLF>/SOB-A-LONG TEST/
4924	114663	200	102	105	MSG056:	.ASCIZ	<CRLF>/BELL = EACH PASS COMPLETE/
4925	114716	040	077	040	MSG057:	.ASCIZ	/ ? /
4926	114722	200	040	040	MSG058:	.ASCIZ	<CRLF>/ CSR CSR+2 CSR CSR+2 .../
4927	114766	200	123	125	MSG059:	.ASCIZ	<CRLF>/SUPER TIGHT SCOPE LOOP/
4928	115016	200	123	127	MSG060:	.ASCIZ	<CRLF>/SWREG ----> MEM, MEM ----> DISPLAY/
4929	115060	077	077	077	MSG061:	.ASCIZ	/?????/
4930	115067	111	116	120	MSG062:	.ASCIZ	/INPUT MUST BE A/
4931	115107	116	040	117	MSG063:	.ASCIZ	/N OCTAL /
4932	115120	116	125	115	MSG064:	.ASCIZ	/NUMBER/<CRLF>
4933	115130	040	104	105	MSG065:	.ASCIZ	/ DECIMAL /
4934	115142	200	105	122	MSG066:	.ASCIZ	<CRLF>/ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN/
4935	115225	106	101	124	MSG067:	.ASCIZ	/FATAL /
4936	115234	115	112	061	MSG068:	.ASCIZ	/MJ11/
4937	115241	115	113	061	MSG069:	.ASCIZ	/MK11/
4938	115246	113	040	127	MSG070:	.ASCIZ	/K WORDS OF MEMORY/<CRLF>
4939	115271	113	040	117	MSG071:	.ASCIZ	/K OF MJ11/<CRLF>
4940	115304	113	040	117	MSG072:	.ASCIZ	/K OF MK11/<CRLF>
4941	115317	200	122	105	MSG073:	.ASCIZ	<CRLF>/REFRESH TEST/
4942						.EVEN	;THIS STAYS WITH MSG074
4943	115336	077	077	000	MSG074:	.ASCIZ	/??/
4944	115341	200	122	105	MSG075:	.ASCIZ	<CRLF>/RELOCATION NOT POSSIBLE/<32>
4945	115373	200	040	040	MSG076:	.ASCIZ	<CRLF>/ BANK ERRORS/<CRLF>
4946	115414	200	105	116	MSG077:	.ASCIZ	<CRLF>/END PASS #/
4947	115430	200	127	101	MSG078:	.ASCIZ	<CRLF>/WARNING! ECC IS DISABLED ON BOX /<32>
4948	115474	040	105	122	MSG079:	.ASCIZ	/ ERROR(S) DETECTED/<CRLF>
4949	115520	200	105	116	MSG080:	.ASCIZ	<CRLF>/ENTER # OF SLAVE CPU'S (0-3)? /
4950	115560	200	123	124	MSG081:	.ASCIZ	<CRLF>/START SLAVE #/
4951	115577	040	101	124	MSG082:	.ASCIZ	/ AT ADDRESS 200/

```

4952 115617 200 120 122 MSG084: .ASCIZ <CRLF>/PROCEEDING AS A SINGLE PORT TEST/
4953 115661 200 106 111 MSG085: .ASCIZ <CRLF>/FILL COUNT(OCTAL)? /
4954 115706 200 115 125 MSG086: .ASCIZ <CRLF>/MULTIPOINT DIRECTORY/
4955 115732 200 115 101 .ASCIZ <CRLF>/MASTER = /
4956 115745 200 123 114 MSG087: .ASCIZ <CRLF>/SLAVE/
4957 115753 060 .BYTE '0
4958 115754 040 075 040 .ASCIZ / = /
4959 115760 200 113 105 MSG088: .ASCIZ <CRLF>/KERNEL STACK/
4960 115776 200 123 125 MSG089: .ASCIZ <CRLF>/SUPERVISOR STACK/
4961 116020 200 125 123 MSG090: .ASCIZ <CRLF>/USER STACK/
4962 116034 040 111 123 MSG091: .ASCIZ / IS EMPTY/
4963 116046 122 105 114 MSG092: .ASCIZ /RELOCATED /
4964 116062 102 101 116 MSG093: .ASCIZ /BANK=/
4965 116070 040 040 115 MSG094: .ASCIZ / MAR=/
4966 116077 040 040 120 MSG095: .ASCIZ / PAT=/
4967 116106 200 114 117 MSG096: .ASCIZ <CRLF>/LOADERS BEING DESTROYED/
4968 116137 200 127 101 MSG097: .ASCIZ <CRLF>/WARNING! ILLEGAL INTERLEAVE SELECTED (FIELD SERVICE ONLY)/<32>
4969 116234 200 123 114 MSG098: .ASCIZ <CRLF>/SLAVE/
4970 116242 000 MSGA98: .BYTE 0
4971 116243 076 000 .ASCIZ />/
4972 116245 200 115 101 MSG099: .ASCIZ <CRLF>/MASTER>/
4973 116256 200 115 125 MSG100: .ASCIZ <CRLF>/MULTIPROCESSOR TEST #/
4974 116305 200 105 116 MSG101: .ASCIZ <CRLF>/ENTERING KAMIKAZE MODE/
4975 116335 200 114 105 MSG102: .ASCIZ <CRLF>/LEAVING KAMIKAZE MODE/
4976 116364 200 114 105 MSG103: .ASCIZ <CRLF>/LEAVING FIELD SERVICE MODE/<CRLF>
4977 116421 032 000 MSG104: .BYTE 32,0 ;CONTROL Z
4978 116423 200 105 116 MSG105: .ASCIZ <CRLF>/ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINATE (177)/
4979 116527 200 103 101 MSG106: .ASCIZ <CRLF>/CACHE IS OFF/
4980 116545 200 103 101 MSG107: .ASCIZ <CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
4981 116622 200 117 116 MSG108: .ASCIZ <CRLF>/ONLY MULTIPOINT TESTS WILL BE RUN/
4982 116664 200 102 117 MSG109: .ASCIZ <CRLF>/BOTH SINGLE & MULTIPOINT TESTS WILL BE RUN/
4983 116737 200 117 116 MSG110: .ASCIZ <CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
4984 117003 200 101 114 MSG111: .ASCIZ <CRLF>/ALL BANKS WILL BE TESTED/
4985 117035 200 120 122 MSG112: .ASCIZ <CRLF>/PROGRAM HALTING BECAUSE OF SWITCH #8/
4986 117103 040 115 101 MSG113: .ASCIZ / MASTER/
4987 117113 040 123 114 MSG114: .ASCIZ / SLAVE1/
4988 117123 040 123 114 MSG115: .ASCIZ / SLAVE2/
4989 117133 040 123 114 MSG116: .ASCIZ / SLAVE3/
4990 .EVEN
4996 117144 $$$END
4997 117144 END:
4998 117144 000406 .PRINT 60000-SUPLIMIT ;SUPERVISOR ADDRESSES LEFT
5003 117144 002004 .PRINT 100000-SLAVEND ;ADDRESSES LEFT IN SLAVE UNIQUE CODE
5008 117144 020634 .PRINT 140000-END ;ADDRESSES LEFT IN 24k
5012 000200 .END START3
  
```

SYMBOL TABLE

ABORTF	002142	BIT4	= 000020	B144	062364	B36	024446	CMD20A	050576
ACFLAG	002124	BIT5	= 000040	B145	062450	B37	024452	CMD5A	045456
ACK	002672	BIT6	= 000100	B146	062572	B4	010550	CMD5B	045666
ACTFLA	002442	BIT7	= 000200	B147	062630	B40	024616	CMD5C	046052
ADDPAR	014570	BIT8	= 000400	B15	013004	B41	024624	CMD7A	046336
ADDRES	002040	BIT9	= 001000	B150	062632	B42	025034	CMD8A	046550
ADD..0	071152	BLOCK1	043710	B151	062674	B43	033412	CMD8B	046740
ADD..1	071162	BLOCK2	043730	B152	063032	B44	033450	CMD8C	047014
ADD..2	071202	BLOCK3	043744	B153	063070	B45	033464	CMD9A	047240
ADD..3	071232	BMFLAG	002136	B154	063120	B46	033604	CMD9B	047456
ADD..4	071262	BOOT	043416	B155	063170	B47	033764	CMD9C	047464
ADD..5	071264	BOOTLO=	000012	B156	063172	B5	010554	CMD9LO	047444
ADD..6	071304	BOOT1	043460	B157	063330	B50	034006	CONFGE	002576
ADD..7	071334	BRGOBB	035112	B16	013014	B51	040014	CONFGT	006666
ALLCPU	002510	B0	005500	B160	063404	B52	040020	CONF G1	006304
APTCOR	002522	B1	007742	B161	063410	B53	040160	CONF G3	006520
APTDOW	043640	B10	011352	B162	063472	B54	040164	CONF G5	006712
APTFLA	002444	B100	053774	B163	063522	B55	040450	CONF IE	004336
APTHAN	015050	B101	054102	B164	063602	B56	040454	CONF IG	003016
APTHLT	043706	B102	054146	B165	063604	B57	041144	CONF SP	006754
APTMOS	002524	B103	054226	B166	063754	B6	010642	CONTF L	002332
APTSIZ	002544	B104	054234	B167	064016	B60	041152	CONTP	073112
ASRBAL=	000110	B105	054256	B17	013016	B61	041302	CONTRL=	177746
A3	= 000002	B106	054370	B170	064052	B62	041306	CONTS	073544
BACKGN	035344	B107	054416	B171	064122	B63	042530	CONTS1	073052
BAD	002056	B11	011360	B172	064124	B64	046374	CONTS2	073546
BADCSR	002000	B110	054472	B173	064246	B65	046744	CONTT	073404
BADPC	002026	B111	054474	B174	064260	B66	047612	COREHI	043300
BADPSW	002036	B112	055374	B175	064316	B67	050170	COUNT	002462
BADSP	002032	B113	055400	B176	064372	B7	010732	CPUBIT	002114
BADSTA	036770	B114	055700	B177	064512	B70	050512	CPUERR=	177766
BADXOR	002064	B115	055706	B2	010312	B71	050512	CPUID	011656
BAD2	002060	B116	057452	B20	013124	B72	050600	CR	= 000015
BAD3	002062	B117	057474	B200	070344	B73	050752	CRLF	= 000200
BAFPAF	015204	B12	012140	B201	070436	B74	053016	CSR	002144
BAFPAR	015320	B120	057622	B202	070736	B75	053024	CSRCAS	020114
BAKPAT	002762	B121	057756	B203	070740	B76	053660	CSRFB	002214
BALANC=	000155	B122	060072	B204	071610	B77	053710	CSRFB A	002410
BANK	002106	B123	060244	B205	071614	CACHK	002700	CSRFB IR	002336
BANKIN	002110	B124	060256	B206	071730	CACHOF=	104424	CSRINC	002414
BANKMO	042072	B125	060264	B207	072012	CACHON=	104423	CSRIND	002416
BANKOK	042730	B126	060514	B21	013220	CACHVE=	000114	CSRLB	002234
BAWPAF	015434	B127	060754	B210	072076	CBCSR =	104474	CSRLBA	002412
BAWPAR	015572	B13	012144	B211	072174	CB1CSR=	104475	CSRL00	002420
BDSWIT	002500	B130	061340	B22	013270	CHECK	002404	CSRNO	002256
BEGIN	005752	B131	061352	B23	013632	CHKDIS=	104504	CSROUD	002417
BGTEST	035110	B132	061412	B24	014144	CHKGEN	040540	CSROUT	040442
BIT0	= 000001	B133	061520	B25	014512	CHKTAB	040640	CSRREL	002254
BIT1	= 000002	B134	061572	B26	015114	CHK1DI=	104505	CSRSTU=	000031
BIT10	= 002000	B135	061724	B27	016722	CKEND	073316	CSRO	002154
BIT11	= 004000	B136	061736	B3	010316	CKSWR =	104410	CSR2	002174
BIT12	= 010000	B137	061774	B30	016744	CLRCR=	104502	DATBUF	002340
BIT13	= 020000	B14	012556	B31	016750	CLR1CS=	104503	DBEMSK	002354
BIT14	= 040000	B140	062102	B32	016762	CMD11A	047774	DDISP =	177570
BIT15	= 100000	B141	062154	B33	017044	CMD11B	050164	DECMAR	043152
BIT2	= 000004	B142	062312	B34	017060	CMD11C	050240	DEENER=	104421
BIT3	= 000010	B143	062350	B35	023232	CMD19L=	000070	DETAIL	071364

SYMBOL TABLE									
DEFLA	002326	DT13	102504	EM37	105760	E121	060040	E203	071110
DETPSW	002324	DT14	102530	EM38	106025	E122	060212	E204	071670
DETR0	002306	DT16	102556	EM39	106133	E123	060460	E205	071670
DETR1	002310	DT17	102610	EM4	103311	E124	060450	E206	071760
DETR2	002312	DT18	102616	EM40	106220	E125	060432	E207	072044
DETR3	002314	DT19	102634	EM41	106272	E126	060536	E21	013244
DETR4	002316	DT2	102264	EM42	106374	E127	061006	E210	072130
DETR5	002320	DT20	102652	EM43	106477	E13	012530	E211	072330
DETRSP	002322	DT21	102662	EM44	106575	E130	061702	E22	013620
DF1	102740	DT22	102700	EM45	106672	E131	061672	E23	013714
DF10	103076	DT23	102706	EM46	106773	E132	061656	E24	014264
DF11	103103	DT24	102714	EM47	107062	E133	061540	E25	014562
DF12	103114	DT25	102720	EM48	107170	E134	061646	E26	015202
DF13	103116	DT3	102302	EM49	107267	E135	062264	E27	017256
DF14	103125	DT4	102312	EM5	103357	E136	062254	E3	010364
DF2	102743	DT5	102322	EM6	103434	E137	062240	E30	017230
DF3	102767	DT6	102346	EM7	103461	E14	012716	E31	017166
DF4	103004	DT7	102362	EM8	103521	E140	062122	E32	017166
DF5	103021	DT8	102416	EM9	103556	E141	062230	E33	017152
DF6	103036	DT9	102432	ENASBE=	104506	E142	062544	E34	017122
DF7	103053	DUMMY	002304	ENA1SB=	104507	E143	062530	E35	023316
DF8	103067	DUMPCS=	000204	END	117144	E144	062436	E36	024574
DF9	103071	ECCDIS=	104470	ENERGI=	104420	E145	062520	E37	024560
DH1	107366	ECCINI=	104472	ERRMAX	002702	E146	062772	E4	010612
DH10	110145	ECC1DI=	104471	ERROR =	104000	E147	062756	E40	024746
DH11	110243	ECC1IN=	104473	ERRPC	002024	E15	013066	E41	024732
DH12	110272	EMTVEC=	000030	ERRPSW	002034	E150	062662	E42	025112
DH13	110307	EM1	103134	ERRSP	002030	E151	062744	E43	033756
DH14	110415	EM10	103633	ERRVEC=	000004	E152	063254	E44	033742
DH15	110533	EM11	103677	EVEN	002456	E153	063112	E45	033556
DH16	110655	EM12	103721	EXBANK	042422	E154	063170	E46	033726
DH17	110672	EM13	103745	EXCMD3	044726	E155	063240	E47	034002
DH18	110746	EM14	103777	EXCMD4	045236	E156	063240	E5	010612
DH19	111023	EM15	104043	EXIT	043522	E157	063576	E50	034026
DH2	107423	EM16	104111	EXIT2	043526	E16	013066	E51	040066
DH20	111037	EM17	104137	E0	005530	E160	063562	E52	040066
DH21	111056	EM18	104177	E1	010242	E161	063470	E53	040232
DH22	111063	EM19	104256	E10	011444	E162	063514	E54	040232
DH23	111111	EM2	103220	E100	054364	E163	063542	E55	040504
DH3	107446	EM20	104333	E101	054124	E164	063672	E56	040504
DH4	107502	EM21	104415	E102	054216	E165	063672	E57	041302
DH5	107507	EM22	104454	E103	054342	E166	064174	E6	010672
DH6	107615	EM23	104501	E104	054330	E167	064044	E60	041262
DH7	107664	EM24	104530	E105	054304	E17	013056	E61	041442
DH8	110006	EM25	104607	E106	054412	E170	064122	E62	041422
DH9	110106	EM26	104634	E107	054466	E171	064160	E63	042562
DIAGFL	002006	EM27	104705	E11	011432	E172	064160	E64	046456
DISPLA	002770	EM28	104774	E110	054542	E173	064466	E65	047006
DISPRE=	000174	EM29	105027	E111	054542	E174	064456	E66	047720
DISPTB	014370	EM3	103256	E112	055440	E175	064354	E67	050232
DNSLAV	055170	EM30	105111	E113	055440	E176	064442	E7	011076
DOBACK	015104	EM31	105230	E114	055746	E177	064570	E70	050552
DSWR =	177570	EM32	105330	E115	055746	E2	010364	E71	050552
DT1	102252	EM33	105435	E116	057602	E20	013160	E72	050626
DT10	102444	EM34	105543	E117	057516	E200	070402	E73	051042
DT11	102466	EM35	105624	E12	012530	E201	070474	E74	053074
DT12	102476	EM36	105711	E120	057752	E202	071110	E75	053062

E76	053734	IIII =	177777	KI2	002554	L1017	070542	L13	005376
E77	053730	IISTAC	002736	KI3	002556	L102	011414	L130	012700
FASTCI=	177640	IISTAD	002740	KI4	002560	L1020	070602	L131	012764
FATAL\$	002070	IISTIN	061074	KI5	002562	L1021	070570	L132	013046
FCMD10	047550	IISTOF	061174	KI6	002564	L1022	070576	L133	013046
FCMD11	047734	IISTVE	002742	KI7	002566	L1023	070624	L134	013160
FCMD12	050330	IITRAP	036764	KMAP =	104422	L1024	070640	L135	013174
FCMD13	050356	INCBNK	042772	KPFLAG	002122	L1025	070654	L136	013254
FCMD14	050400	INCMAR	043006	KSTACK	002706	L1026	070670	L137	013226
FCMD15	050420	INCPAT	042746	LAST =	157776	L1027	071072	L14	005470
FCMD16	050436	INCRPT	042746	LASTB	002262	L103	011416	L140	013620
FCMD17	050454	INTERL	002264	LASTER	002020	L1030	071072	L141	013472
FCMD18	050470	INTER0	006756	LBLS0 =	001057	L1031	071056	L142	013472
FCMD19	050502	INTER1	006760	LBLS1 =	001055	L1032	071122	L143	013472
FCMD20	050566	INTER2	006760	LBLS10=	000223	L1033	071200	L144	013416
FIELDS	043774	INTER3	006766	LBLS11=	000034	L1034	071230	L145	013434
FINDBA=	000061	INTER4	006766	LBLS2 =	001050	L1035	071260	L146	013452
FIRST =	060000	INTER5	006774	LBLS3 =	001051	L1036	071302	L147	013470
FIRSTB	002260	INTER6	006766	LBLS4 =	001032	L1037	071332	L15	005510
FIRSTP=	000010	INTER7	006774	LBLS5 =	000712	L104	011532	L150	013604
FLIPLO	002734	INTKRA	017610	LBLS6 =	000714	L1040	071362	L151	013534
FLIPWA	035214	INT..0	020234	LBLS7 =	000033	L1041	071646	L152	013542
FLUSH	014656	INT..1	020260	LCSR0U=	000055	L1042	072046	L153	013604
FSCMD0	044170	INT..2	020324	LCSRRE=	000114	L1043	072052	L154	013702
FSCMD1	044256	INT..3	020350	LCSRSA=	000112	L1044	072132	L155	013656
FSCMD2	044346	INT..4	020414	LF =	000012	L1045	072136	L156	013702
FSCMD3	044520	INT..5	020440	LKS =	177546	L1046	072316	L157	014032
FSCMD4	044764	INT..6	020504	LOADBA	002532	L1047	072264	L16	005570
FSCMD5	045274	INT..7	020530	LOADCS=	104425	L105	011574	L160	014112
FSCMD6	046152	INT.10	020574	LOADER=	000057	L1050	072316	L161	014116
FSCMD7	046160	INT.11	020574	LOADHO	002710	L1051	072316	L162	014336
FSCMD8	046510	INT.12	020616	LOADRS=	177740	L1052	073034	L163	014254
FSCMD9	047104	INT.13	020642	LOCK	002670	L1053	073452	L164	014172
FSINFL	002542	INT.14	020706	LOOP	014342	L1054	073540	L165	014174
FSPAT	045752	INT.15	020732	LOWMAP	042026	L1055	073464	L166	014234
FSSTAC	002374	INT.16	020776	LWDBE =	000053	L1056	073600	L167	014250
FS1	044052	INT.17	021022	LWSBE =	000051	L1057	073602	L17	005654
FS7FLA	002546	INVALI=	104511	L0	004426	L106	011614	L170	014304
GBLENG=	000076	IOTVEC=	000020	L1	004422	L107	011634	L171	014306
GDSWIT	002476	ISECCO	017300	L10	005376	L11	005376	L172	014336
GETDAT	055030	J	002604	L100	011552	L110	011654	L173	014456
GETDA1	055156	KAMIKA	002010	L1000	067226	L111	011672	L174	014566
GETDIS	066514	KAMITE	026006	L1001	067240	L112	011714	L175	014546
GETSIZ	017736	KDIAG =	000010	L1002	067276	L113	011766	L176	014546
GOOD	002050	KDPAR0=	172360	L1003	067306	L114	012034	L177	014622
GOOD2	002052	KDPAR7=	172376	L1004	067630	L115	012130	L2	004536
GOOD3	002054	KDPDR7=	172336	L1005	070316	L116	012250	L20	005616
GTSWR =	104407	KERNEL=	104417	L1006	070234	L117	012372	L200	014622
HEADER	002724	KERSTK=	002000	L1007	070234	L12	005370	L201	014656
HIACBA	002514	KIPAR0=	172340	L101	011416	L120	012422	L202	014734
HIADRS=	177742	KIPAR4=	172350	L1010	070274	L121	012416	L203	014744
HIBANK	002512	KIPAR5=	172352	L1011	070262	L122	012420	L204	015100
HIPAT	042762	KIPAR6=	172354	L1012	070270	L123	012514	L205	015166
HT =	000011	KIPAR7=	172356	L1013	070360	L124	012474	L206	015306
HUNGMS	002572	KIPDR0=	172300	L1014	070366	L125	012474	L207	015422
HUNGTI	002574	KI0	002550	L1015	070452	L126	012550	L21	005634
I	002602	KI1	002552	L1016	070460	L127	012700	L210	015560

L211	015716	L274	022600	L356	025152	L44	007700	L521	045714
L212	016054	L275	022614	L357	025224	L440	040210	L522	045716
L213	016234	L276	022624	L36	006630	L441	040252	L523	046232
L214	016372	L277	022624	L360	025350	L442	040260	L524	046234
L215	016552	L3	005046	L361	025540	L443	040376	L525	046442
L216	016674	L30	006242	L362	025756	L444	040432	L526	046602
L217	017230	L300	022662	L363	025770	L445	040462	L527	046604
L22	005636	L301	022672	L364	026030	L446	041114	L53	010202
L220	017152	L302	022672	L365	026036	L447	041144	L530	046630
L221	017034	L303	022732	L366	026042	L45	007714	L531	046644
L222	017152	L304	022746	L367	033434	L450	041246	L532	046646
L223	017134	L305	022756	L37	006712	L451	041276	L533	046772
L224	017214	L306	022756	L370	033444	L452	041410	L534	047154
L225	017366	L307	023010	L371	033556	L453	041410	L535	047156
L226	017372	L31	006356	L372	033506	L454	041410	L536	047272
L227	017724	L310	023024	L373	033520	L455	041410	L537	047274
L23	005714	L311	023034	L374	033536	L456	041410	L54	010300
L230	020016	L312	023034	L375	033544	L457	041436	L540	047320
L231	020030	L313	023066	L376	033570	L46	010226	L541	047720
L232	020072	L314	023102	L377	033726	L460	041534	L542	047704
L233	020110	L315	023112	L4	005144	L461	042550	L543	047654
L234	020224	L316	023112	L40	006734	L462	042720	L544	050026
L235	020306	L317	023172	L400	033626	L463	043026	L545	050030
L236	020314	L32	006406	L401	033640	L464	043114	L546	050054
L237	020376	L320	023202	L402	033672	L465	043134	L547	050070
L24	005774	L321	023202	L403	033656	L466	043230	L55	010304
L240	020404	L322	023300	L404	033670	L467	043232	L550	050072
L241	020466	L323	023270	L405	033714	L47	010226	L551	050216
L242	020474	L324	023400	L406	033702	L470	043256	L552	050534
L243	020556	L325	023402	L407	033714	L471	043264	L553	050536
L244	020564	L326	023432	L41	007152	L472	043410	L554	050724
L245	020614	L327	023436	L410	034026	L473	043410	L555	050726
L246	020670	L33	006476	L411	035504	L474	043364	L556	051030
L247	020676	L330	023466	L412	036150	L475	043410	L557	050774
L25	006010	L331	023472	L413	036220	L476	043444	L56	010306
L250	020760	L332	023526	L414	036246	L477	043542	L560	051010
L251	020766	L333	023532	L415	036240	L5	005222	L561	051026
L252	021050	L334	023562	L416	036246	L50	010166	L562	051056
L253	021056	L335	023566	L417	037136	L500	043546	L563	051170
L254	021174	L336	023744	L42	007156	L501	043554	L564	051414
L255	021216	L337	023776	L420	037142	L502	043574	L565	051514
L256	021232	L34	006564	L421	037202	L503	043606	L566	051644
L257	021250	L340	024042	L422	037206	L504	044016	L567	051656
L26	006160	L341	024206	L423	037424	L505	044026	L57	010350
L260	021254	L342	024216	L424	037432	L506	044160	L570	052414
L261	021256	L343	024216	L425	037510	L507	044220	L571	052442
L262	021322	L344	024544	L426	037544	L51	010166	L572	052510
L263	021344	L345	024530	L427	037634	L510	044224	L573	052522
L264	021372	L346	024544	L43	007160	L511	044246	L574	053076
L265	021372	L347	024610	L430	037670	L512	044252	L575	053202
L266	021522	L35	006614	L431	037730	L513	045416	L576	053200
L267	021566	L350	024716	L432	037772	L514	045456	L577	053232
L27	006160	L351	024702	L433	040044	L515	045510	L6	005246
L270	022522	L352	024716	L434	040044	L516	045512	L60	010346
L271	022536	L353	025076	L435	040106	L517	045536	L600	053232
L272	022546	L354	025076	L436	040114	L52	010166	L601	053734
L273	022546	L355	025200	L437	040210	L520	045574	L602	053722

L603	053726	L666	061630	L75	011306	MHALT3	012760	MSG025	113340
L604	054350	L667	062240	L750	065774	MINDEX	002666	MSG026	113371
L605	054200	L67	011110	L751	066006	MJPAT	021600	MSG027	113403
L606	054272	L670	062040	L752	066030	MJSIZE	002466	MSG028	113420
L607	054314	L671	062066	L753	066076	MJTEST	021512	MSG029	113434
L61	010576	L672	062220	L754	066156	MKCONT	016664	MSG030	113454
L610	054450	L673	062212	L755	066172	MKCSRS	002334	MSG031	113473
L611	054526	L674	062530	L756	066174	MKCSRT	020124	MSG032	113533
L612	054532	L675	062402	L757	066250	MKFLAG	002126	MSG033	113552
L613	054626	L676	062502	L76	011320	MKPAT	021402	MSG034	113571
L614	054642	L677	062756	L760	066316	MKSIZE	002470	MSG035	113707
L615	054646	L7	005316	L761	066312	MKTEST	021222	MSG036	113712
L616	054664	L70	011224	L762	066316	MK11AD=	172100	MSG037	113731
L617	055024	L700	062726	L763	066350	MMR0 =	177572	MSG038	113750
L62	010672	L701	063010	L764	066360	MMR1 =	177574	MSG039	113766
L620	055174	L702	063240	L765	066624	MMR2 =	177576	MSG040	114010
L621	055312	L703	063152	L766	066746	MMR3 =	172516	MSG041	114055
L622	055416	L704	063224	L767	066746	MMTRAP	036740	MSG042	114102
L623	055550	L705	063230	L77	011332	MMVEC =	000250	MSG043	114123
L624	055724	L706	063272	L770	067116	MOVELO=	000124	MSG044	114150
L625	057152	L707	063306	L771	067050	MP1	012070	MSG045	114175
L626	057136	L71	011236	L772	067074	MP2	012224	MSG046	114231
L627	057152	L710	063562	L773	067076	MP3	012716	MSG047	114264
L63	011076	L711	063562	L774	067116	MP4	012552	MSG048	114303
L630	057314	L712	063454	L775	067300	MP5	012764	MSG049	114343
L631	057202	L713	063450	L776	067164	MRUNSE	053640	MSG050	114375
L632	057216	L714	063454	L777	067276	MSEEDH	002720	MSG051	114503
L633	057232	L715	063662	MAINT =	177750	MSEEDL	002722	MSG052	114523
L634	057326	L716	063662	MAPH0 =	170202	MSG034	113625	MSG053	114554
L635	057604	L717	063722	MAPH36=	170372	MSG087	115753	MSG054	114572
L636	057564	L72	011250	MAPH37=	170376	MSG098	116242	MSG055	114642
L637	060040	L720	064160	MAPL0 =	170200	MSG034	113642	MSG056	114663
L64	011076	L721	064104	MAPL1 =	170204	MSG000	111176	MSG057	114716
L640	057724	L722	064150	MAPL36=	170370	MSG001	111254	MSG058	114722
L641	060022	L723	064224	MAPPER	040700	MSG002	111336	MSG059	114766
L642	060174	L724	064230	MARGIN	002112	MSG003	111413	MSG060	115016
L643	060174	L725	064442	MASTER=	000020	MSG004	111520	MSG061	115060
L644	060416	L726	064336	MAST01	057372	MSG005	111626	MSG062	115067
L645	060366	L727	064344	MAST02	060046	MSG006	111640	MSG063	115107
L646	060554	L73	011262	MAST03	060220	MSG007	111675	MSG064	115120
L647	060572	L730	064424	MAST04	061322	MSG008	111707	MSG065	115130
L65	011060	L731	064552	MAST05	061706	MSG009	111721	MSG066	115142
L650	060610	L732	065346	MAST06	062270	MSG010	111733	MSG067	115225
L651	060626	L733	065360	MAST07	062550	MSG011	111745	MSG068	115234
L652	060644	L734	065376	MAST10	062776	MSG012	112033	MSG069	115241
L653	060662	L735	065400	MAST11	063260	MSG013	112130	MSG070	115246
L654	060700	L736	065420	MAST12	063732	MSG014	112132	MSG071	115271
L655	060716	L737	065432	MAST13	064200	MSG015	112134	MSG072	115304
L656	060734	L74	011274	MAST14	064204	MSG016	112136	MSG073	115317
L657	060746	L740	065450	MAST15	064472	MSG017	112150	MSG074	115336
L66	011060	L741	065452	MASWR	002674	MSG018	112161	MSG075	115341
L660	061162	L742	065466	MDISPL	002676	MSG019	112164	MSG076	115373
L661	061162	L743	065676	MEMDON	014410	MSG020	112170	MSG077	115414
L662	061656	L744	065704	MEMERR=	177744	MSG021	112224	MSG078	115430
L663	061456	L745	065732	MEMSIZ=	000004	MSG022	113252	MSG079	115474
L664	061504	L746	065750	MHALT1	004532	MSG023	113274	MSG080	115520
L665	061636	L747	065762	MHALT2	005770	MSG024	113316	MSG081	115560

MSG082	115577	MTPD03	026630	MT0030	024754	O.CT	075730	O.SVR	100360
MSG084	115617	MTPD21	033354	MT0031	025214	O.CTLC	076162	O.SVTT	100500
MSG085	115661	MTPD25	034412	MT0033	025340	O.CTLE	076242	O.T	075572
MSG086	115706	MTPD26	034666	MT0034	025530	O.CTLF	076210	O.TBIT	077644
MSG087	115745	MTPE25	034434	MT0035	025652	O.C1	100000	O.TBT =	000020
MSG088	115760	MTP000	026376	MT020X	023604	O.DCD	076520	O.TCL2	076442
MSG089	115776	MTP001	026422	MT020Y	023472	O.DCDA	077144	O.TCSR=	177564
MSG090	116020	MTP002	026454	MT020Z	023322	O.DCD1	076550	O.TDB =	177566
MSG091	116034	MTP005	026724	MT0999	025772	O.DCD3	076530	O.TL	101345
MSG092	116046	MTP006	026760	MULHLT	056150	O.DOT	075562	O.TRTC	101370
MSG093	116062	MTP007	027150	MULTIE	060746	O.ERR	076510	O.TVEC=	000014
MSG094	116070	MTP010	027250	MULTIO	002004	O.ERR1	077572	O.TYPE	101126
MSG095	116077	MTP011	027356	MULTIP	003014	O.ERR2	077164	O.TYP1	101004
MSG096	116106	MTP012	027766	MULTIU	056142	O.ERR3	077150	O.UIN	075752
MSG097	116137	MTP013	030344	MUT	002116	O.FIL	075576	O.UPC	075700
MSG098	116234	MTP014	030716	M13BAL	054472	O.FTYP	100730	O.URO	075662
MSG099	116245	MTP015	031312	NEMCNT	002072	O.GET	101006	O.USP	075676
MSG100	116256	MTP016	031700	NEWBAN	002400	O.GO	077576	O.UST	075702
MSG101	116305	MTP017	032302	NEWKER	042322	O.GOGO	077624	O.WRD	076760
MSG102	116335	MTP020	032360	NEWLOA	042370	O.GO2	077676	O.WRD1	077036
MSG103	116364	MTP022	033404	NOBALA=	000164	O.ID	101300	O.WST	101202
MSG104	116421	MTP025	034152	NOCLOC	002104	O.IDND=	101315	O.XXX	075564
MSG105	116423	MTP030	034704	NOECCF	002474	O.LG =	000020	PADDRE	002042
MSG106	116527	MTP031	034714	NOERRO	002530	O.LGCH	101321	PAFBAF	015730
MSG107	116545	MTP033	035054	NOFSMO	002526	O.MIN	076502	PAFBAW	016066
MSG108	116622	MTP034	035106	NOIIST	002756	O.MINS	075603	PARBAF	016246
MSG109	116664	MTP20A	032360	NONEM	002102	O.ODT	075774	PARBAW	016404
MSG110	116737	MTP20B	032376	NONEXI	036646	O.OFST	077466	PARCNT	002074
MSG111	117003	MTP20C	032430	NOPAR	002100	O.OLD	077154	PARITY	036454
MSG112	117035	MTP20D	032462	NOSCOP	002540	O.OP1	077160	PARTHE	002372
MSG113	117103	MTP20E	032514	NOTAB	002464	O.OP2	077220	PARVEC=	000114
MSG114	117113	MTP20F	032546	NULLFL	002436	O.OP2A	077226	PASFLG	002362
MSG115	117123	MTV020	023320	OLDCAC	002370	O.ORAB	076364	PATERR	002076
MSG116	117133	MT0000	021704	OLDMAR	002376	O.ORPC	076342	PATPLU	005472
MST11A	063602	MT0001	021740	ONCE	002732	O.ORRB	076374	PATTER	002120
MTA030	024760	MT0002	022000	ONES	002726	O.P	075573	PCBUMP	002406
MTEST	016564	MT0003	022050	O.ADR1	075706	O.PRI	075704	PCONF I	035402
MTPA03	026506	MT0004	022216	O.BACK	077302	O.PROC	077732	PCONFS	035662
MTPA04	026650	MT0005	022300	O.BD	101366	O.PR1	077764	PCONF1	035572
MTPA20	032572	MT0006	022360	O.BIAS	075604	O.RALL	077430	PCONF2	035630
MTPA21	033234	MT0007	022414	O.BKP =	000016	O.RCSR=	177560	PDP110	036752
MTPA24	034060	MT0010	022456	O.BKPT	077324	O.RDB =	177562	PENDBO	002330
MTPA25	034464	MT0011	022512	O.BK1	100060	O.REGT	076270	PERA05	064732
MTPA26	034576	MT0012	022570	O.BRK	100042	O.REM	100602	PERBNK	065564
MTPA32	034772	MT0013	022646	O.BW	075566	O.RORA	101272	PERECC	065650
MTPB03	026550	MT0014	022722	O.BYT	076772	O.RRST	101172	PERRAB	065402
MTPB04	026704	MT0015	023000	O.CAD	075560	O.RSB	100450	PERRAW	065330
MTPB21	033264	MT0016	023056	O.CADV	100634	O.RSR	100416	PERRA3	054652
MTPB24	034120	MT0017	023134	O.CLGT=	000024	O.RSTT	100534	PERRA7	065454
MTPB25	034504	MT0020	023156	O.CLSE	101214	O.S	075571	PERR01=	104427
MTPB26	034612	MT0021	023612	O.CMFD	075600	O.SCAN	076554	PERR02=	104430
MTPB32	035020	MT0022	023734	O.CR	101316	O.SCRN	075602	PERR03=	104431
MTPC03	026610	MT0023	023766	O.CRET	077140	O.SEMI	076742	PERR04=	104432
MTPC21	033320	MT0024	024032	O.CRLF	101246	O.SEQ	075570	PERR05	064726
MTPC24	034134	MT0025	024172	O.CRLS	101254	O.SMFD	075601	PERR06	064754
MTPC25	034536	MT0026	024240	O.CSR1	075574	O.SNGL	076446	PERR07=	104433
MTPC26	034646	MT0027	024370	O.CSR2	075575	O.STM =	000340	PERR10=	104434

PERR11=	104435	RESVEC=	000010	START2	000310	TAG73\$	067722	TST2	007004
PERR12=	104436	RLFLAG	002134	START3	000200	TAG74\$	067730	TST3	007160
PERR13=	104437	RRFLAG	002132	START4	000322	TAG75\$	067742	TST4	010310
PERR14=	104440	RTNVAL=	%000000	STOPCP	011720	TAG76\$	067754	TST5	014342
PERR15=	104441	RUNCOU=	000171	STOPOK	002520	TAG77\$	067776	TST6	014414
PERR16=	104442	SAMESA=	000015	STRIPE	002460	TAG78\$	070000	TST7	014456
PERR17=	104443	SAVREG=	104415	STRTDI	002364	TAG79\$	070006	TWOCSR	002600
PERR20=	104444	SBECRSR	002150	SUBAAA	005532	TAG80\$	070066	TYPDS =	104405
PERR21=	104445	SBEFLA	002022	SUBAAB	005200	TAG81\$	070100	TYPEIT=	104401
PERR22=	104446	SBEMSK	002350	SUBAAQ	007720	TAG82\$	070122	TYPOC =	104402
PERR23=	104447	SBETES	017374	SUBAAR	013160	TAG83\$	070324	TYPOS =	104403
PERR24=	104450	SCOPE =	000004	SUBAAS	007002	TAG84\$	070410	TYPS0 =	000000
PERR25=	104451	SDPAR0=	172260	SUBAAT	011334	TAG85\$	070502	TYPS1 =	000000
PERR26=	104452	SDPAR7=	172276	SUBAAU	014602	TAG86\$	070612	TYPS10=	000000
PERR27=	104453	SDPDR7=	172236	SUCCE\$	002424	TAG9\$	007232	TYPS11=	000002
PERR30=	104454	SECOND=	011661	SUPDOA	002360	TASK	002616	TYPS2 =	000000
PERR31=	104455	SEEDHI	002714	SUPDO1	026042	TASKDO	050726	TYPS3 =	000000
PERR32=	104456	SEEDLO	002716	SUPDO2	026056	TASKRE	050710	TYPS4 =	000000
PERR33=	104457	SELONL	002002	SUPDO3	026220	TASK00	051234	TYPS5 =	000000
PERR34=	104460	SETMAR	043212	SUPDO4	026234	TASK01	051236	TYPS6 =	000000
PERR35=	104461	SETPAT	042762	SUPDR0	002266	TASK02	051330	TYPS7 =	000002
PERR36=	104462	SHALT	004420	SUPDR1	002270	TASK03	051430	T12A	031700
PERR37=	104463	SHUTUP	043554	SUPDR2	002272	TASK04	051664	T12B	031722
PERR40=	104464	SIDE	002422	SUPDR3	002274	TASK05	051774	UDPAR0=	177660
PERR41=	104465	SIDEOK	021066	SUPDR4	002276	TASK06	052300	UDPAR1=	177662
PERR42=	104466	SIPARO=	172240	SUPDR5	002300	TASK07	052326	UDPAR7=	177676
PERR43=	104467	SIPAR3=	172246	SUPDR6	002302	TASK10	052336	UDPDR7=	177636
PERXOR	065540	SIPDRO=	172200	SUPLIM	057372	TASK11	052536	UIPAR0=	177640
PFLAG	002130	SIZE =	040000	SUPM13	053752	TASK12	052644	UIPAR2=	177644
PHYADD	002044	SIZELO=	177760	SUPSTK=	000740	TASK13	052726	UIPAR3=	177646
PORTCO	002646	SKIPKA	002012	SWAPAT	002764	TASK14	053276	UIPAR4=	177650
PORTDI	002606	SLAVEM	061244	SWR	002766	TASK15	053310	UIPAR5=	177652
PROTEC=	000002	SLAVEN	075774	SWREG =	000176	TASK16	053316	UIPAR6=	177654
PSW =	177776	SLAVER	002570	SW0 =	000001	TASK17	053334	UIPAR7=	177656
PTYBUF	002636	SLAVES	002730	SW1 =	000002	TASK20	053422	UIPDRO=	177600
PTYFLA	002626	SLAVE1=	000040	SW10 =	002000	TASK7A	052344	UNITOP	002516
PWLOCK	002760	SLAVE2=	000100	SW11 =	004000	TBG4\$	026314	UNRELO	041604
PWRVEC=	000024	SLAVE3=	000200	SW12 =	010000	TCFIG1	036012	UPPFLG	002363
QUE	072154	SLAVUP	061216	SW13 =	020000	TCONFI	035664	UP2	056154
QUE1	072222	SLFLAG	002656	SW14 =	040000	TEMP	002534	USERMA	042240
QUICK	002536	SLPAT	051570	SW15 =	100000	TESTAD	002506	USESTK=	000700
QVFLAG	002440	SLPAT0	051530	SW2 =	000004	TIME	002434	USP =	%000006
RANODD	034626	SLPAT1	051534	SW3 =	000010	TIMEOU	036726	WAITST	012034
RDCHR =	104411	SLPAT2	051542	SW4 =	000020	TKVEC =	000060	WAIT5	063674
RDDEC =	104414	SLPAT3	051550	SW5 =	000040	TMFLAG	002140	WAKEUP	054544
RDLIN =	104412	SLPAT4	051556	SW6 =	000100	TOOMAN	002472	WARN1	007320
RDOCT =	104413	SLPAT5	051564	SW7 =	000200	TRAPVE=	000034	WARN2	026522
READCS=	104426	SOBK	002704	SW8 =	000400	TRT =	000003	WARN3	026536
READON	002504	SOBLEN=	000056	SW9 =	001000	TSK05A	052240	WARN4	026562
REALPA	002366	SOF TPA	002744	SYSSIZ	011210	TSK13A	053100	WARN5	026576
REFRES	033760	SOURCE	002402	SYSTID=	177764	TSK13B	053136	WARN6	035364
REFSUB	034030	SPECIA	002432	TAG2\$	007366	TSK20A	053602	WARN7	024416
REGCOP	035204	SSP =	%000006	TAG3\$	007412	TSTBAN	007606	WASDBE=	104500
RELOCA	041130	ST =	177776	TAG4\$	026136	TSTDAT	002344	WASSBE=	104476
RELOC1	041456	STACK =	002000	TAG70\$	067636	TSTRD1	037434	WAS1DB=	104501
RESREG=	104416	START	004352	TAG71\$	067646	TSTREA=	104510	WAS1SB=	104477
RESTAR	002752	START1	000300	TAG72\$	067656	TST1	006260	WHI CHC	050630

WHOBOX 070672	\$DDW6 075234	\$KERNE 037020	\$PER02 064622	\$SWR = 163000
WHODUN= 000202	\$DEENE 037040	\$KMAP 041016	\$PER03 064650	\$SWREG 075156
WOOPEN 056512	\$DEVCT 075144	\$KS\$ = 000204	\$PER04 064700	\$T = 001060
WOOPS 056160	\$DEVN 075212	\$L = 000212	\$PER07 064762	\$TESTN 075140
WOOPSA 056542	\$DIDDO= 000000	\$LF = 003011	\$PER10 065004	\$TKB 002774
WOOPUP 056336	\$DOAGA 015040	\$LL = 000210	\$PER11 065034	\$TKS 002772
WORST 002712	\$DOAGN 015010	\$LOADC 037076	\$PER12 065054	\$TN = 000010
WRITEO 002502	\$DOWN 055532	\$LOAD1 037146	\$PER13 065076	\$TPB 003000
XXDPCH 002446	\$DTBL 073002	\$LPADR 002746	\$PER14 065116	\$TPFLG 002452
ZERO 002426	\$ECCDI 037466	\$LPERR 002750	\$PER15 065140	\$TPS 002776
ZEROS 002430	\$ECCIN 037554	\$LS\$ = 000000	\$PER16 065162	\$TRAP 075252
\$APTHD 075236	\$ECC1D 037522	\$MADR1 075166	\$PER17 065202	\$TRAP2 075274
\$AUTO 002066	\$ECC1I 037574	\$MADR2 075172	\$PER20 065220	\$TRPAD 075314
\$BANK 002015	\$ENASB 037612	\$MADR3 075176	\$PER21 065236	\$TSTM 075242
\$BASE 075210	\$ENATS 037646	\$MADR4 075202	\$PER22 065256	\$TSTRD 037310
\$BELL 003003	\$ENDAD 015000	\$MAIL 075134	\$PER23 065274	\$TTYIN 074200
\$CACHF 037066	\$ENERG 037030	\$MAMS1 075164	\$PER24 065312	\$TYPDS 072576
\$CACHN 037050	\$ENV 075154	\$MAMS2 075170	\$PER25 054570	\$TYPE 056752
\$CBCSR 037700	\$ENVM 075155	\$MAMS3 075174	\$PER26 065502	\$TYPEC 057076
\$CB1CS 037742	\$EOP 014662	\$MAMS4 075200	\$PER27 065522	\$TYPEX 057360
\$CDW1 075214	\$ERFLG 002016	\$MBADR 075240	\$PER30 054762	\$TYPOC 072374
\$CDW2 075216	\$ERROR 066662	\$MNEW 074254	\$PER31 065720	\$TYPON 072410
\$CHARC 057356	\$ERRTB 101372	\$MSGAD 075150	\$PER32 066016	\$TYPOS 072350
\$CHKDI 040354	\$ERRTY 067324	\$MSGLG 075152	\$PER33 066064	\$T1 = 000000
\$CHK1D 040410	\$ERTTL 002754	\$MSGTY 075134	\$PER34 066144	\$T2 = 001057
\$CKSWR 073022	\$ESCAP 002454	\$MSWR 074243	\$PER35 066176	\$UNIT 075146
\$CLRCS 040322	\$ETABL 075154	\$MTYP1 075165	\$PWRDN 055160	\$UNITM 075246
\$CLR1C 040340	\$ETEND 075236	\$MTYP2 075171	\$PWRUP 055536	\$USWR 075160
\$CMTAG 002000	\$EXHAL 043546	\$MTYP3 075175	\$QUES 003007	\$VECT1 075204
\$CMTGE 002700	\$ES\$ = 000001	\$MTYP4 075201	\$R = 177777	\$VECT2 075206
\$CNTLC 074224	\$FATAL 075136	\$NOTRA 075306	\$RAND 074720	\$WASDB 040146
\$CNTLG 074236	\$FILLC 003002	\$NULL 002450	\$RDCHR 073602	\$WASSB 040002
\$CNTLU 074231	\$FILLS 002451	\$NWTST= 000001	\$RDDEC 074436	\$WAS1D 040262
\$CPUOP 075162	\$F\$ = 000000	\$OCNT 072572	\$RDLIN 073722	\$WAS1S 040116
\$CRLF 003010	\$GTSWR 073160	\$OCTVL 075116	\$RDOCT 074266	\$XTSTR 066370
\$DBLK 073012	\$HALT 067136	\$OCT8 = 075122	\$READC 037220	\$YS\$ = 000000
\$DB20 075014	\$HALT2 075312	\$OMODE 072574	\$RESRE 074662	\$ZAP42 014760
\$DDW0 075220	\$HIBTS 075236	\$OVER 066502	\$SAVRE 074624	\$Z\$ = 000000
\$DDW1 075222	\$HIOCT 074434	\$OS\$ = 000000	\$SAVR6 056140	\$SS = 000000
\$DDW2 075224	\$ILLUP 056134	\$PASS 075142	\$SCOPE 066232	\$ST = 001040
\$DDW3 075226	\$INVAL 040510	\$PASTM 075244	\$STN = 000001	\$STT = 001042
\$DDW4 075230	\$ITEMB 002017	\$PATMA 002014	\$SVLAD 066466	\$OFILL 072573
\$DDW5 075232	\$IS\$ = 000001	\$PER01 064574	\$SV\$ = 000000	

. ABS. 117144 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 30712 WORDS (120 PAGES)
 DYNAMIC MEMORY: 21560 WORDS (82 PAGES)
 ELAPSED TIME: 01:51:14
 CEMKAB.BIC,CEMKAB.LS1/CRF:SYM/-SP=CEMKAB.MLB/ML,CEMKAB.MA1,CEMKAB.MA2

CEMKAB SYMBOL	CREATED BY	MACRO ON	PAGE 1	E 12	CR:F	V01	SEQ	0561		
SYMBOL	CROSS REFERENCE	REFERENCES								
ABORTF	002142	#38-1166	*281-6181	393-570	395-607	*395-625	448-1990	448-1993	448-2002	448-2005
ACFLAG	002124	#38-1159	71-2036	75-2113	*77-2125	77-2126	81-2210	94-2576	94-2590	102-2754
		109-2847	111-2863	113-2897	115-2931	117-2972	120-3018	122-3060	124-3109	126-3151
		131-3240	169-4136	169-4159	171-4190	313-6654	*324-6878	*324-6889	351-7479	353-7533
		357-7682	389-466	412-1076	420-1252	423-1317	425-1377	427-1425	429-1467	433-1534
		437-1623	439-1684							
ACK	002672	#40-1308	*45-1404	45-1410	*89-2385	90-2424	90-2431	*90-2440	*90-2479	
ACTFLA	002442	#38-1239	*52-1534	65-1826	69-2022	149-3655	149-3668	158-3888	158-3897	158-3905
		158-3915	160-3926	160-3935	162-3950	162-3989	162-3995	162-4002	162-4008	165-4091
		179-4326	335-7055	460-2263						
ADDPAR	014570	102-2751	#102-2768							
ADDRES	002040	#38-1134	*65-1820	*79-2174	*206-4742	*206-4756	*209-4814	*209-4828	*212-4901	*283-6213
		*385-408	*385-416	*393-569	*393-569	*393-591	*395-606	*395-606	395-609	395-619
		*445-1883	*445-1884	*445-1889	*445-1890	*445-1895	*445-1896	*445-1901	*445-1902	*445-1909
		*445-1917	*445-1922	*445-1922	*445-1927	*445-1932	*447-1938	*447-1943	*447-1948	*447-1953
		*447-1958	*447-1963	*447-1968	*447-1973	*447-1978	*447-1983	*452-2053	*454-2083	*454-2092
		468-2434	477-2631	477-2633	477-2633	477-2635	477-2635	477-2638	477-2640	477-2640
		477-2642	477-2642	560-4700	560-4703	560-4706	560-4708	560-4713	560-4721	
ADD..0	071152	#475-2604	#477-2629							
ADD..1	071162	475-2605	#477-2631							
ADD..2	071202	475-2606	#477-2633							
ADD..3	071232	475-2607	#477-2635							
ADD..4	071262	475-2608	#477-2637							
ADD..5	071264	475-2609	#477-2638							
ADD..6	071304	475-2610	#477-2640							
ADD..7	071334	475-2611	#477-2642							
ALLCPU	002510	#38-1261	*90-2496	94-2572	389-463	*414-1113	*414-1115	*414-1116	420-1249	423-1314
		425-1374	427-1422	429-1464	433-1529	437-1620	439-1681			
APTCOR	002522	#38-1266	*98-2686	98-2694	560-4705					
APTDOW	043640	52-1529	#335-7085	335-7088	*335-7090					
APTFLA	002444	#38-1240	*52-1528	54-1547	98-2665	158-3888	158-3897	158-3905	158-3915	160-3926
		160-3935	162-3950	165-4091	179-4326	335-7055	460-2263			
APTHAN	015050	92-2558	#107-2832	107-2838						
APTHLT	043706	#335-7092								
APTMO5	002524	#38-1267	*98-2689	98-2694	560-4705					
APTSIZ	002544	#38-1275	*52-1524	98-2665						
BACKGN	035344	154-3840	154-3851	156-3870	164-4025	165-4070	#270-5940	331-7015		
BAD	002056	#38-1140	*79-2176	*90-2447	*90-2484	*98-2703	*385-407	*385-415	*389-484	*389-530
		*393-572	*393-594	*395-609	*395-623	*408-975	*408-1018	*410-1050	*412-1084	*412-1098
		*420-1267	*420-1289	*423-1331	*423-1353	*425-1403	*427-1444	*429-1480	*435-1591	*435-1605
		*437-1638	*439-1703	*441-1728	*445-1885	*445-1891	*445-1898	*445-1903	*445-1908	*445-1918
		*445-1923	*445-1928	*445-1933	*447-1939	*447-1944	*447-1949	*447-1954	*447-1959	*447-1964
		*447-1969	*447-1974	*447-1979	*447-1984	450-2031	*452-2054	*454-2084	*454-2093	*454-2094
		*454-2113	474-2567	474-2570	474-2573	474-2576	560-4700	560-4706	560-4708	560-4716
		560-4720	560-4721	560-4723	560-4725					
BADCSR	002000	#38-1115	470-2475	*475-2597	*475-2598					
BADPC	002026	#38-1129	*283-6240	393-583	448-1991	448-2003	450-2014	452-2064	454-2082	454-2091
		454-2102	459-2234	459-2235	*459-2239					
BADPSW	002036	#38-1133	*283-6241	454-2113	459-2238					
BADSP	002032	#38-1131	*283-6238	*283-6239	459-2237					
BADSTA	036770	281-6182	281-6196	283-6223	283-6227	283-6230	#283-6237	393-583	445-1911	448-1991
		448-2003	450-2014	452-2064	454-2082	454-2091	454-2102	454-2112		

CEMKAB
SYMBOL
SYMBOL
BADXOR
BAD2
BAD3
BAFPAF
BAFPAF
BAKPAT
BANK

CREATED BY MACRO ON 1-AUG-79 AT 07:46
CROSS REFERENCE
VALUE

REFERENCES
#38-1143 *450-2031 *450-2032 560-4706
#38-1141 *393-595 560-4716
#38-1142 *393-597 *393-598 560-4716
99-2716 #111-2859 111-2888
99-2717 #113-2893 113-2922
#42-1342 *47-1451 164-4024 165-4069
#38-1152 *67-1842 *67-1849 67-1850

BANK IN
BANKMO
BANKOK
BAWPAF
BAWPAR
BDSWIT
BEGIN
BGTEST
BIT0
BIT1
BIT10
BIT11
BIT12
BIT13

002110
042072
042730
015434
015572
002500
005752
035110
= 000001
= 000002
= 002000
= 004000
= 010000
= 020000

69-1967 *69-2018 69-2018 *69-2020 *71-2034 *71-2042 71-2042 *75-2111 *75-2116
75-2116 75-2118 *77-2124 77-2126 *77-2127 77-2131 *81-2208 *81-2215 81-2215
*81-2216 81-2219 *94-2564 *94-2622 94-2622 *102-2752 *102-2761 102-2761 *109-2845
*109-2853 109-2853 *111-2860 *113-2894 *115-2928 *117-2969 *120-3013 *122-3055 *124-3104
*126-3146 131-3228 *131-3238 131-3241 *131-3264 131-3264 *131-3279 *169-4134 169-4138
*169-4150 169-4150 *169-4157 169-4160 *169-4173 169-4173 *171-4187 *171-4196 171-4196
*171-4206 173-4224 175-4244 177-4282 177-4303 182-4335 185-4369 302-6520 *313-6652
313-6655 *313-6665 313-6665 317-6725 *317-6726 *317-6730 *317-6734 324-6880 *327-6943
327-6944 344-7232 *344-7237 *344-7241 344-7244 344-7250 *344-7265 346-7271 *346-7276
*346-7280 346-7283 346-7289 *346-7317 348-7323 *348-7329 348-7330 348-7332 348-7360
*348-7435 348-7436 351-7448 *351-7477 351-7481 *351-7488 351-7488 *351-7491 353-7497
*353-7531 *353-7537 353-7537 *353-7544 353-7545 355-7551 *355-7561 *355-7565 355-7568
355-7587 *355-7613 355-7614 355-7619 *355-7624 355-7625 355-7633 *355-7639 355-7639
*355-7641 357-7646 *357-7680 *357-7686 357-7686 *357-7693 357-7694 367-68 367-83
*367-83 369-103 *369-103 371-147 *371-147 373-166 *373-166 373-168 375-234
*375-234 377-265 *377-265 379-305 *379-305 379-308 381-351 383-372 *383-372
383-374 385-388 *385-388 385-390 387-435 *389-462 389-467 389-477 *389-518
389-518 403-796 *403-797 403-798 *403-816 405-833 *405-834 405-835 *405-848
407-930 *412-1074 *412-1097 412-1097 *414-1167 *414-1171 414-1171 *414-1174 *420-1248
*420-1299 420-1299 *423-1313 *423-1363 423-1363 *425-1373 *425-1412 425-1412 *427-1421
*427-1453 427-1453 *429-1463 *429-1501 429-1501 *433-1528 *433-1571 433-1571 *437-1619
*437-1658 437-1658 *439-1680 *439-1711 439-1711 452-2040 457-2177 466-2382 466-2406
468-2436 475-2599 475-2599 489-3021
#38-1153 *64-1741 64-1742 *64-1762 *67-1854 67-1902 *69-1964 69-2007 81-2220
94-2566 313-6660 317-6728 317-6732 *324-6882 412-1092 414-1169 414-1176 433-1532
81-2226 107-2830 313-6658 317-6723 #319-6767 335-7079
120-3016 122-3058 124-3107 126-3149 #327-6918
99-2718 #115-2927 115-2955 115-2963
99-2719 #117-2968 117-2996 117-3004
#38-1257 *79-2183 560-4708
45-1414 47-1454 47-1455 #58-1618
#269-5862 269-5883
#11-315 64-1715 64-1735 79-2190 79-2191 281-6170 285-6254 285-6258 285-6263
285-6268 290-6338 290-6340 293-6367 293-6372 293-6377 293-6383 296-6425 296-6427
296-6433 296-6436 298-6460 298-6462 298-6473 298-6475 315-6711 315-6713 317-6743
317-6745 340-7179 344-7246 346-7285 355-7570 359-7723 359-7729 375-221 385-423
385-429 395-620 395-628 470-2482 474-2548
#11-314 79-2191 79-2195 102-2768 133-3295 139-3425 293-6355 293-6361 293-6377
293-6383 293-6390 293-6397 300-6489 300-6496
#11-305 77-2147
#11-304 67-1909 67-1928 89-2399 276-6016 313-6673 324-6904 395-621 395-627
408-1007 416-1202 452-2042
#11-303 60-1651 60-1652 60-1658 64-1747 64-1751 69-2008 94-2593 139-3434
278-6140 313-6678 313-6678 324-6883 481-2717 481-2718 481-2734
#11-302 60-1651 60-1658 64-1744 64-1751 69-1966 96-2631 96-2632 139-3435

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES									
BIT14	= 040000	#11-301	177-4309	177-4315	278-6118	302-6522	313-6662	313-6676	313-6677	313-6692	317-6733
		96-2631	317-6737	317-6760	333-7047	361-7770	412-1094	481-2717	481-2734		
		250-5564	36-1089	64-1745	67-1868	67-1881	67-1898	67-1920	69-1966	69-1969	
		319-6807	96-2632	135-3332	146-3630	173-4225	175-4251	177-4284	182-4357	185-4391	
		365-22	250-5565	290-6345	290-6347	313-6676	313-6677	315-6699	317-6740	319-6791	
		373-169	319-6807	324-6910	335-7087	344-7251	346-7291	346-7309	355-7602	361-7756	361-7769
		383-375	365-30	365-59	367-67	367-81	369-101	371-146	371-158	373-165	
		395-608	373-190	375-232	377-263	377-280	379-304	379-312	379-333	383-371	
BIT15	= 100000	#11-300	383-375	385-387	385-398	387-441	387-452	389-468	389-496	393-571	393-593
		135-3349	395-608	395-622	398-670	398-674	401-756	401-760	403-799	405-836	407-925
		298-6449	#11-300	56-1579	56-1580	56-1603	81-2206	81-2221	83-2274	83-2285	83-2298
		319-6807	135-3361	135-3364	135-3364	135-3368	162-3979	288-6284	288-6295	290-6337	290-6347
		416-1189	313-6661	313-6673	313-6674	315-6699	315-6706	317-6729	317-6740	319-6791	
		#11-313	319-6807	324-6898	365-60	379-306	398-670	401-760	412-1093	414-1170	414-1177
		300-6489	416-1189	416-1217	454-2103	457-2190	468-2439	470-2478			
		477-2640	#11-313	79-2190	79-2192	83-2296	139-3425	281-6185	285-6268	293-6390	293-6397
		#11-312	300-6489	300-6496	340-7179	359-7723	439-1670	477-2631	477-2633	477-2635	477-2638
		285-6268	477-2640	477-2642							
		340-7179	#11-312	64-1752	77-2150	79-2193	79-2194	90-2456	139-3424	139-3428	278-6114
		#11-311	285-6268	293-6356	293-6362	293-6378	293-6384	293-6391	293-6398	300-6490	300-6497
		135-3368	340-7179	359-7723	433-1533	439-1670	477-2633	477-2635	477-2640	477-2642	
		470-2478	#11-311	64-1768	64-1769	79-2192	79-2194	90-2434	90-2468	107-2826	135-3349
		#11-310	340-7179	290-6337	296-6414	296-6434	410-1038	412-1070	414-1113	414-1116	454-2103
		317-6750	470-2478								
		353-7523	#11-310	52-1520	62-1679	79-2195	79-2196	107-2826	315-6702	315-6710	317-6747
		#11-309	317-6750	333-7038	335-7081						
		357-7671	#11-309	79-2196	79-2197	273-5960	273-5993	348-7373	348-7430	351-7467	351-7473
		#11-308	353-7523	353-7542	355-7596	355-7611	357-7672	357-7691	373-177		
		357-7671	#11-308	52-1523	79-2193	79-2197	273-5959	348-7372	351-7466	353-7522	355-7595
		#11-307	357-7671	363-7777							
		#11-306	#11-307	64-1715	64-1746	89-2374	285-6268	340-7179	359-7723	385-423	385-429
		317-6745	#11-306	146-3647	146-3648	285-6254	285-6258	285-6268	315-6711	315-6713	317-6743
		67-1846	317-6745	340-7179	359-7723	375-221	470-2482	474-2548			
		162-3982	67-1846	83-2286	102-2751	152-3786	152-3795	152-3806	154-3826	154-3843	154-3854
		164-4041	162-3982	162-3987	162-3993	162-4000	162-4006	162-4012	164-4031	164-4034	164-4038
		270-5945	164-4041	165-4077	167-4110	167-4115	169-4129	171-4184	175-4258	177-4306	240-5406
		385-386	270-5945	290-6323	#337-7102	369-133	375-237	375-243	377-292	377-295	379-302
		#337-7107	385-386								
		154-3822	#337-7107	373-170							
		173-4226	BLOCK2	043730							
		#38-1164	BLOCK3	043744							
		*324-6909	BLOCK3	043744							
		#333-7024	BLOCK3	043744							
		333-7033	BLOCK3	043744							
		#269-5863	BLOCK3	043744							
		#42-1315	BLOCK3	043744							
		#9-193	BLOCK3	043744							
		92-2513	BLOCK3	043744							
		290-6332	BLOCK3	043744							
		385-419	BLOCK3	043744							
		#9-192	BLOCK3	043744							
		47-1475	BLOCK3	043744							
		65-1813	BLOCK3	043744							
		67-1871	BLOCK3	043744							
		67-1884	BLOCK3	043744							
		67-1901	BLOCK3	043744							
		67-1924	BLOCK3	043744							
		69-2028	BLOCK3	043744							
		75-2107	BLOCK3	043744							
		83-2255	BLOCK3	043744							
		290-6314	BLOCK3	043744							
		#375-257	BLOCK3	043744							
		403-805	BLOCK3	043744							
		*324-6906	BLOCK3	043744							
		403-802	BLOCK3	043744							
		*324-6879	BLOCK3	043744							
		403-802	BLOCK3	043744							
		#337-7111	BLOCK3	043744							
		313-6654	BLOCK3	043744							
		126-3153	BLOCK3	043744							
		117-2974	BLOCK3	043744							
		115-2933	BLOCK3	043744							
		315-6700	BLOCK3	043744							
		154-3828	BLOCK3	043744							
		154-3844	BLOCK3	043744							
		154-3855	BLOCK3	043744							
		165-4078	BLOCK3	043744							
		165-4079	BLOCK3	043744							
		167-4111	BLOCK3	043744							
		167-4112	BLOCK3	043744							
		154-3822	BLOCK3	043744							
		154-3827	BLOCK3	043744							
		154-3828	BLOCK3	043744							
		154-3844	BLOCK3	043744							
		154-3855	BLOCK3	043744							
		165-4078	BLOCK3	043744							
		165-4079	BLOCK3	043744							
		167-4111	BLOCK3	043744							
		167-4112	BLOCK3	043744							
		154-3822	BLOCK3	043744							
		154-3827	BLOCK3	043744							
		154-3828	BLOCK3	043744							
		154-3844	BLOCK3	043744							
		154-3855	BLOCK3	043744							
		165-4078	BLOCK3	043744							
		165-4079	BLOCK3	043744							
		167-4111	BLOCK3	043744							
		167-4112	BLOCK3	043744							
		154-3822	BLOCK3	043744							
		154-3827	BLOCK3	043744							
		154-3828	BLOCK3	043744							
		154-3844	BLOCK3	043744							
		154-3855	BLOCK3	043744							
		165-4078	BLOCK3	043744							
		165-4079	BLOCK3	043744							
		167-4111	BLOCK3	043744							
		167-4112	BLOCK3	043744							
		154-3822	BLOCK3	043744							
		154-3827	BLOCK3	043744							
		154-3828	BLOCK3	043744							
		154-3844	BLOCK3	043744							
		154-3855	BLOCK3	043744							
		165-4078	BLOCK3	043744							
		165-4079	BLOCK3	043744							
		167-4111	BLOCK3	043744							
		167-4112	BLOCK3	043744							

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES	90-2508	92-2530	131-3257	135-3372	135-3379	146-3645	182-4361	185-4395	248-5504
			90-2508	92-2530	131-3257	135-3372	135-3379	146-3645	182-4361	185-4395	248-5504
			359-7730	397-645							
CACHVE	= 000114	#11-330	11-331								
CBCSR	= 104474	#9-236									
CB1CSR	= 104475	#9-237	146-3629	206-4716	209-4802	212-4893	217-4966	220-5044	223-5121	260-5709	
CHECK	002404	#38-1223	*146-3627	293-6389	293-6396	*305-6551	560-4713				
CHKDIS	= 104504	#9-244	69-1971	69-1980	69-1988	69-1996					
CHKGEN	040540	206-4705	209-4793	212-4882	217-4954	220-5031	223-5109	258-5649	#305-6528		
CHKTAB	040640	305-6538	#307-6556								
CHK1DI	= 104505	#9-245	236-5291	236-5312	236-5320	238-5335	238-5356	238-5365	240-5381	240-5400	
		393-585									
CKEND	073316	487-2953	487-2955	487-2957	#487-2986						
CKSWR	= 104410	#9-175	456-2134	459-2214	460-2255	483-2755					
CLRCR	= 104502	#9-242	69-1974	69-1983	69-1991	69-1999	69-2013	#149-3658	169-4128	171-4199	
		#348-7386	#351-7453	#353-7514	#355-7557	#357-7663	369-132	375-254			
CLR1CS	= 104503	#9-243	133-3293	135-3341	135-3359	135-3375	146-3634	146-3643	162-3981	162-4014	
		206-4688	206-4733	206-4746	206-4777	209-4785	209-4804	209-4855	212-4862	212-4876	
		212-4895	214-4924	217-4948	217-4970	217-5005	220-5025	220-5046	220-5082	223-5125	
		225-5162	240-5411	258-5684	393-590						
CMD11A	047774	#357-7649	357-7653	357-7657							
CMD11B	050164	#357-7680	357-7688								
CMD11C	050240	357-7668	#357-7690								
CMD20A	050576	361-7746	#361-7768								
CMD5A	045456	#348-7349	348-7353	348-7357							
CMD5B	045666	#348-7381	348-7394								
CMD5C	046052	348-7365	348-7369	#348-7429							
CMD7A	046336	351-7463	#351-7472								
CMD8A	046550	#353-7500	353-7504	353-7508							
CMD8B	046740	#353-7531	353-7539								
CMD8C	047014	353-7519	#353-7541								
CMD9A	047240	#355-7573	355-7577	355-7581							
CMD9B	047456	355-7586	#355-7607								
CMD9C	047464	355-7592	#355-7610								
CMD9LO	047444	355-7585	#355-7603	355-7605	355-7607						
CONFGE	002576	#38-1288	*67-1910	*67-1929	94-2576	*94-2577	94-2617	*94-2618	96-2659		
CONFGT	006666	64-1705	#64-1767								
CONF G1	006304	#64-1707	64-1773								
CONF G3	006520	#64-1742	64-1765								
CONF G5	006712	64-1736	64-1766	#64-1772							
CONF IE	004336	#44-1398	50-1486								
CONF IG	003016	#44-1395	50-1484	*56-1579	*56-1580	*64-1743	*64-1744	*64-1745	*64-1747	*64-1752	
		*64-1756	*64-1761	*65-1822	67-1859	*67-1909	*67-1911	*67-1928	*67-1932	69-1965	
		69-1966	69-1966	*69-2008	*81-2206	*81-2221	94-2571	94-2579	94-2580	94-2581	
		94-2582	94-2591	94-2593	94-2610	96-2630	96-2631	96-2632	96-2632	177-4309	
		*177-4315	276-6016	276-6030	278-6051	278-6058	278-6090	278-6096	278-6113	278-6136	
		278-6140	*302-6522	*313-6661	*313-6662	313-6673	313-6674	313-6675	313-6676	313-6677	
		313-6677	313-6678	313-6678	*313-6692	*317-6729	*317-6733	*317-6737	324-6883	324-6888	
		324-6893	324-6898	324-6904	324-6910	348-7334	355-7628	355-7634	*361-7756	*361-7769	
		*361-7770	*412-1093	*412-1094	*414-1170	*414-1177	433-1533	*452-2042	*452-2044	*452-2046	
		452-2047									
CONF SP	006754	*64-1706	64-1767	#64-1777							
CONF L	002332	#38-1200	*99-2711	129-3203	*131-3278						

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES								
CONTP	073112	#487-2948	*508-3660	510-3708						
CONTRL	= 177746	#11-338	*45-1426	64-1704	*64-1775	*285-6262	*285-6263	*285-6268	288-6289	*288-6297
		290-6313	*290-6317	290-6331	*290-6334	338-7133	*340-7182	*385-423	*385-429	*395-621
		*395-627	397-644	*401-780	407-909	*407-945	483-2756	*483-2784	560-4714	
CONTS	073544	487-2951	#489-3033							
CONTS1	073052	#487-2938	489-3040							
CONTS2	073546	#489-3035	489-3036	489-3042						
CONTT	073404	479-2679	487-2943	#489-3007						
COREHI	043300	329-6957	329-6971	329-6984	331-7001	#331-7006	348-7367	348-7434		
COUNT	002462	#38-1250	*248-5505	248-5508	*248-5508	*248-5509	248-5510	*248-5517	*250-5526	250-5530
CPUBIT	002114	*250-5530	*250-5531	250-5532	*250-5551					
		#38-1155	45-1407	*50-1490	58-1625	62-1665	62-1682	65-1822	67-1911	67-1932
		69-1965	81-2205	83-2257	83-2261	83-2265	*89-2382	*90-2470	*90-2495	*90-2506
		94-2563	*94-2572	*94-2574	96-2630	310-6640	313-6675	324-6888	348-7334	365-4
		379-309	*389-463	*389-465	397-638	407-921	407-922	407-923	*410-1040	*410-1051
		*410-1060	*412-1072	*412-1085	*414-1112	416-1195	*420-1249	*420-1251	*423-1314	*423-1316
		*425-1374	*425-1376	*427-1422	*427-1424	*429-1464	*429-1466	*433-1529	*433-1531	*437-1620
		*437-1622	*439-1681	*439-1683	456-2135	487-2931				
CPUERR	= 177766	#11-348	*50-1502	*50-1513	*56-1612	*60-1646	64-1768	*64-1774	*65-1823	*67-1874
		*75-2098	*340-7194	*340-7198	*342-7210	*342-7227	*344-7266	*346-7318	*456-2158	*457-2195
		*460-2277	560-4704	560-4709	560-4714					
CPUID	011656	83-2259	83-2263	83-2267	#83-2271					
CR	= 000015	#11-271	407-937							
CRLF	= 000200	#11-272	407-890	564-4782	568-4839	568-4840	568-4841	568-4842	568-4844	568-4846
		568-4847	568-4848	568-4849	568-4850	568-4851	568-4857	568-4861	568-4862	568-4863
		568-4864	568-4865	568-4866	568-4867	568-4868	568-4869	568-4870	568-4871	568-4872
		568-4873	568-4874	568-4875	568-4876	568-4877	568-4878	568-4879	568-4880	568-4881
		568-4882	568-4883	568-4885	568-4886	568-4887	568-4888	568-4889	568-4890	568-4891
		568-4892	568-4893	568-4894	568-4895	568-4896	568-4897	568-4901	568-4903	568-4904
		568-4905	568-4906	568-4907	568-4908	568-4909	568-4910	568-4911	568-4912	568-4913
		568-4913	568-4914	568-4916	568-4917	568-4918	568-4919	568-4920	568-4922	568-4923
		568-4924	568-4926	568-4927	568-4928	568-4932	568-4934	568-4938	568-4939	568-4940
		568-4941	568-4944	568-4945	568-4945	568-4946	568-4947	568-4948	568-4949	568-4950
		568-4952	568-4953	568-4954	568-4955	568-4956	568-4959	568-4960	568-4961	568-4967
		568-4968	568-4969	568-4972	568-4973	568-4974	568-4975	568-4976	568-4976	568-4978
		568-4979	568-4980	568-4981	568-4982	568-4983	568-4984	568-4985		
CSR	002144	#38-1167	64-1712	64-1731	64-1732	64-1751	*69-1968	*69-1979	*69-1987	*69-1995
		*77-2142	*77-2143	*77-2145	*77-2146	*77-2147	*77-2149	*77-2149	*77-2150	77-2153
		*92-2549	*92-2550	133-3295	135-3349	135-3368	137-3396	*139-3424	*139-3425	*139-3428
		139-3429	139-3432	146-3637	149-3672	149-3673	236-5283	236-5284	*236-5290	236-5304
		236-5306	*236-5311	*236-5319	238-5326	*238-5334	238-5348	238-5350	*238-5355	*238-5364
		240-5372	*240-5380	240-5391	240-5393	*240-5399	288-6280	288-6282	288-6291	288-6293
		*290-6309	*290-6310	*290-6315	*290-6316	290-6337	290-6337	*290-6348	*290-6349	*293-6355
		*293-6356	*293-6357	*293-6361	*293-6362	*293-6363	*293-6367	*293-6368	*293-6372	*293-6373
		*293-6377	*293-6378	*293-6379	*293-6383	*293-6384	*293-6385	*293-6389	*293-6390	*293-6391
		*293-6392	*293-6396	*293-6397	*293-6398	*293-6399	296-6414	296-6434	298-6449	298-6468
		298-6470	*300-6480	*300-6480	*300-6484	*300-6484	*300-6489	*300-6490	*300-6491	*300-6496
		*300-6497	*300-6498	*342-7213	*342-7216	393-587	398-685	398-685	*400-739	*400-739
		454-2103	470-2474	470-2474	470-2478	470-2479	470-2482	*470-2495	*470-2495	479-2682
		479-2682	479-2691	479-2693	*479-2699	*479-2699	560-4711	560-4711	560-4712	560-4712
		560-4713	560-4713							
CSRCAS	020114	131-3255	#142-3448							

CEMKAB SYMBOL	VALUE	REFERENCES								
CSRFB	002214	#38-1171	*64-1733	475-2595						
CSRFBA	002410	#38-1225	131-3264	*137-3392	*144-3505	*144-3521	*144-3537	*144-3553	*144-3581	*144-3597
		*144-3613								
CSRF IR	002336	#38-1207	131-3246	*144-3499	*144-3507	*144-3509	*144-3515	*144-3523	*144-3525	*144-3531
		*144-3539	*144-3541	*144-3547	*144-3555	*144-3557	*144-3575	*144-3583	*144-3585	*144-3591
		*144-3599	*144-3601	*144-3607	*144-3615	*144-3617	146-3628	162-3975	162-3999	236-5285
		348-7323	*348-7382	*348-7433						
CSRINC	002414	#38-1227	131-3261	*144-3501	*144-3511	*144-3517	*144-3527	*144-3533	*144-3543	*144-3549
		*144-3559	*144-3577	*144-3587	*144-3593	*144-3603	*144-3609	*144-3619	*162-3964	162-3977
		236-5286								
CSRIND	002416	#38-1228	*131-3229	131-3232	*131-3274					
CSRLB	002234	#38-1172	*64-1738	475-2596						
CSRLBA	002412	#38-1226	131-3238	*137-3393						
CSRLOO	002420	#38-1230	*131-3231	131-3267	*131-3276	131-3276	133-3297			
CSRNO	002256	#38-1174	*64-1703	64-1729	64-1749	*64-1772	64-1773	*77-2138	*77-2161	77-2161
		*92-2551	*92-2556	92-2556	*133-3292	*162-3958	288-6277	290-6306	290-6325	*296-6410
		*296-6420	296-6420	*298-6445	*298-6455	298-6455	*302-6507	*302-6513	302-6513	338-7133
		*340-7178	351-7448	*351-7491	*363-7779	*363-7791	397-648	*398-681	*398-688	398-688
		*400-736	*400-743	400-743	*401-776	470-2474	*470-2475	*470-2495	475-2588	*475-2625
		479-2682	*479-2687	*479-2697	479-2697	*479-2699	560-4709			
CSROUD	002417	#38-1229	*131-3230	*131-3275	133-3287					
CSROUT	040442	293-6358	293-6369	293-6380	293-6393	300-6481	300-6492	#302-6503		
CSRREL	002254	#38-1173	*60-1640	*60-1645	288-6278	290-6307	290-6326			
CSRO	002154	#38-1169	*64-1731							
CSR2	002174	#38-1170	*64-1732	475-2600						
DATBUF	002340	#38-1208	*198-4553	*198-4554	198-4555	198-4556	198-4558	198-4563	198-4567	*198-4569
		*198-4569	*198-4572	*198-4573	198-4574	198-4575	198-4577	198-4582	198-4586	*198-4588
		*198-4588	*206-4690	*206-4691	206-4696	206-4697	206-4765	*206-4767	*206-4767	*209-4786
		*209-4787	209-4790	209-4791	209-4847	*209-4849	*209-4849	*212-4864	*212-4865	212-4870
		212-4871	*217-4936	*217-4937	217-4942	217-4943	*220-5013	*220-5014	220-5019	220-5020
		*223-5092	*223-5093	223-5098	223-5099	445-1919	445-1924			
DBEMSK	002354	#38-1211	*212-4868	*212-4869	212-4877	212-4879	212-4887	212-4889	214-4906	*214-4908
		*214-4908	*217-4940	*217-4941	217-4949	217-4951	217-4959	217-4961	217-4987	*217-4989
		*217-4989	*220-5017	*220-5018	220-5026	220-5028	220-5036	220-5038	220-5064	*220-5066
		*220-5066	*223-5096	*223-5097	223-5104	223-5106	223-5114	223-5116	225-5144	*225-5146
		*225-5146								
DDISP	= 177570	#11-265	42-1347	50-1497						
DECMAR	043152	113-2907	117-2984	124-3117	126-3161	#329-6976				
DEENER	= 104421	#9-188	60-1648	333-7037	335-7080					
DETAIL	071364	464-2360	#477-2645	477-2674						
DETFLA	002326	#38-1196	460-2262	464-2359	*477-2646	477-2647	477-2649	*481-2749		
DETSPW	002324	#38-1195	560-4710							
DETRO	002306	#38-1188	*477-2654	477-2663	560-4710					
DETR1	002310	#38-1189	477-2655	560-4710						
DETR2	002312	#38-1190	560-4710							
DETR3	002314	#38-1191	560-4710							
DETR4	002316	#38-1192	560-4710							
DETR5	002320	#38-1193	560-4710							
DETSP	002322	#38-1194	560-4710							
DF1	102740	#562-4729								
DF10	103076	550-4513	#562-4738							
DF11	103103	548-4421	550-4488	550-4493	550-4498	552-4520	552-4525	552-4540	556-4649	556-4654

CEMKAB
SYMBOL CROSS REFERENCE
SYMBOL VALUE

CREATED BY MACRO ON 1-AUG-79 AT 07:46

PAGE 7
CREF V01

K 12

SEQ 0567

SYMBOL	VALUE	REFERENCES
DF12	103114	#562-4739
DF13	103116	554-4622 #562-4740
DF14	103125	552-4555 552-4570 554-4587 556-4634 556-4639 556-4679 #562-4741
DF2	102743	558-4696 #562-4742
		548-4426 548-4441 548-4446 548-4451 548-4456 550-4473 550-4503 552-4530 552-4535
		552-4545 552-4550 554-4577 554-4627 556-4644 556-4664 556-4669 556-4674 558-4691
		#562-4730
DF3	102767	550-4463 552-4565 554-4592 554-4597 554-4607 554-4612 554-4617 556-4684 #562-4731
DF4	103004	550-4468 #562-4732
DF5	103021	550-4478 550-4508 554-4602 #562-4733
DF6	103036	550-4483 #562-4734
DF7	103053	552-4560 #562-4735
DF8	103067	554-4582 #562-4736
DF9	103071	548-4431 548-4436 #562-4737
DH1	107366	548-4424 548-4454 #566-4807
DH10	110145	550-4471 #566-4818
DH11	110243	550-4501 552-4533 #566-4819
DH12	110272	552-4528 #566-4820
DH13	110307	548-4419 550-4486 550-4491 550-4496 552-4518 552-4523 552-4538 #566-4821
DH14	110415	552-4543 #566-4823
DH15	110533	552-4558 #566-4825
DH16	110655	552-4563 554-4580 556-4682 #566-4827
DH17	110672	552-4553 552-4568 554-4585 556-4632 556-4637 556-4677 #566-4828
DH18	110746	556-4647 556-4652 #566-4829
DH19	111023	554-4620 #566-4830
DH2	107423	556-4642 #566-4808
DH20	111037	556-4662 556-4672 #566-4831
DH21	111056	556-4667 #566-4832
DH22	111063	558-4689 #566-4833
DH23	111111	558-4694 #566-4834
DH3	107446	548-4429 548-4434 #566-4809
DH4	107502	#566-4810
DH5	107507	548-4439 548-4444 548-4449 552-4548 #566-4811
DH6	107615	554-4625 #566-4813
DH7	107664	550-4461 550-4466 550-4476 550-4481 550-4506 554-4590 554-4595 554-4600 554-4605
		554-4610 554-4615 #566-4814
DH8	110006	550-4511 #566-4816
DH9	110106	554-4575 #566-4817
DIAGFL	002006	#38-1118 *164-4059 *164-4061 248-5522
DISPLA	002770	#42-1347 *50-1497 *50-1504 355-7604 365-34 375-223 389-461 401-774 408-954
		410-1035 412-1067 414-1180 420-1244 423-1309 425-1372 427-1420 429-1462 431-1524
		437-1618 439-1675 441-1720 457-2192
DISPRE	= 000174	#13-549 50-1504 *401-775 *457-2193
DISPTB	014370	99-2714 #99-2716
DNISLAV	055170	#397-639 397-640
DOBACK	015104	98-2708 102-2729 104-2781 #109-2843 414-1182
DSWR	= 177570	#11-264 42-1346 50-1496
DT1	102252	548-4425 548-4455 #560-4700
DT10	102444	550-4472 #560-4710
DT11	102466	550-4502 552-4534 #560-4711
DT12	102476	552-4529 #560-4712
DT13	102504	548-4420 550-4487 550-4492 550-4497 552-4519 552-4524 552-4539 #560-4713

SYMBOL CROSS REFERENCE

CREF V01

SEQ 0568

SYMBOL	VALUE	REFERENCES
DT14	102530	552-4544 #560-4714
DT16	102556	552-4559 #560-4715
DT17	102610	552-4564 554-4581 556-4683 #560-4717
DT18	102616	552-4569 556-4633 556-4638 #560-4718
DT19	102634	554-4586 #560-4719
DT2	102264	552-4554 556-4678 #560-4701
DT20	102652	556-4643 558-4690 #560-4720
DT21	102662	556-4648 556-4653 #560-4721
DT22	102700	554-4621 #560-4722
DT23	102706	556-4663 556-4673 #560-4723
DT24	102714	556-4668 #560-4724
DT25	102720	558-4695 #560-4725
DT3	102302	548-4430 #560-4702
DT4	102312	548-4435 #560-4703
DT5	102322	548-4440 548-4445 548-4450 552-4549 #560-4704
DT6	102346	554-4626 #560-4705
DT7	102362	550-4462 550-4467 550-4477 550-4482 550-4507 554-4591 554-4596 554-4601 554-4606
		554-4611 554-4616 #560-4706
DT8	102416	550-4512 #560-4708
DT9	102432	554-4576 #560-4709
DUMMY	002304	#38-1186 560-4701 560-4706 560-4706 560-4707 560-4707 560-4707 560-4707 560-4707 560-4707
		560-4707 560-4713 560-4713 560-4715 560-4715 560-4716 560-4716 560-4716 560-4717 560-4717
		560-4718 560-4719 560-4721 560-4721 560-4722 560-4725 560-4725 560-4725 560-4725
ECCDIS =	104470	#9-232 69-2010 149-3656 171-4185 348-7384 351-7451 353-7512 355-7555 357-7661
ECCINI =	104472	#9-234 65-1795 #65-1829 #69-2025 131-3277 149-3676 171-4202 171-4209 171-4213
		333-7032 335-7071
ECC1DI =	104471	#9-233 135-3334 135-3351 146-3640 206-4714 206-4721 206-4774 209-4800 209-4852
		212-4891 214-4921 217-4964 217-5002 220-5041 220-5078 223-5119 225-5158 240-5409
		258-5689 260-5697 260-5707
ECC1IN =	104473	#9-235 135-3371 135-3378 258-5654 260-5700
EMTVEC =	000030	#11-325 *47-1459 *47-1460
EM1	103134	556-4656 #564-4752
EM10	103633	548-4453 #564-4762
EM11	103677	550-4460 550-4465 550-4475 550-4480 #564-4763
EM12	103721	550-4470 #564-4764
EM13	103745	550-4485 #564-4765
EM14	103777	550-4490 #564-4766
EM15	104043	550-4495 #564-4767
EM16	104111	550-4500 #564-4768
EM17	104137	550-4505 #564-4769
EM18	104177	550-4510 #564-4770
EM19	104256	552-4517 #564-4771
EM2	103220	548-4423 #564-4753
EM20	104333	552-4522 #564-4772
EM21	104415	552-4532 #564-4773
EM22	104454	552-4537 #564-4774
EM23	104501	552-4547 #564-4775
EM24	104530	548-4418 #564-4776
EM25	104607	552-4557 #564-4777
EM26	104634	552-4562 #564-4778
EM27	104705	554-4579 #564-4779
EM28	104774	554-4584 #564-4780

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
EM29		105027	554-4594 #564-4781
EM3		103256	548-4428 #564-4754
EM30		105111	554-4599 #564-4782
EM31		105230	554-4604 #564-4784
EM32		105330	554-4609 #564-4785
EM33		105435	554-4614 #564-4786
EM34		105543	554-4619 #564-4787
EM35		105624	554-4589 #564-4788
EM36		105711	554-4624 #564-4789
EM37		105760	556-4631 #564-4790
EM38		106025	552-4552 #564-4791
EM39		106133	556-4636 #564-4792
EM4		103311	548-4433 #564-4755
EM40		106220	556-4641 #564-4793
EM41		106272	556-4646 #564-4794
EM42		106374	556-4651 #564-4795
EM43		106477	556-4661 #564-4796
EM44		106575	556-4666 #564-4797
EM45		106672	556-4671 #564-4798
EM46		106773	556-4676 #564-4799
EM47		107062	556-4681 #564-4800
EM48		107170	558-4688 #564-4801
EM49		107267	558-4693 #564-4802
EM5		103357	548-4438 #564-4756
EM6		103434	548-4443 #564-4757
EM7		103461	548-4448 #564-4758
EM8		103521	552-4567 #564-4759
EM9		103556	554-4574 #564-4761
ENASBE	=	104506	#9-246 65-1827 69-2023
ENA1SB	=	104507	#9-247 258-5658 258-5691 260-5702
END		117144	45-1430 #568-4997 568-5007 568-5008
ENERGI	=	104420	#9-187 62-1686 107-2827 416-1215 418-1236
ERRMAX	=	002702	#42-1316 452-2047
ERROR	=	104000	#11-259 64-1770 65-1817 65-1821 71-2038 79-2184 89-2395 90-2448 90-2485
			92-2521 #94-2583 94-2619 98-2695 98-2704 102-2758 104-2775 209-4815 212-4902
			258-5663 260-5720 260-5731 281-6197 283-6224 283-6228 283-6231 283-6234 298-6472
			340-7193 342-7209 342-7223 365-28 385-409 385-417 389-485 389-511 389-531
			391-550 393-600 408-976 408-1002 408-1019 410-1052 412-1086 412-1099 416-1197
			420-1268 420-1290 423-1332 423-1354 425-1404 427-1445 429-1481 429-1495 435-1592
			435-1606 437-1639 437-1652 439-1704 441-1729 448-1994 #448-1996 448-2006 #448-2008
			450-2017 452-2067 452-2070 452-2073 452-2076 454-2087 454-2098 454-2104 #454-2106
			454-2115 477-2665 477-2671 483-2778
ERRPC		002024	#38-1128 *459-2228 *459-2229 459-2232 *459-2235 *459-2236 459-2241 462-2295 560-4700
			560-4701 560-4702 560-4703 560-4704 560-4705 560-4706 560-4709 560-4711 560-4713
			560-4715 560-4717 560-4718 560-4719 560-4720 560-4721 560-4722 560-4723 560-4724
			560-4725
ERRPSW		002034	#38-1132 *459-2231 *459-2238 477-2662
ERRSP		002030	#38-1130 *459-2230 *459-2237 477-2661
ERRVEC	=	000004	#11-318 *47-1469 *47-1470 456-2152 *456-2153 *456-2155 *456-2159
EVEN		002456	#38-1248 *248-5493 248-5494 *250-5557 250-5557
EXBANK		042422	71-2035 75-2112 77-2128 81-2209 81-2217 94-2565 94-2573 94-2580 102-2753
			109-2846 111-2862 113-2896 115-2930 117-2971 120-3015 122-3057 124-3106 126-3148

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	131-3239	131-3280	169-4135	169-4158	171-4188	171-4207	313-6653	317-6727	317-6731
			#324-6861	348-7362	348-7437	351-7478	353-7532	353-7546	355-7615	357-7681	357-7695
			389-464	412-1075	414-1168	414-1175	420-1250	423-1315	425-1375	427-1423	429-1465
			433-1530	437-1621	439-1682						
EXCMD3	044726		344-7255	344-7258	344-7262	#344-7264					
EXCMD4	045236		346-7298	346-7302	#346-7316						
EXIT	043522		#335-7053	456-2143	460-2265						
EXIT2	043526		#335-7055								
FASTCI =	177640		#11-378	67-1869	67-1899	83-2291	154-3824	182-4360	240-5410	290-6328	290-6333
			337-7103	385-399							
FATAL\$	002070		#38-1145	*65-1817	*65-1821	*258-5663	*281-6197	*283-6224	*283-6228	*283-6231	*365-28
			460-2263	462-2308							
FCMD10	047550		338-7154	#355-7618							
FCMD11	047734		338-7155	#357-7645							
FCMD12	050330		338-7156	#359-7700							
FCMD13	050356		338-7157	#359-7710							
FCMD14	050400		338-7158	#359-7715							
FCMD15	050420		338-7159	#359-7721							
FCMD16	050436		338-7160	#359-7727							
FCMD17	050454		338-7161	#359-7733							
FCMD18	050470		338-7162	#361-7739							
FCMD19	050502		338-7163	#361-7744							
FCMD20	050566		338-7164	#361-7763							
FIELDS	043774		#338-7124	487-2940	510-3699						
FIRST =	060000		#13-556	67-1862	67-1891	67-1917	69-1959	69-1970	69-1970	*69-2014	*69-2014
			102-2749	144-3499	144-3507	144-3509	144-3515	144-3523	144-3525	144-3531	144-3539
			144-3541	144-3547	144-3555	144-3557	144-3575	144-3583	144-3585	144-3591	144-3599
			144-3601	144-3607	144-3615	144-3617	152-3783	152-3791	152-3804	154-3815	154-3841
			154-3852	156-3871	164-4030	165-4074	167-4107	169-4139	169-4162	171-4192	173-4226
			173-4231	173-4232	175-4248	175-4264	175-4268	177-4285	177-4288	177-4289	177-4290
			177-4291	177-4299	228-5174	240-5407	248-5506	250-5527	250-5561	250-5563	250-5564
			270-5943	*335-7088	*335-7089	*335-7090	344-7245	346-7284	348-7382	351-7482	355-7569
			365-23	365-31	365-33	365-35	*365-60	367-68	367-82	367-83	367-86
			369-102	369-103	369-106	371-147	371-151	*371-159	373-166	373-170	373-179
			373-181	*373-191	375-233	375-234	375-238	375-246	377-264	377-265	377-281
			377-291	379-305	*379-306	379-310	379-313	379-319	*381-350	*381-352	383-372
			383-376	383-377	385-388	385-391	385-392	385-408	385-416	387-436	*389-469
			389-494	389-514	*391-558	403-800	403-801	403-802	*403-803	*403-804	403-805
			403-807	*405-837	*405-838	405-843	407-926	*407-928	*407-929	425-1381	425-1384
			468-2435								
FIRSTB	002260		#38-1175	64-1733	64-1734	64-1739	*64-1763	64-1764	137-3392	*139-3438	139-3439
			139-3441	*139-3441	144-3505	144-3521	144-3537	144-3553	144-3581	144-3597	144-3613
			*475-2595	475-2599							
FLIPLO	002734		#42-1329	*47-1445	154-3831	*270-5915	*270-5916	270-5917	270-5919	270-5921	
FLIPWA	035214		154-3814	#270-5913							
FLUSH	014656		#104-2784	414-1184							
FSCMD0	044170		#338-7144	#340-7169							
FSCMD1	044256		338-7145	#340-7187							
FSCMD2	044346		338-7146	#342-7202							
FSCMD3	044520		338-7147	#344-7231							
FSCMD4	044764		338-7148	#346-7270							
FSCMD5	045274		338-7149	#348-7322							

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
FSCMD6		046152	338-7150 #348-7440
FSCMD7		046160	338-7151 #351-7447
FSCMD8		046510	338-7152 #353-7496
FSCMD9		047104	338-7153 #355-7550
FSINFL		002542	#38-1274 *107-2797 144-3568 *144-3569
FSPAT		045752	348-7392 #348-7396
FSSTAC		002374	#38-1219 *340-7189 340-7197 *342-7204 342-7226 *348-7325 348-7429 *351-7468 351-7472 *353-7498 353-7541 *355-7552 355-7610 *357-7647 357-7690
FS1		044052	#338-7136 338-7142 338-7166
FS7FLA		002546	#38-1276 169-4152 *351-7449 *351-7490
GBLENG	=	000076	177-4285 177-4288 #269-5902
GDSWIT		002476	#38-1256 *79-2179 560-4708
GETDAT		055030	#395-614 448-1990 448-2002
GETDA1		055156	*395-616 395-626 #395-631
GETDIS		066514	182-4336 185-4370 456-2172 #457-2176 459-2217
GETSIZ		017736	64-1727 137-3391 #139-3421
GOOD		002050	#38-1137 *64-1769 *79-2175 *98-2702 *385-406 *385-414 *393-576 *393-578 *393-592 *395-605 *445-1886 *445-1892 *445-1897 *445-1904 *445-1907 *445-1914 *445-1919 *445-1924 *445-1929 *445-1934 *447-1940 *447-1945 *447-1950 *447-1955 *447-1960 *447-1965 *447-1970 *447-1975 *447-1980 *447-1985 450-2030 *452-2056 *452-2058 *454-2085 *454-2095 *454-2114 560-4700 560-4706 560-4708 560-4715 560-4720 560-4721
GOOD2		002052	#38-1138 *450-2020 *450-2024 560-4715
GOOD3		002054	#38-1139 *450-2021 *450-2025 560-4715
GTSWR	=	104407	#9-174 62-1667
HEADER		002724	#42-1325 *47-1446 *102-2734 *102-2736 *109-2848 *109-2851 *129-3197 *129-3210 *131-3252 *206-4741 *206-4744 *206-4755 *206-4758 *209-4813 *209-4816 *209-4827 *209-4830 *212-4900 *212-4903 *260-5721 *260-5732 *348-7380 *353-7529 *357-7678 *452-2061 *452-2078 *454-2086 *454-2088 *454-2097 *454-2099 *460-2275 462-2306 462-2318 *464-2357 477-2651 *477-2652 477-2667 *477-2668 *477-2672
HIACBA		002514	#38-1263 *77-2124 *77-2131 77-2135 77-2143
HIADRS	=	177742	#11-336 560-4704 560-4714
HIBANK		002512	#38-1262 *75-2120
HIPAT		042762	124-3102 126-3144 #327-6937
HT	=	000011	#11-269 407-887
HUNGMS		002572	#38-1286 *90-2441 *90-2445 *90-2477 *90-2482 *410-1044 *410-1048
HUNGTI		002574	#38-1287 *89-2391 *90-2442 *90-2478 *389-481 *389-527 *408-972 *408-999 *408-1015 *410-1045 *412-1081 *416-1193 *420-1264 *420-1286 *423-1328 *423-1350 *425-1400 *427-1441 *429-1477 *435-1602 *437-1635 *439-1700 *441-1725 *483-2785
I		002602	#38-1290 *54-1568 *54-1574 54-1574 *96-2650 *96-2650 96-2651 *96-2651 96-2653 *131-3250 131-3254 *131-3256 131-3256 *169-4133 169-4143 169-4146 *169-4151 169-4151 *169-4156 169-4166 169-4169 *169-4174 169-4174 *379-311 *379-318 379-318 *389-495 *389-513 389-513 *433-1537 433-1539 *433-1568 433-1568 435-1581 435-1589 435-1591 #42-1330 56-1603 56-1604 83-2274 83-2275 83-2295 89-2392 89-2403 90-2418 365-18 407-915 408-994 408-1000 416-1189 416-1190 416-1194 416-1209 #42-1331 56-1608 83-2296 89-2399 89-2404 89-2405 90-2419 90-2420 408-995 408-996 408-1007 416-1191 416-1192 416-1202 416-1210
IISTAC		002736	83-2293 #416-1187 416-1208
IISTAD		002740	89-2402 408-993 #416-1207
IISTIN		061074	#42-1332 89-2386 90-2464 379-303 379-316 381-341 381-342 381-343 381-343
IISTOF		061174	381-346 381-347 381-347 383-358 391-563 408-985
IISTVE		002742	
IITRAP		036764	47-1473 90-2464 #283-6233
INCBNK		042772	111-2879 113-2913 115-2949 117-2990 120-3029 122-3073 124-3120 126-3164 #327-6941

CEMKAB SYMBOL	CREATED BY	MACRO	ON	1-AUG-79	AT	07:46	PAGE 13	D 13	SEQ 0573			
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES				CREF	V01				
KIPAR6	=	172354	#11-465									
KIPAR7	=	172356	#11-466									
KIPDRO	=	172300	#11-439	310-6632	321-6819	397-661	400-725	400-746	405-823	418-1227		
K10		002550	#38-1277	81-2233	418-1224							
K11		002552	#38-1278									
K12		002554	#38-1279									
K13		002556	#38-1280									
K14		002560	#38-1281									
K15		002562	#38-1282									
K16		002564	#38-1283									
K17		002566	#38-1284									
KMAP	=	104422	#9-190	62-1683								
KPFLAG		002122	#38-1158	*69-1958	71-2037	81-2211	293-6356	293-6362	293-6378	293-6384	293-6391	
				293-6398	300-6490	300-6497	*324-6877	*324-6885				
KSTACK		002706	#42-1318	45-1406	56-1611	*89-2378	89-2379	*90-2471	*90-2507	107-2825	*367-71	
				367-73	*408-962	*408-978	*408-990	*408-1004	*408-1021	*408-1028	*412-1105	416-1213
				481-2707								
LAST	=	157776	#13-557	131-3261	152-3800	162-3980	164-4028	165-4075	175-4249	175-4263	228-5176	
				236-5289	248-5507	250-5529	250-5565	265-5805	269-5893	344-7247	346-7286	355-7571
				375-244	387-437	425-1384						
LASTB		002262	#38-1176	64-1738	64-1764	137-3393	*139-3431	*139-3434	*139-3435	*139-3439	*139-3440	
				139-3442	*139-3442	*475-2596	475-2599					
LASTER		002020	#38-1126	*79-2185	104-2774	*107-2798						
LF	=	000012	#11-270	407-941								
LKS	=	177546	#11-266	50-1510	*373-177	*373-184						
LOADBA		002532	#38-1270	*313-6656	317-6721	317-6726						
LOADCS	=	104425	#9-195	77-2148	77-2151	92-2554	139-3426	293-6364	293-6374	293-6386	293-6400	
				300-6485	300-6499	302-6510	342-7218	400-740				
LOADHO		002710	#42-1319	*81-2219	81-2222	81-2224	107-2828	310-6641	313-6657	317-6722	317-6730	
				335-7078	414-1174							
LOADRS	=	177740	#11-335	281-6176	560-4704	560-4714						
LOCK		002670	#40-1307	*45-1436	45-1437	*56-1614	*58-1620	58-1621	*77-2157	*83-2287	*83-2299	
				*89-2368	*90-2501	90-2502	*335-7082	*408-961	*408-989	*410-1036	*412-1068	*414-1183
LOOP		014342	#99-2710	107-2840								
LOWMAP		042026	62-1678	315-6709	317-6749	#317-6754	400-722					
MAINT	=	177750	#11-339	*62-1672	*102-2768	*129-3196	*169-4127	*171-4182	*331-7012	*333-7034	*335-7072	
				*348-7378	*353-7527	*355-7600	*357-7676	*369-131	*373-193	*375-253	*383-366	*395-620
				*395-628	397-644	*401-780	457-2182	489-3023	560-4714			
MAPHO	=	170202	#13-484	*62-1677	*315-6708	*317-6748	398-698	*400-721				
MAPH36	=	170372	#13-544	*62-1675	*75-2084	*75-2091	75-2092	*75-2097	*75-2101	75-2102	*75-2105	
				398-698	*400-721							
MAPH37	=	170376	#13-546									
MAPLO	=	170200	#13-483	*62-1677	*315-6706	*317-6748	317-6756	398-698	*400-721			
MAPL1	=	170204	#13-485	317-6757								
MAPL36	=	170370	#13-543	*62-1676	*75-2085	*75-2090	*75-2096	*75-2100	*75-2106	398-698	*400-721	
MAPPER		040700	62-1685	67-1858	69-1967	69-2019	131-3241	173-4224	175-4244	177-4282	182-4335	
				185-4369	#310-6604	335-7086	344-7250	346-7289	348-7360	348-7436	353-7545	355-7587
				355-7614	357-7694	365-11	365-58	367-75	371-156	373-168	373-188	379-308
				381-349	381-351	383-374	385-390	387-435	389-467	389-477	403-798	405-835
				407-924	407-930							
MARGIN		002112	#38-1154	*111-2869	*115-2939	*120-3022	*122-3066	129-3194	*169-4127	*171-4182	329-6955	
				*329-6956	329-6959	*329-6960	329-6961	*329-6963	329-6964	*329-6967	*329-6970	329-6972

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	329-6979	*329-6980	329-6981	*329-6983	329-6985	331-6995	*331-6997	*331-6999	*331-7003
			331-7008	331-7009	331-7009	331-7010	348-7323	348-7324	*348-7351	348-7352	348-7352
			348-7355	348-7375	*348-7433	351-7448	*351-7491	353-7497	*353-7502	353-7503	353-7503
			353-7506	353-7525	*353-7544	355-7551	*355-7575	355-7576	355-7576	355-7579	355-7598
			*355-7613	357-7646	*357-7651	357-7652	357-7652	357-7655	357-7674	*357-7693	*369-131
MASTER	= 000020		*373-193	*375-253	*383-367	468-2448					
			#13-561	45-1407	50-1490	58-1625	62-1665	62-1682	81-2205	89-2380	90-2495
			90-2496	90-2506	94-2563	94-2571	94-2574	94-2579	278-6096	310-6640	324-6886
			365-4	379-309	389-465	397-638	410-1051	410-1060	412-1085	414-1112	416-1195
			420-1251	423-1316	425-1376	427-1424	429-1466	433-1531	437-1622	439-1683	456-2135
			487-2931								
MAST01	057372		104-2783	#408-949							
MAST02	060046		#410-1032								
MAST03	060220		#412-1064								
MAST04	061322		414-1118	#420-1241							
MAST05	061706		414-1123	#423-1306							
MAST06	062270		414-1128	#425-1369							
MAST07	062550		414-1133	#427-1417							
MAST10	062776		414-1138	#429-1458							
MAST11	063260		414-1143	#431-1506							
MAST12	063732		414-1148	#437-1615							
MAST13	064200		414-1153	#437-1662							
MAST14	064204		414-1158	#439-1666							
MAST15	064472		414-1163	#441-1717							
MASWR	002674		#40-1309	*89-2370	89-2372	90-2494	*408-956	408-958	412-1107		
MDISPL	002676		#40-1310	365-33	*365-33	365-34	*375-222	*389-460	389-461	*408-953	408-954
			*410-1034	*412-1066	412-1067	*414-1179	*420-1243	420-1244	*423-1308	423-1309	*425-1371
			425-1372	*427-1419	427-1420	*429-1461	429-1462	*431-1523	431-1524	*437-1617	437-1618
			*439-1674	439-1675	*441-1719	441-1720	457-2191	457-2191	489-3010	489-3010	489-3012
MEMDON	014410		99-2715	#102-2729							
MEMERR	= 177744		#11-337	281-6170	281-6177	*281-6177	281-6184	*281-6184	281-6190	*281-6190	283-6214
			*283-6214	283-6218	*283-6218	560-4704	560-4709	560-4714			
MHALT1	004532		#45-1438	45-1439							
MHALT2	005770		#58-1622	58-1623							
MHALT3	012760		#90-2503	90-2504							
MINDEX	002666		#40-1306								
MJPAT	021600		54-1560	151-3730	151-3730	151-3733	#151-3741				
MJSIZE	002466		#38-1252	*96-2627	*96-2631	*96-2637	*96-2638	*96-2639	*96-2640	96-2650	96-2655
			98-2694	560-4705							
MJTEST	021512		129-3208	#151-3724							
MKCONT	016664		129-3205	#131-3214	383-362						
MKCSRS	002334		#38-1201	*64-1707	*64-1710	*64-1715	64-1746	92-2548	131-3232	162-3956	296-6407
			298-6442	302-6505	363-7776	398-679	400-733	475-2594	479-2685		
MKCSRT	020124		54-1550	#142-3454							
MKFLAG	002126		#38-1160	71-2036	75-2113	81-2211	94-2569	94-2590	129-3200	171-4189	*324-6877
			*324-6897	327-6924	329-6964	329-6967	331-6996	331-7007	466-2412	470-2473	475-2582
MKPAT	021402		54-1555	149-3664	149-3664	149-3667	#149-3684				
MKSIZE	002470		#38-1253	*96-2628	*96-2633	*96-2641	*96-2642	*96-2643	*96-2644	96-2650	96-2657
			98-2694	560-4705							
MKTEST	021222		109-2849	129-3206	#149-3653						
MK11AD	= 172100		#13-553	60-1638	288-6276	290-6305	290-6324				
MMRO	= 177572		#11-352	*285-6254	*285-6258	*315-6711	*315-6713	*317-6743	*317-6745	*375-221	398-691

SYMBOL	VALUE	REFERENCES
MMR1	= 177574	*400-731 403-814 *405-831 560-4704 560-4714
MMR2	= 177576	#11-353 398-691 *400-731 403-813 *405-830 560-4704 560-4714
MMR3	= 172516	#11-354 398-691 *400-731 403-812 *405-829 560-4704 560-4714
		#11-355 *62-1671 *62-1679 *107-2826 *315-6702 *315-6710 *317-6747 *317-6750 *333-7038
		*335-7081 398-691 *400-731 403-811 *405-828 560-4704 560-4714
MMTRAP	036740	47-1471 #283-6227
MMVEC	= 000250	#11-333 *47-1471 *47-1472
MP1	012070	83-2284 83-2294 #87-2319
MP2	012224	#89-2392 89-2400 90-2437
MP3	012716	90-2466 90-2487 #90-2494
MP4	012552	89-2373 #90-2467
MP5	012764	#90-2506
MRUNSE	053640	#387-434 437-1624
MSEEDH	002720	#42-1323 *47-1447 47-1449
MSEEDL	002722	#42-1324 *47-1448 47-1450
MSG34	113625	*131-3267 #568-4898
MSG87	115753	*92-2536 *92-2540 #568-4957
MSG98	116242	*483-2763 #568-4970
MSG34	113642	*131-3268 #568-4900
MSG000	111176	56-1582 #568-4839
MSG001	111254	273-5962 #568-4840
MSG002	111336	273-5963 #568-4841
MSG003	111413	273-5964 #568-4842
MSG004	111520	273-5970 273-5988 #568-4844
MSG005	111626	276-6015 #568-4846
MSG006	111640	497-3467 #568-4847
MSG007	111675	278-6049 #568-4848
MSG008	111707	276-6029 #568-4849
MSG009	111721	278-6089 #568-4850
MSG010	111733	278-6135 #568-4851
MSG011	111745	273-5974 273-5987 #568-4852
MSG012	112033	273-5977 273-5989 #568-4853
MSG013	112130	276-6018 #568-4854
MSG014	112132	273-5986 276-6020 278-6145 459-2222 466-2388 468-2456 #568-4855
MSG015	112134	*133-3297 133-3298 *276-6038 276-6039 *278-6056 *278-6077 278-6078 *278-6094 *278-6097
		*278-6099 278-6102 *278-6116 *278-6120 *278-6125 278-6126 *278-6138 *278-6142 278-6143
		*472-2510 *472-2512 472-2514 *472-2530 *472-2532 472-2534 #568-4856
MSG016	112136	278-6112 #568-4857
MSG017	112150	273-5973 273-5976 #568-4858
MSG018	112161	464-2352 468-2454 472-2505 472-2523 479-2692 479-2694 481-2712 481-2726 481-2742
		#568-4859
MSG019	112164	#568-4860
MSG020	112170	338-7126 #568-4861
MSG021	112224	338-7141 #568-4862
MSG022	113252	363-7782 #568-4885
MSG023	113274	342-7211 #568-4886
MSG024	113316	342-7214 #568-4887
MSG025	113340	340-7196 342-7225 #568-4888
MSG026	113371	338-7136 #568-4889
MSG027	113403	342-7207 #568-4890
MSG028	113420	342-7221 #568-4891
MSG029	113434	344-7234 #568-4892

SYMBOL	VALUE	REFERENCES					
MSG030	113454	348-7327	361-7749	#568-4893			
MSG031	113473	344-7235	346-7274	355-7559	#568-4894		
MSG032	113533	344-7257	346-7297	#568-4895			
MSG033	113552	344-7261	346-7301	#568-4896			
MSG034	113571	131-3269	#568-4897				
MSG035	113707	107-2803	#568-4902				
MSG036	113712	346-7273	#568-4903				
MSG037	113731	346-7304	#568-4904				
MSG038	113750	346-7313	#568-4905				
MSG039	113766	346-7306	#568-4906				
MSG040	114010	348-7326	#568-4907				
MSG041	114055	348-7335	#568-4908				
MSG042	114102	348-7339	#568-4909				
MSG043	114123	348-7344	#568-4910				
MSG044	114150	348-7349	353-7500	355-7573	357-7649	#568-4911	
MSG045	114175	348-7356	353-7507	355-7580	357-7656	#568-4912	
MSG046	114231	348-7368	351-7462	353-7518	355-7591	357-7667	#568-4913
MSG047	114264	351-7489	#568-4914				
MSG048	114303	338-7129	#568-4915				
MSG049	114343	348-7364	#568-4916				
MSG050	114375	351-7455	#568-4917				
MSG051	114503	401-778	#568-4919				
MSG052	114523	351-7459	#568-4920				
MSG053	114554	351-7461	#568-4921				
MSG054	114572	351-7476	#568-4922				
MSG055	114642	353-7499	#568-4923				
MSG056	114663	353-7516	357-7665	#568-4924			
MSG057	114716	475-2585	*475-2621	475-2623	*475-2624	#568-4925	
MSG058	114722	479-2683	#568-4926				
MSG059	114766	355-7553	#568-4927				
MSG060	115016	355-7589	#568-4928				
MSG061	115060	468-2428	#568-4929				
MSG062	115067	491-3191	491-3250	#568-4930			
MSG063	115107	491-3192	#568-4931				
MSG064	115120	491-3193	491-3252	#568-4932			
MSG065	115130	491-3251	#568-4933				
MSG066	115142	460-2268	#568-4934				
MSG067	115225	462-2310	#568-4935				
MSG068	115234	466-2414	#568-4936				
MSG069	115241	466-2416	#568-4937				
MSG070	115246	96-2654	#568-4938				
MSG071	115271	96-2656	#568-4939				
MSG072	115304	96-2658	#568-4940				
MSG073	115317	357-7648	#568-4941				
MSG074	115336	*470-2472	*470-2493	470-2497	*474-2544	*474-2558	474-2560 #568-4943
MSG075	115341	313-6667	313-6688	#568-4944			
MSG076	115373	355-7630	#568-4945				
MSG077	115414	107-2801	#568-4946				
MSG078	115430	133-3296	#568-4947				
MSG079	115474	355-7621	#568-4948				
MSG080	115520	56-1592	#568-4949				
MSG081	115560	90-2474	#568-4950				

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES							
MSG082		115577	90-2476	#568-4951						
MSG084		115617	92-2522	#568-4952						
MSG085		115661	359-7702	#568-4953						
MSG086		115706	92-2534	#568-4954						
MSG087		115745	92-2541	#568-4956						
MSG088		115760	481-2706	#568-4959						
MSG089		115776	481-2721	#568-4960						
MSG090		116020	481-2737	#568-4961						
MSG091		116034	481-2730	481-2746	#568-4962					
MSG092		116046	489-3018	#568-4963						
MSG093		116062	489-3020	#568-4964						
MSG094		116070	489-3022	#568-4965						
MSG095		116077	489-3027	#568-4966						
MSG096		116106	#568-4967							
MSG097		116137	144-3570	#568-4968						
MSG098		116234	483-2764	#568-4969						
MSG099		116245	483-2774	#568-4972						
MSG100		116256	489-3014	#568-4973						
MSG101		116305	359-7711	#568-4974						
MSG102		116335	359-7716	#568-4975						
MSG103		116364	340-7170	#568-4976						
MSG104		116421	464-2362	#568-4977						
MSG105		116423	361-7745	#568-4978						
MSG106		116527	359-7722	#568-4979						
MSG107		116545	359-7728	#568-4980						
MSG108		116622	359-7734	#568-4981						
MSG109		116664	361-7740	#568-4982						
MSG110		116737	361-7759	#568-4983						
MSG111		117003	361-7764	#568-4984						
MSG112		117035	456-2142	#568-4985						
MSG113		117103	474-2568	#568-4986						
MSG114		117113	474-2571	#568-4987						
MSG115		117123	474-2574	#568-4988						
MSG116		117133	474-2577	#568-4989						
MST11A		063602	433-1567	#435-1575						
MTA030		024760	#171-4181	171-4208						
MTEST		016564	111-2871	113-2905	115-2941	117-2982	120-3024	122-3068	124-3115	126-3159 #129-3193
MTPA03		026506	154-3822	#191-4451						
MTPA04		026650	154-3843	#195-4513						
MTPA20		032572	162-3955	#236-5279						
MTPA21		033234	164-4031	#243-5416						
MTPA24		034060	165-4077	#253-5582						
MTPA25		034464	258-5653	258-5688	260-5696	#260-5707				
MTPA26		034576	167-4110	#263-5738						
MTPA32		034772	#267-5816	377-292						
MTPB03		026550	154-3826	#191-4476						
MTPB04		026704	154-3844	154-3855	#195-4526					
MTPB21		033264	164-4034	#243-5433						
MTPB24		034120	165-4078	#255-5626						
MTPB25		034504	258-5690	260-5698	#260-5714					
MTPB26		034612	167-4115	#263-5745						
MTPB32		035020	#267-5826	377-295						

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES							
MTPC03	026610	154-3827	#193-4494						
MTPC21	033320	164-4038	#243-5450						
MTPC24	034134	165-4079	#255-5634						
MTPC25	034536	258-5692	260-5701	260-5703	#260-5725				
MTPC26	034646	167-4111	#263-5763						
MTPD03	026630	154-3828	#193-4503						
MTPD21	033354	164-4041	#245-5468						
MTPD25	034412	258-5665	258-5667	258-5670	258-5672	#258-5688			
MTPD26	034666	167-4112	#263-5776						
MTPE25	034434	258-5676	258-5678	258-5681	258-5683	#260-5696			
MTP000	026376	67-1846	152-3786	169-4129	177-4306	#188-4413	188-4415	270-5945	369-133
MTP001	026422	152-3795	#188-4424	375-237					
MTP002	026454	152-3806	#188-4436	375-243					
MTP005	026724	154-3854	#195-4535						
MTP006	026760	156-3864	#198-4550						
MTP007	027150	156-3874	#201-4594						
MTP010	027250	156-3881	#203-4634						
MTP011	027356	158-3892	#206-4671						
MTP012	027766	158-3901	#209-4782						
MTP013	030344	158-3909	#212-4860						
MTP014	030716	158-3919	#217-4929						
MTP015	031312	160-3930	#220-5010						
MTP016	031700	160-3939	#223-5087						
MTP017	032302	160-3944	#228-5167						
MTP020	032360	#231-5201							
MTP022	033404	164-4050	164-4058	#248-5487					
MTP025	034152	165-4095	#258-5645						
MTP030	034704	171-4184	240-5406	#265-5784					
MTP031	034714	173-4226	#265-5790	265-5812					
MTP033	035054	175-4258	#267-5839						
MTP034	035106	177-4285	#269-5859	269-5902					
MTP20A	032360	162-3982	#231-5206	231-5210					
MTP20B	032376	162-3987	#231-5214	231-5223					
MTP20C	032430	162-3993	#231-5227	231-5236					
MTP20D	032462	162-4000	#231-5240	231-5249					
MTP20E	032514	162-4006	#233-5254	233-5263					
MTP20F	032546	162-4012	#233-5267	233-5274					
MTV020	023320	*162-3957	162-3958	*162-3971	162-3971	#162-3973			
MT0000	021704	#152-3781	348-7396						
MT0001	021740	149-3688	151-3745	#152-3789	348-7397				
MT0002	022000	149-3689	151-3746	#152-3798	348-7398				
MT0003	022050	151-3747	#154-3811	348-7399					
MT0004	022216	149-3690	151-3748	#154-3836	348-7400				
MT0005	022300	149-3691	151-3749	#154-3847	348-7401				
MT0006	022360	142-3454	#156-3860	348-7402					
MT0007	022414	149-3687	151-3744	#156-3867	348-7403				
MT0010	022456	142-3455	#156-3877	348-7404					
MT0011	022512	142-3457	#158-3886	348-7405					
MT0012	022570	142-3458	#158-3895	348-7406					
MT0013	022646	142-3459	#158-3904	348-7407					
MT0014	022722	142-3460	#158-3913	348-7408					
MT0015	023000	142-3461	#160-3924	348-7409					

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES												
MT0016	023056	142-3462	#160-3933	348-7410										
MT0017	023134	149-3686	151-3743	#160-3942	348-7411									
MT0020	023156	149-3693	#162-3949	348-7412										
MT0021	023612	149-3692	151-3750	#164-4021	348-7413									
MT0022	023734	149-3694	#164-4046	348-7414	357-7684									
MT0023	023766	151-3751	#164-4054	348-7415										
MT0024	024032	149-3696	151-3753	#165-4064	348-7416									
MT0025	024172	142-3456	#165-4090	348-7417										
MT0026	024240	149-3695	151-3752	#167-4100	348-7418									
MT0027	024370	102-2735	#169-4123	348-7419	351-7457									
MT0030	024754	104-2785	#171-4179	333-7036	335-7075	348-7420								
MT0031	025214	149-3697	151-3754	#173-4218	348-7421	353-7535								
MT0033	025340	149-3698	151-3755	#175-4238	348-7423									
MT0034	025530	149-3699	151-3756	#177-4276	348-7424									
MT0035	025652	149-3664	149-3685	149-3700	151-3730	151-3742	151-3757	#177-4297						
MT020X	023604	162-3991	162-3997	162-4004	162-4010	#162-4017								
MT020Y	023472	162-3985	#162-3999											
MT020Z	023322	162-3968	#162-3975											
MT0999	025772	54-1571	142-3463	142-3464	142-3465	142-3466	142-3467	142-3468	142-3469	142-3470				
		142-3471	142-3472	142-3473	142-3474	142-3475	142-3476	142-3477	142-3478	142-3479				
		142-3480	142-3481	142-3482	142-3483	142-3484	142-3485	149-3664	149-3702	149-3703				
		149-3704	149-3705	149-3706	149-3707	149-3708	149-3709	149-3710	149-3711	149-3712				
		149-3713	149-3714	149-3715	149-3716	149-3717	149-3718	149-3719	149-3720	149-3721				
		151-3730	151-3759	151-3760	151-3761	151-3762	151-3763	151-3764	151-3765	151-3766				
		151-3767	151-3768	151-3769	151-3770	151-3771	151-3772	151-3773	151-3774	151-3775				
		151-3776	#179-4320	348-7422	348-7425	348-7426	348-7427							
MULHLT	056150	#403-791	403-792											
MULTIE	060746	408-979	408-1005	408-1022	410-1054	412-1088	412-1102	414-1121	414-1126	414-1131				
		414-1136	414-1141	414-1146	414-1151	414-1156	414-1161	#414-1167						
MULTIO	002004	#38-1117	149-3654	151-3725	*359-7735	*361-7741								
MULTIP	003014	#42-1358	45-1407	*56-1610	83-2256	*90-2463	*90-2486	*92-2524	92-2533	104-2783				
		400-711	407-911	407-920	*408-977	*408-1003	*408-1020	*410-1053	*412-1087	*412-1101				
		*414-1120	*414-1125	*414-1130	*414-1135	*414-1140	*414-1145	*414-1150	*414-1155	*414-1160				
		*414-1165	*416-1198											
MULTIU	056142	400-711	#403-788											
MUT	002116	#38-1156	*102-2734	*102-2737	*109-2848	*109-2850	*129-3198	*129-3209	*348-7380	*353-7529				
		*357-7678	464-2355	477-2651	*477-2653	477-2667	*477-2669	*477-2672						
M13BAL	054472	389-516	#391-541											
NEMCNT	002072	#38-1146	*65-1805	65-1818	*67-1857	67-1877	67-1889	67-1907	67-1931	*283-6210				
		283-6211	*283-6216	560-4703										
NEWBAN	002400	#38-1221	171-4206	*315-6697	315-6703	*317-6738	321-6825	321-6838						
NEWKER	042322	315-6712	317-6744	#321-6835										
NEWLOA	042370	81-2227	310-6642	313-6659	317-6724	#321-6848								
NOCLOC	002104	#38-1151	*50-1512	429-1460	431-1518									
NOECCF	002474	#38-1255	*133-3299	*133-3301	158-3887	158-3896	158-3914	160-3925	160-3934					
NOERRO	002530	#38-1269	*340-7192	*342-7208	*342-7222	459-2213	459-2233	460-2251	*460-2277	462-2304				
		462-2316	464-2359	*477-2664	*477-2670									
NOFSMO	002526	#38-1268	*169-4132	*169-4153	*169-4175	*171-4186	*171-4203	*171-4214	338-7128					
NOIIST	002756	*36-1106	#42-1338	56-1590	83-2271	83-2283	89-2373	365-15	407-912	408-959				
NONEM	002102	#38-1150	*65-1806	*67-1844	*67-1916	*67-1925	283-6208							
NONEXI	036646	65-1807	67-1845	281-6172	#283-6208									
NOPAR	002100	#38-1149	*45-1424	*45-1434	*65-1804	*67-1843	*67-1875	*75-2087	*75-2099	*81-2223				

CEMKAB
SYMBOL CROSS REFERENCE
SYMBOL VALUE

CREATED BY MACRO ON 1-AUG-79 AT 07:46

PAGE 20
CREF V01

K 13

SEQ 0580

REFERENCES

NOSCOPI	002540	*81-2228	*102-2750	*102-2762	*131-3251	*149-3660	*149-3677	*151-3726	*151-3734	*158-3910
		*162-3954	*171-4183	*173-4223	*175-4243	*177-4281	*258-5650	*258-5685	281-6168	281-6179
		281-6187	281-6192	344-7232	*344-7233	*344-7265	346-7271	*346-7272	*346-7317	*348-7388
		*348-7393	353-7497	*353-7544	355-7551	*355-7584	*355-7613	357-7646	*357-7693	
		#38-1273	*164-4022	*164-4043	*165-4067	*165-4088	*171-4186	*171-4203	*171-4214	*173-4221
		*173-4234	*175-4241	*175-4270	*177-4279	*177-4294	*367-77	*408-952	*414-1183	456-2145
		460-2256								
NOTAB	002464	#38-1251	278-6079	*466-2384	*466-2386					
NULLFL	002436	#38-1237	*179-4322	331-7007	*331-7020					
OLDCAC	002370	#38-1217	*288-6289	288-6297	*290-6313	290-6317	*290-6331	290-6334		
OLDMAR	002376	#38-1220	*329-6955	*329-6959	*329-6979	*331-6995	331-7008	331-7009	331-7009	*348-7324
ONCE	002732	#42-1328	56-1581	56-1591	*56-1613					
ONES	002726	#42-1326	45-1407	67-1895	69-1981	69-1982	94-2579	94-2580	94-2581	94-2582
		135-3354	135-3356	154-3839	248-5496	248-5498	270-5924	270-5929	331-7014	365-14
		365-27	416-1191	416-1192	416-1210	447-1945	477-2631	477-2633	477-2635	477-2638
		477-2640	477-2642							
O.ADR1	075706	#506-3633	522-3944	522-3949	522-3953	*522-3961	*522-3968	530-4089	*532-4105	532-4121
		536-4158	536-4159	536-4193						
O.BACK	077302	516-3831	#522-3929							
O.BD	101366	530-4096	530-4097	#546-4399						
O.BIAS	075604	#504-3613	514-3816							
O.BKP	= 000016	#501-3573	506-3634	506-3636	506-3638	522-3949	522-3953	522-3959	522-3966	526-4011
		528-4051	530-4065	530-4088	532-4104	536-4157	536-4196			
O.BKPT	077324	516-3833	#522-3937							
O.BK1	100060	508-3656	#530-4066							
O.BRK	100042	508-3678	#530-4063							
O.BW	075566	#504-3588	512-3752	*514-3783	*518-3858	*518-3861	518-3872	518-3881	520-3902	520-3909
		520-3912	*520-3913	*520-3916	522-3929	522-3932	524-3979	*524-3997	*524-3999	*532-4119
		538-4206	544-4323							
O.BYT	076772	516-3824	#518-3861							
O.CAD	075560	#504-3585	*512-3738	512-3754	*518-3871	518-3874	518-3877	518-3879	518-3885	*520-3907
		*520-3909	520-3910	520-3914	*522-3932	524-3985	544-4326	544-4328		
O.CADV	100634	518-3880	524-3989	524-3998	#538-4204	546-4348				
O.CLGT	= 000024	514-3808	#546-4379							
O.CLSE	101214	512-3751	520-3894	520-3904	522-3931	#544-4321				
O.CMFD	075600	#504-3604	*514-3786	514-3814	*518-3850	518-3864				
O.CR	101316	544-4330	544-4332	544-4333	#546-4355					
O.CRET	077140	516-3825	#520-3894							
O.CRLF	101246	520-3911	530-4095	536-4176	#544-4330					
O.CRLS	101254	514-3784	#544-4332							
O.CSR1	075574	#504-3601	*536-4167	536-4183						
O.CSR2	075575	#504-3602	*536-4168	536-4184						
O.CT	075730	#506-3635	*522-3962	*522-3970	*528-4047	*532-4107	*532-4109			
O.CTLC	076162	#508-3685	516-3839							
O.CTLE	076242	#510-3705	516-3841							
O.CTLF	076210	#510-3695	516-3840							
O.C1	100000	#528-4051	532-4108							
O.DCD	076520	512-3766	512-3768	#514-3783	520-3895	532-4123				
O.DCDA	077144	#520-3895	522-3956	522-3963	522-3967					
O.DCD1	076550	514-3775	#514-3790	518-3851						
O.DCD3	076530	#514-3785	518-3882	518-3889	524-4000					
O.DOT	075562	#504-3586	*518-3870	520-3907	*520-3910					

CEMKAB SYMBOL	CREATED BY CROSS REFERENCE VALUE	MACRO ON 1-AUG-79 AT 07:46	REFERENCES	PAGE 21 CREF	V01	L 13					SEQ 0581	
O.ERR	076510		512-3757	#514-3781	514-3809	518-3866	520-3897	524-4002				
O.ERR1	077572		522-3942	522-3950	522-3954	522-3960	524-3980	524-3984	#524-4002	526-4008	526-4010	
			526-4013	528-4039	528-4042	528-4044						
O.ERR2	077164		#520-3903	522-3930								
O.ERR3	077150		#520-3897	520-3903								
O.FIL	075576		#504-3603	*542-4292	*542-4295							
O.FTYP	100730		514-3782	518-3884	518-3886	518-3888	520-3921	524-3982	532-4112	532-4116	532-4118	
			538-4219	538-4224	#538-4229	538-4230	540-4269	542-4289	542-4294			
O.GET	101006		510-3716	514-3792	#540-4250	540-4251	540-4270	540-4274				
O.GO	077576		516-3827	#526-4007								
O.GOGO	077624		510-3709	#526-4015								
O.GO2	077676		526-4022	#526-4025	528-4059							
O.ID	101300		508-3663	#546-4350								
O.IDND	= 101315		508-3664	#546-4354								
O.LG	= 000020		510-3720	#546-4397								
O.LGCH	101321		514-3805	#546-4359	546-4379							
O.MIN	076502		#514-3774	516-3838								
O.MINS	075603		#504-3611	*514-3774	*514-3787	514-3811	*514-3817					
O.ODT	075774		487-2948	#508-3647	510-3700							
O.OFST	077466		516-3832	#524-3979								
O.OLD	077154		516-3830	#520-3901								
O.OP1	077160		516-3828	#520-3902								
O.OP2	077220		#520-3910	522-3933								
O.OP2A	077226		512-3739	#520-3911								
O.ORAB	076364		#512-3740	516-3835								
O.ORPC	076342		#512-3734	516-3829								
O.ORRB	076374		#512-3743	516-3836								
O.P	075573		#504-3598	*508-3654	*508-3676	*526-4011	528-4040	528-4051	*530-4065	*532-4106	532-4113	
			532-4120									
O.PRI	075704		#506-3629	508-3669	530-4070	530-4074						
O.PROC	077732		516-3834	#528-4038								
O.PR1	077764		528-4046	#528-4048								
O.RALL	077430		508-3679	522-3958	#522-3964							
O.RCSR	= 177560		#501-3579	536-4167	*536-4169	536-4179	536-4181	*536-4183	540-4250			
O.RDB	= 177562		#501-3578	540-4252								
O.REGT	076270		#510-3716	516-3826								
O.REM	100602		508-3668	530-4069	#536-4190							
O.RORA	101272		520-3915	532-4122	#546-4348							
O.RRST	101172		508-3649	#544-4302								
O.RSB	100450		526-4024	#536-4157								
O.RSR	100416		526-4025	#534-4143								
O.RSTT	100534		526-4018	528-4050	#536-4176							
O.S	075571		#504-3592	*508-3655	*508-3675	*512-3765	*512-3767	526-4021	528-4053	530-4084	536-4190	
O.SCAN	076554		510-3727	#514-3792	514-3803							
O.SCRN	075602		#504-3609	540-4260	540-4271							
O.SEMI	076742		516-3822	#518-3847								
O.SEQ	075570		#504-3591	*520-3901	520-3905	*520-3908						
O.SMFD	075601		#504-3606	512-3761	*514-3785	*518-3849	518-3867	526-4007	528-4038			
O.SNGL	076446		#512-3761	516-3837								
O.STM	= 000340		#501-3575	508-3677	526-4016	528-4048	528-4055					
O.SVR	100360		508-3661	508-3667	530-4066	#534-4129						
O.SVTT	100500		508-3659	530-4094	532-4110	#536-4167						

CEMKAB SYMBOL	CREATED BY	MACRO	ON	DATE	TIME	PAGE	22	M	13	SEQ	0582
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	CREF	V01						
O.T		075572	#504-3597	*526-4019	*528-4057	530-4067					
O.TBIT		077644	#526-4019	528-4052	528-4054	530-4068					
O.TBT	=	000020	#501-3576	526-4020	526-4023	526-4030	528-4058	530-4080			
O.TCL2		076442	#512-3757	512-3762							
O.TCSR	=	177564	#501-3581	536-4168	*536-4170	536-4171	536-4177	*536-4184	538-4229		
O.TDB	=	177566	#501-3580	*538-4243							
O.TL		101345	510-3717	510-3720	510-3728	#546-4381	546-4397				
O.TRTC		101370	522-3937	522-3965	536-4159	#546-4400					
O.TVEC	=	000014	#501-3574	508-3650	*508-3677	*508-3678					
O.TYPE		101126	508-3665	508-3688	510-3698	510-3707	530-4098	#542-4286	542-4291	542-4297	544-4334
O.TYP1		101004	#538-4244	542-4287							
O.UIN		075752	#506-3637	*522-3969	*536-4158	536-4193					
O.UPC		075700	#506-3627	*508-3653	*526-4015	526-4027	*530-4063	530-4083	*530-4087	*530-4100	
O.URO		075662	#506-3620	508-3662	510-3725						
O.USP		075676	#506-3626	508-3685	510-3695	*534-4130	534-4131	534-4150			
O.UST		075702	#506-3628	*526-4020	*526-4023	526-4026	526-4029	*528-4058	*530-4064	530-4072	*544-4302
O.WRD		076760	516-3823	#518-3858							
O.WRD1		077036	518-3869	#518-3872	520-3922						
O.WST		101202	508-3651	508-3674	526-4017	526-4031	528-4049	528-4056	530-4082	#544-4312	
O.XXX		075564	#504-3587	*534-4129	534-4137	*534-4144	534-4151				
PADDRE		002042	#38-1135	*281-6176	560-4702						
PAFBAF		015730	99-2720	#120-3010	120-3039	120-3047					
PAFBAW		016066	99-2721	#122-3052	122-3083	122-3088	122-3096				
PARBAF		016246	99-2722	#124-3101	124-3130	124-3138					
PARBAW		016404	99-2723	#126-3143	126-3174	126-3179	126-3187				
PARCNT		002074	#38-1147	*65-1803	65-1815	*67-1856	67-1887	67-1905	67-1927	*81-2229	*102-2755
PARITY		036454	102-2757	*281-6173	281-6174	560-4702					
PARTHE		002372	47-1465	#281-6168							
PARVEC	=	000114	#38-1218	*258-5652	*260-5714	*260-5725	*260-5728	281-6189	344-7232	*344-7248	*344-7265
PASFLG		002362	346-7271	*346-7287	*346-7317	*355-7585					
PATERR		002076	#11-331	*47-1465	*47-1466						
PATPLU		005472	#38-1213	*129-3199	*131-3253	*171-4180	171-4197	*171-4198	206-4698	206-4769	*206-4771
PATTER		002120	212-4872	214-4916	*214-4918	217-4944	217-4997	*217-4999	220-5021	220-5074	*220-5076
PCBUMP		002406	223-5100	225-5154	*225-5156	*348-7381					
PCONF I		035402	#38-1148	67-1848	*67-1855	67-1885	67-1903				
PCONF S		035662	54-1551	54-1553	54-1556	54-1558	54-1561	54-1563	#54-1567		
PCONF 1		035572	#38-1157	*109-2844	*111-2867	*113-2910	*115-2937	*117-2987	*120-3011	*122-3053	*124-3123
PCONF 2		035630	*126-3167	149-3662	151-3728	*327-6932	327-6933	*327-6938	348-7323	*348-7341	348-7342
PDP110		036752	348-7343	348-7390	*348-7435	351-7448	*351-7491	353-7497	*353-7544	357-7646	*357-7693
PENDBO		002330	#38-1224	*149-3661	*151-3727	*154-3813	*154-3838	*154-3849	*156-3862	*156-3879	*167-4102
PERA05		064732	281-6183	348-7323	*348-7389	*348-7433					
PERBNK		065564	96-2660	#273-5953	348-7441						
PERECC		065650	*273-5955	273-5992	#273-5998						
PERRAB		065402	273-5966	#273-5983							
PERRAW		065330	273-5956	273-5981	#273-5992						
			47-1467	#283-6230							
			#38-1197	*64-1708	*64-1735	77-2136	77-2137				
			#445-1908	445-1915							
			393-599	448-1989	448-2001	450-2016	#452-2037	454-2111			
			#452-2053	452-2065							
			445-1930	445-1935	#448-2000						
			393-580	395-612	445-1887	445-1893	445-1899	445-1905	445-1920	445-1925	447-1941

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES	447-1951	447-1956	447-1961	447-1966	447-1976	447-1981	447-1986	#448-1988
PERRA3	054652	447-1946	447-1951	447-1956	447-1961	447-1966	447-1976	447-1981	447-1986	#448-1988
PERRA7	065454	#393-582	450-2022	450-2026						
PERRO1	= 104427	447-1971	#450-2013							
PERRO2	= 104430	#9-198	188-4432	267-5854						
PERRO3	= 104431	#9-199	188-4420	188-4445	267-5848					
PERRO4	= 104432	#9-200	193-4498	193-4506						
PERR05	064726	#9-201	195-4522	195-4544						
PERR06	064754	#445-1907								
PERR07	= 104433	#445-1914								
PERR10	= 104434	#9-202	198-4560	198-4579						
PERR11	= 104435	#9-203	198-4565	198-4584						
PERR12	= 104436	#9-204	201-4602	203-4652	236-5309	238-5353	240-5396			
PERR13	= 104437	#9-205	201-4608	203-4659						
PERR14	= 104440	#9-206	201-4616							
PERR15	= 104441	#9-207	201-4622							
PERR16	= 104442	#9-208								
PERR17	= 104443	#9-209								
PERR20	= 104444	#9-210	243-5421	243-5443	243-5454	245-5478	250-5535	250-5539		
PERR21	= 104445	#9-211	243-5427	243-5437	243-5460	245-5472	250-5544	250-5548		
PERR22	= 104446	#9-212	238-5332	240-5378						
PERR23	= 104447	#9-213	228-5189	267-5834						
PERR24	= 104450	#9-214	253-5611	255-5640						
PERR25	= 104451	#9-215	253-5616							
PERR26	= 104452	#9-216	263-5751	263-5756						
PERR27	= 104453	#9-217	231-5219	231-5244	233-5271					
PERR30	= 104454	#9-218	231-5231	233-5259						
PERR31	= 104455	#9-219	265-5797							
PERR32	= 104456	#9-220	206-4725	206-4731	217-4975	217-4981	220-5055	220-5060	223-5131	223-5137
PERR33	= 104457	#9-221	206-4737	206-4751						
PERR34	= 104460	#9-222	209-4821							
PERR35	= 104461	#9-223	206-4743	206-4757	209-4829					
PERR36	= 104462	#9-224	269-5871	269-5878						
PERR37	= 104463	#9-225								
PERR40	= 104464	#9-226								
PERR41	= 104465	#9-227								
PERR42	= 104466	#9-228								
PERR43	= 104467	#9-229								
PERXOR	065540	#9-230								
PFLAG	002130	448-1992	448-2004	450-2015	#450-2028	452-2060	454-2096			
PHYADD	002044	#38-1161	109-2847	313-6654	*324-6879	*324-6900	389-466	412-1076	420-1252	423-1317
PORTCO	002646	425-1377	427-1425	429-1467	433-1534	437-1623	439-1684			
PORTDI	002606	#38-1136	*468-2434	*468-2435	*468-2436	*468-2437	*468-2439	468-2440		
PSW	= 177776	#40-1304	*373-172	373-191	*373-191	*373-198	*389-469	*389-469	*389-469	*389-469
		*389-504	391-546	391-547	429-1491	429-1492	435-1583	435-1584	560-4718	560-4718
		560-4718	560-4718	560-4725	560-4725	560-4725	560-4725			
		#40-1300	*56-1600	*56-1607	*83-2279	92-2520	92-2520	92-2535	92-2542	365-23
		407-920	560-4719	560-4719	560-4719	560-4719				
		#11-261	*60-1651	*60-1652	*60-1658	*67-1868	*67-1881	*67-1898	*67-1920	*69-1969
		*135-3332	*146-3630	*173-4225	*175-4251	*177-4284	*182-4357	*185-4391	*273-5959	*290-6345
		*290-6347	*315-6699	*317-6740	*319-6791	*319-6807	*335-7087	*344-7251	*346-7291	*346-7309
		*348-7372	*351-7466	*353-7522	*355-7595	*355-7602	*357-7671	*365-22	*365-30	*365-59
		*367-67	*367-81	*369-101	*371-146	*371-158	*373-165	*373-169	*373-190	*375-232

CEMKAB
SYMBOL CROSS REFERENCE
SYMBOL VALUE

MACRO ON 1-AUG-79 AT 07:46

PAGE 24
CREF V01

SEQ 0584

REFERENCES

		*377-263	*377-280	*379-304	*379-312	*379-333	*383-371	*383-375	*385-387	*385-398
		*387-441	*387-452	*389-468	*389-496	*393-571	*393-593	*395-608	*395-622	*398-670
		*398-674	*401-756	*401-760	*403-799	*405-836	*407-925	*481-2717	*481-2718	*481-2734
PTYBUF	002636	#40-1303	83-2288	*371-149	371-153	371-159	*371-159	*407-928	*408-966	*437-1628
		437-1649	437-1650	483-2765	483-2766					
PTYFLA	002626	#40-1302	407-926	*407-929	456-2136	456-2136	456-2136	456-2136	483-2759	*483-2767
		483-2770	*483-2775							
PWLOCK	002760	#42-1339	*397-642	*403-789						
PWRVEC	= 000024	#11-324	*47-1463	*47-1464	*52-1529	*397-646	*397-647	*398-704	*400-712	*401-777
		403-800	403-801	*403-803	*403-804	*405-837	*405-838			
QUE	072154	89-2393	90-2443	90-2480	389-482	389-528	408-973	408-1016	410-1046	412-1082
		420-1265	420-1287	423-1329	423-1351	425-1401	427-1442	429-1478	435-1603	437-1636
		439-1701	441-1726	456-2137	#483-2754					
QUE1	072222	#483-2765	483-2772							
QUICK	002536	#38-1272	*52-1528	335-7074						
QVFLAG	002440	#38-1238	*52-1528	*52-1532	107-2802	*107-2804	162-3984	206-4764	209-4846	214-4911
		217-4992	220-5069	225-5149	329-6952	331-6992				
RANODD	034626	*167-4114	#263-5753	*263-5757						
RDCHR	= 104411	#9-177	490-3085							
RDDEC	= 104414	#9-180	338-7137							
RDLIN	= 104412	#9-178	491-3162	491-3213						
RDOCT	= 104413	#9-179	56-1593	342-7212	342-7215	344-7236	346-7275	346-7307	348-7328	348-7340
		348-7345	348-7350	353-7501	355-7560	355-7574	357-7650	359-7703	361-7750	363-7783
READCS	= 104426	#9-196	64-1709	77-2152	133-3294	137-3395	139-3427	146-3636	236-5303	238-5325
		238-5347	240-5371	240-5390	258-5662	260-5719	260-5730	296-6413	296-6432	298-6448
		298-6467	340-7191	342-7206	342-7220	393-586	398-684	470-2477	479-2690	
READON	002504	#38-1259	*369-105	*369-116	369-139	*375-231	375-256	*375-258	*377-270	*377-276
REALPA	002366	#38-1216	*152-3782	*152-3790	*152-3799	*154-3812	*154-3837	*154-3848	*156-3861	*156-3868
		*156-3878	*158-3891	*158-3900	*158-3908	*158-3918	*160-3929	*160-3938	*160-3943	*162-3953
		*164-4023	*164-4049	*164-4057	*165-4068	*165-4094	*167-4101	*169-4126	*171-4181	*173-4222
		*175-4242	*177-4280	*177-4298	*179-4321	*371-150	452-2066	452-2069	452-2072	452-2075
		457-2178	466-2400	489-3028						
REFRES	033760	248-5522	#250-5559							
REFSUB	034030	250-5562	250-5566	#250-5570						
REGCOP	035204	152-3785	152-3794	#270-5907	369-136					
RELOCA	041130	111-2885	113-2919	115-2960	117-3001	120-3044	122-3093	124-3135	126-3184	171-4200
		#313-6649								
RELOC1	041456	#315-6697	367-70							
RESREG	= 104416	#9-183	154-3829	165-4084	167-4116	270-5948	319-6774	319-6811	338-7130	340-7184
		#369-140	377-296	418-1237	464-2358	495-3346				
RESTAR	002752	*36-1102	*36-1104	#42-1336	50-1483					
RESVEC	= 000010	#11-319	*47-1467	*47-1468						
RLFLAG	002134	#38-1163	111-2882	113-2916	115-2957	117-2998	120-3041	122-3090	124-3132	126-3181
		*315-6714	*317-6746	324-6901	333-7035	335-7073	338-7128	*367-76	397-659	457-2188
		483-2776	489-3017							
RRFLAG	002132	#38-1162	111-2865	113-2899	115-2935	117-2976	120-3020	122-3064	124-3111	126-3155
		131-3240	169-4136	169-4159	171-4190	*324-6879	*324-6885	*324-6900	*324-6903	*324-6911
		348-7363	351-7479	353-7533	357-7682					
SAVREG	= 104415	#9-182	154-3823	165-4076	167-4109	270-5942	319-6772	319-6775	338-7125	369-137
		377-293	418-1223	462-2289	495-3331					
SBECSR	002150	#38-1168	*149-3672	*149-3673	*236-5283	*236-5284	474-2545	474-2548		
SBEFLA	002022	#38-1127	104-2773	*107-2796	*149-3671	*236-5282				

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
SLPAT1	051536	367-90	369-110 #369-121 377-272
SLPAT2	051542	367-91	369-111 #369-123 377-273
SLPAT3	051550	367-92	369-112 #369-125 377-274
SLPAT4	051556	367-93	369-113 #369-127 377-275
SLPAT5	051564	367-94	369-114 #369-129
SOBK	002704	#42-1317	173-4228
SOBLEN	= 000056	173-4226	173-4231 #265-5812
SOFTPA	002744	#42-1333	*47-1453 *104-2780 177-4301
SOURCE	002402	#38-1222	*206-4704 *209-4792 *212-4881 *217-4953 *220-5030 *223-5108 *258-5648 305-6539
SPECIA	002432	#38-1235	131-3244 *137-3394 *144-3500 *144-3516 *144-3532 *144-3548 *144-3576 *144-3592
		*144-3608	162-3963
SSP	=%000006	#11-277	*60-1655 *182-4358 *185-4392 398-675 *401-757 481-2719
ST	= 177776	#501-3572	544-4302 *544-4312
STACK	= 002000	#11-255	11-256 36-1094 36-1108 42-1318 45-1428 52-1531 90-2507 107-2809
		367-71	408-978 408-1004 408-1021 408-1028 412-1105
START	004352	36-1103	36-1105 #45-1403 77-2158 89-2387 335-7059 403-793
START1	000300	36-1098	#36-1102 36-1107
START2	000310	36-1099	#36-1104
START3	000200	#36-1098	568-5012
START4	000322	36-1100	#36-1106
STOPCP	011720	#83-2282	375-224
STOPOK	002520	#38-1265	*81-2239 456-2140 *456-2141
STRIPE	002460	#38-1249	*248-5501 248-5505 250-5526 *250-5556 250-5556
STRTDI	002364	#38-1215	
SUBAAA	005532	54-1565	#56-1578
SUBAAB	005200	#52-1516	
SUBAAQ	007720	67-1876	#69-1937
SUBAAR	013160	83-2256	#92-2546
SUBAAS	007002	64-1776	#65-1795
SUBAAT	011334	79-2173	79-2186 #81-2201
SUBAAU	014602	102-2764	#104-2773
SUCCES	002424	#38-1232	*131-3237 *131-3258 131-3266 *355-7623 355-7629 *355-7631 *475-2589 475-2613
		475-2620	*477-2629 *477-2631 *477-2633 *477-2635 *477-2638 *477-2640 *477-2642
SUPDOA	002360	#38-1212	*154-3824 *156-3864 *156-3874 *156-3881 *158-3892 *158-3901 *158-3909 *158-3919
		*160-3930	*160-3939 *160-3944 *162-3955 *164-4050 *164-4058 *165-4080 *165-4095 *169-4131
		*173-4232	*175-4264 *177-4289 *177-4308 185-4394 *369-135 *371-151 *373-181 *375-252
SUPDO1	026042	102-2756	152-3787 152-3796 152-3807 162-3983 164-4032 167-4113 167-4117 169-4144
		169-4167	171-4193 177-4314 #182-4334 270-5947 369-138 375-255 377-294
SUPDO2	026056	154-3830	154-3845 154-3856 162-3988 162-3994 162-4001 162-4007 162-4013 164-4035
		164-4039	164-4042 175-4269 #182-4336 377-397
SUPDO3	026220	154-3825	156-3865 156-3875 156-3882 158-3893 158-3902 158-3911 158-3920 160-3931
		160-3940	160-3945 164-4051 164-4060 165-4096 169-4147 169-4170 177-4311 #185-4369
		351-7486	369-139 371-154 375-256
SUPDO4	026234	162-4017	165-4081 165-4087 173-4233 175-4265 177-4293 #185-4370 373-182 373-186
SUPDRO	002266	#38-1179	*182-4338 182-4346 *185-4372 185-4380
SUPDR1	002270	#38-1180	182-4339 185-4373
SUPDR2	002272	#38-1181	
SUPDR3	002274	#38-1182	
SUPDR4	002276	#38-1183	
SUPDR5	002300	#38-1184	
SUPDR6	002302	#38-1185	182-4349 185-4383
SUPLIM	057372	#407-947	568-4997 568-4998

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
SUPM13		053752	#389-458 437-1663
SUPSTK	=	000740	#11-257 60-1654 182-4358 185-4392 481-2722 481-2728
SWAPAT		002764	#42-1343 *47-1452
SWR		002766	#42-1346 *50-1496 50-1498 *50-1503 *52-1525 62-1666 65-1826 69-2022 89-2370
			*89-2371 89-2372 *90-2494 96-2659 99-2712 107-2802 149-3655 149-3668 162-3984
			162-3989 162-3995 162-4002 162-4008 206-4763 209-4845 214-4910 217-4991 220-5068
			225-5148 273-5966 313-6650 313-6666 313-6687 329-6952 331-6992 348-7355 348-7383
			351-7450 353-7506 353-7511 355-7554 355-7579 355-7603 357-7655 357-7660 398-700
			400-719 408-956 *408-957 408-958 *412-1107 456-2140 456-2148 456-2164 459-2219
			459-2242 459-2245 460-2252 460-2256 464-2359 487-2952 487-2960 487-2982
SWREG	=	000176	#13-550 50-1503 62-1666 89-2371 408-957 487-2952
SWO	=	000001	#11-297 65-1826 69-2022 149-3655 149-3668 162-3989 162-3995 162-4002 162-4008
			348-7383 351-7450 353-7511 355-7554 357-7660
SW1	=	000002	#11-296
SW10	=	002000	#11-287 459-2219
SW11	=	004000	#11-286 107-2802 162-3984 206-4763 209-4845 214-4910 217-4991 220-5068 225-5148
			329-6952 331-6992 348-7355 353-7506 355-7579 357-7655
SW12	=	010000	#11-285 313-6650
SW13	=	020000	#11-284 313-6666 313-6687 459-2242
SW14	=	040000	#11-283 456-2148
SW15	=	100000	#11-282
SW2	=	000004	#11-295
SW3	=	000010	#11-294
SW4	=	000020	#11-293 273-5966
SW5	=	000040	#11-292 459-2245
SW6	=	000100	#11-291 96-2659
SW7	=	000200	#11-290 464-2359
SW8	=	000400	#11-289 456-2140
SW9	=	001000	#11-288 456-2164 460-2256
SYSSIZ		011210	79-2178 79-2182 #79-2189
SYSTID	=	177764	#11-347 56-1600 83-2272 365-16 407-913
TAG2\$		007366	67-1851 #67-1873
TAG3\$		007412	67-1872 #67-1877
TAG4\$		026136	182-4347 #182-4349
TAG70\$		067636	464-2332 #466-2369
TAG71\$		067646	464-2333 #466-2375
TAG72\$		067656	464-2334 #466-2381
TAG73\$		067722	464-2335 #466-2394
TAG74\$		067730	464-2336 #466-2400
TAG75\$		067742	464-2337 #466-2406
TAG76\$		067754	464-2338 #466-2412
TAG77\$		067776	464-2339 #468-2423
TAG78\$		070000	464-2340 #468-2428
TAG79\$		070006	464-2341 #468-2434
TAG80\$		070066	464-2342 #468-2448
TAG81\$		070100	464-2343 #468-2454
TAG82\$		070122	464-2344 #470-2472
TAG83\$		070324	464-2345 #472-2504
TAG84\$		070410	464-2346 #472-2522
TAG85\$		070502	464-2347 #474-2543
TAG86\$		070612	464-2348 #474-2567
TAG9\$		007232	#67-1849 67-1879 67-1912 67-1934

SYMBOL	CROSS REFERENCE	REFERENCES								
TASK	002616	#40-1301	*83-2285	*83-2298	90-2453	90-2490	365-31	365-35	*365-60	367-82
		369-102	375-233	377-264	377-281	*379-306	*389-474	389-488	*389-522	389-534
		408-981	408-1024	*410-1043	410-1057	*412-1080	412-1090	*416-1217	*420-1258	420-1271
		*420-1277	420-1293	*423-1322	423-1335	*423-1341	423-1357	*425-1391	425-1407	*427-1433
		427-1448	*429-1471	429-1484	*433-1542	*433-1555	435-1609	*437-1629	437-1642	*439-1689
		*439-1691	439-1707	*441-1724	441-1732					
TASKDO	050726	83-2301	365-5	#365-10	389-476	416-1218	420-1261	420-1280	423-1325	423-1344
		425-1395	427-1436	429-1473	433-1544	433-1557	437-1631	439-1695		
TASKRE	050710	#365-4	365-62							
TASK00	051234	#365-39	#365-63							
TASK01	051236	365-40	#367-66							
TASK02	051330	365-41	#367-80							
TASK03	051430	365-42	#369-100							
TASK04	051664	365-43	#371-145							
TASK05	051774	365-44	#373-164							
TASK06	052300	365-45	#375-219							
TASK07	052326	365-46	#375-226							
TASK10	052336	365-47	#375-230							
TASK11	052536	365-48	#377-262							
TASK12	052644	365-49	#377-279							
TASK13	052726	365-50	#379-301							
TASK14	053276	365-51	#383-356							
TASK15	053310	365-52	#383-361							
TASK16	053316	365-53	#383-365							
TASK17	053334	365-54	#383-370							
TASK20	053422	365-55	#385-385							
TASK7A	052344	375-228	#375-232							
TBG4\$	026314	185-4381	#185-4383							
TCFIG1	036012	#278-6051	278-6081	466-2385						
TCONF I	035664	273-5971	273-5980	273-5990	#276-6015					
TEMP	002534	#38-1271	*98-2666	98-2683	*98-2684					
TESTAD	002506	#38-1260	*131-3246	*131-3261	131-3261	135-3331	156-3863	156-3880	206-4713	206-4775
		209-4798	209-4836	209-4853	212-4863	214-4922	217-4963	217-4969	217-5003	220-5040
		220-5047	220-5051	220-5079	223-5118	223-5127	225-5159	258-5651	452-2055	
TIME	002434	#38-1236	*351-7458	351-7460						
TIMEOU	036726	47-1469	50-1502	50-1513	56-1612	60-1646	64-1774	65-1823	67-1874	75-2098
		#283-6223	340-7194	340-7198	342-7210	342-7227	344-7266	346-7318	457-2195	
TKVEC	= 000060	#11-327	273-5954	273-5954	*273-5956	*273-5957	*273-5995	*273-5995	348-7323	348-7323
		*348-7369	*348-7370	*348-7431	*348-7431	351-7448	351-7448	*351-7463	*351-7464	*351-7474
		*351-7474	353-7497	353-7497	*353-7519	*353-7520	*353-7544	*353-7544	355-7551	355-7551
		*355-7592	*355-7593	*355-7613	*355-7613	357-7646	357-7646	*357-7668	*357-7669	*357-7693
		*357-7693								
TMFLAG	002140	#38-1165	*120-3035	*122-3079	*124-3126	*126-3170	327-6922			
TOOMAN	002472	#38-1254	*452-2049	459-2245	*460-2277					
TRAPVE	= 000034	#11-326	*47-1461	*47-1462						
TRT	= 000003	#501-3577	522-3969	546-4400						
TSK05A	052240	373-170	#373-201	373-215						
TSK13A	053100	379-302	#379-324							
TSK13B	053136	379-335	#381-338							
TSK20A	053602	385-386	#385-422							
TSTBAN	007606	67-1860	#67-1915							
TSTDAT	002344	#38-1209	*206-4696	*206-4697	*206-4700	*206-4701	206-4702	206-4703	206-4704	*206-4710

CEMKAB
SYMBOL CROSS REFERENCE
SYMBOL VALUE

CREATED BY MACRO ON 1-AUG-79 AT 07:46

PAGE 29
CREF V01

G 14

SEQ 0589

REFERENCES

	*206-4712	206-4715	206-4717	206-4723	206-4729	*209-4790	*209-4791	209-4792	*209-4795
	*209-4797	209-4801	209-4803	*212-4870	*212-4871	*212-4874	*212-4875	212-4881	*212-4884
	*212-4886	*212-4888	*212-4890	212-4892	212-4894	*217-4942	*217-4943	*217-4946	*217-4947
	217-4953	*217-4956	*217-4958	*217-4960	*217-4962	217-4965	217-4967	217-4973	217-4979
	*220-5019	*220-5020	*220-5023	*220-5024	220-5030	*220-5033	*220-5035	*220-5037	*220-5039
	220-5042	220-5045	220-5053	220-5058	*223-5098	*223-5099	*223-5102	*223-5103	223-5108
	*223-5111	*223-5113	*223-5115	*223-5117	223-5120	223-5122	223-5129	223-5135	*258-5646
	*258-5647	258-5648	*258-5664	*258-5666	*258-5668	*258-5669	*258-5671	*258-5674	*258-5675
	*258-5677	*258-5679	*258-5680	*258-5682	260-5708	260-5710	447-1950	447-1955	452-2056
	452-2058	560-4713	560-4713						
	290-6323	#290-6344							
TSTRD1	037434								
TSTREA	= 104510	#9-248	135-3348	135-3367	206-4739	206-4753	209-4811	209-4823	
TST1	006260	#64-1689							
TST2	007004	#65-1796							
TST3	007160	#67-1833							
TST4	010310	#71-2031							
TST5	014342	#99-2710							
TST6	014414	#102-2730							
TST7	014456	#102-2740							
TWOC SR	002600	#38-1289							
TYPDS	= 104405	*77-2141	*77-2149	288-6281	288-6292	*342-7217	*342-7219		
TYPEIT	= 104401	#9-171	96-2653	96-2655	96-2657	107-2806	351-7460	355-7620	355-7636
		#9-167	56-1582	56-1592	90-2474	90-2476	92-2522	92-2534	92-2541
		96-2654	96-2656	96-2658	107-2801	107-2803	131-3269	133-3296	133-3298
		273-5962	273-5963	273-5964	273-5970	273-5972	273-5973	273-5974	273-5975
		273-5977	273-5986	273-5987	273-5988	273-5989	276-6015	276-6018	276-6020
		276-6039	278-6049	278-6078	278-6089	278-6102	278-6112	278-6126	278-6135
		278-6145	313-6667	313-6688	338-7126	338-7129	338-7136	338-7141	340-7170
		342-7207	342-7211	342-7214	342-7221	342-7225	344-7234	344-7235	344-7257
		346-7273	346-7274	346-7297	346-7301	346-7304	346-7306	346-7313	348-7326
		348-7335	348-7339	348-7344	348-7349	348-7356	348-7364	348-7368	351-7455
		351-7461	351-7462	351-7476	351-7489	353-7499	353-7500	353-7507	353-7516
		353-7538	355-7553	355-7559	355-7573	355-7580	355-7589	355-7591	355-7621
		355-7637	357-7648	357-7649	357-7656	357-7665	357-7667	357-7687	359-7702
		359-7716	359-7722	359-7728	359-7734	361-7740	361-7745	361-7749	361-7759
		363-7782	401-778	407-893	456-2142	459-2221	459-2222	460-2268	462-2290
		462-2311	462-2313	462-2320	462-2322	464-2352	464-2362	466-2388	466-2414
		468-2428	468-2443	468-2454	468-2456	470-2497	472-2505	472-2514	472-2523
		474-2560	474-2568	474-2571	474-2574	474-2577	475-2585	475-2623	479-2683
		479-2692	479-2694	481-2706	481-2710	481-2712	481-2721	481-2724	481-2726
		481-2737	481-2740	481-2742	481-2746	481-2748	483-2764	483-2774	485-2849
		487-2958	487-2959	487-2961	487-2970	487-2975	487-2984	487-3002	489-3009
		489-3018	489-3020	489-3022	489-3027	490-3089	490-3098	490-3104	490-3109
		490-3118	490-3119	490-3121	490-3124	490-3128	491-3189	491-3191	491-3192
		491-3248	491-3250	491-3251	491-3252	497-3467			
TYPOC	= 104402	#9-168	462-2295	466-2369	479-2691	479-2693	481-2711	481-2713	481-2725
		481-2741	481-2743	487-2960					
TYPOS	= 104403	#9-169	90-2475	92-2535	92-2542	344-7254	346-7305	346-7314	355-7633
		466-2406	468-2448	468-2455	489-3015	489-3021	489-3026	489-3028	466-2400
T12A	031700	#223-5092	225-5157						
T12B	031722	#223-5096	225-5153						
UDPARO	= 177660	#11-389	255-5641						
UDPAR1	= 177662	#11-390	381-347	381-347					

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
UDPAR7	=	177676	#11-396
UDPDR7	=	177636	#11-375 154-3822
UIPAR0	=	177640	11-378 #11-379 321-6816 379-303 381-343 381-343
UIPAR2	=	177644	#11-381 *67-1861 *67-1896 *169-4130 *177-4307 193-4510 263-5781 *270-5946 *369-134
UIPAR3	=	177646	#11-382 67-1882 169-4131 177-4308 369-135
UIPAR4	=	177650	#11-383 319-6778 319-6795 *375-251
UIPAR5	=	177652	#11-384 375-252
UIPAR6	=	177654	#11-385 *67-1847 195-4533
UIPAR7	=	177656	#11-386 *67-1848
UIPDRO	=	177600	#11-358 321-6818 397-662 400-747
UNITOP	002516	#38-1264 *75-2102 79-2167 79-2167 *96-2645 *96-2646 *96-2647 *96-2648 *96-2649	
UNRELO	041604	96-2651 96-2651	
UPPFLG	002363	111-2889 113-2923 115-2964 117-3005 120-3048 122-3097 124-3139 126-3188 171-4210	
UP2	056154	#317-6718 333-7035 335-7073	
USERMA	042240	#38-1214 *217-4968 217-4983 *217-4985 *223-5123 225-5140 *225-5142	
USESTK	= 000700	403-790 #403-793	
USP	= %000006	315-6698 317-6739 319-6773 #321-6815	
WAITST	012034	#11-258 60-1660 375-220 481-2738 481-2744	
WAIT5	063674	#11-278 *60-1661 398-671 *401-761 481-2735	
WAKEUP	054544	83-2286 #85-2305	
WARN1	007320	433-1546 433-1563 #435-1600	
WARN2	026522	389-498 #391-558	
WARN3	026536	#67-1861	
WARN4	026562	#191-4465 *270-5933	
WARN5	026576	#191-4470 *270-5934	
WARN6	035364	#191-4482 *270-5935	
WARN7	024416	#191-4487 *270-5936	
WASDBE	= 104500	#270-5946	
WASSBE	= 104476	#169-4130	
WAS1DB	= 104501	#9-240 69-1976 69-1985 69-1993 69-2001	
WAS1SB	= 104477	#9-238 149-3669	
WHICHC	050630	#9-241 212-4898	
WHOBX	070672	#9-239 162-3990 162-3996 162-4003 162-4009	
WOOPEN	056512	340-7188 342-7203 #363-7775	
WOOPS	056160	466-2394 #475-2581	
WOOPSA	056542	403-802 403-805 403-807 405-821 405-842 #405-850 405-853	
WOOPUP	056336	397-659 #403-795	
WORST	002712	*403-800 *403-801 403-802 405-837 405-838 405-841 #405-853	
WRITED	002502	403-802 403-802 403-803 403-805 403-805 403-805 #405-820 405-842 405-843	
XXDPCH	002446	405-853	
ZERO	002426	#42-1320 *47-1444 *115-2952 *117-2993 *122-3085 *126-3176 324-6907	
ZEROS	002430	#38-1258 *104-2779 *104-2782 177-4309 *367-85 *367-96 369-138 *375-227 375-255	
\$APTHD	075236	*375-258 *377-270 *377-276 *414-1181 *414-1183	
\$AUTO	002066	#38-1241 *52-1536 460-2267	
\$BANK	002015	#38-1233	
\$BASE	075210	#38-1234 146-3641 146-3642 248-5495 248-5499	
\$BELL	003003	36-1096 #496-3436	
\$CACHF	037066	#38-1144 *52-1528 *52-1532 62-1665 487-2956	
		#38-1123 *457-2177	
		#496-3408	
		#42-1353 353-7538 357-7687 459-2221	
		#285-6268 499-3504	

SYMBOL	VALUE	REFERENCES								
\$CACHN	037050	#285-6262	499-3503							
\$CBCSR	037700	#293-6389	499-3547							
\$CBTCS	037742	#293-6396	499-3548							
\$CDW1	075214	92-2547	92-2549	#496-3413						
\$CDW2	075216	92-2547	92-2550	#496-3414						
\$CHARC	057356	*407-895	*407-905	*407-939	#407-944					
\$CHKDI	040354	#300-6489	499-3555							
\$CHK1D	040410	#300-6496	499-3556							
\$CKSWR	073022	#487-2931	499-3487							
\$CLRCS	040322	#300-6480	499-3553							
\$CLR1C	040340	#300-6484	499-3554							
\$CMTAG	002000	#38-1114	45-1416							
\$CMTGE	002700	#40-1311	45-1418							
\$CNTLC	074224	487-2970	490-3089	#490-3140						
\$CNTLG	074236	487-2958	#490-3142							
\$CNTLU	074231	487-2975	490-3113	#490-3141						
\$CPUOP	075162	#496-3381								
\$CRLF	003010	#42-1355	96-2652	273-5972	273-5975	355-7637	407-894	462-2290	462-2313	462-2322
		479-2684	481-2710	481-2724	481-2740	481-2748	487-2984	489-3009	490-3118	
\$DBLK	073012	486-2884	486-2914	#486-2922						
\$DB20	075014	468-2441	#495-3331							
\$DDW0	075220	54-1548	#496-3425							
\$DDW1	075222	#496-3426								
\$DDW2	075224	#496-3427								
\$DDW3	075226	#496-3428								
\$DDW4	075230	#496-3429								
\$DDW5	075232	#496-3430								
\$DDW6	075234	98-2701	98-2703	#496-3431						
\$DEENE	037040	#285-6258	499-3499							
\$DEVCT	075144	*107-2834	*456-2129	*456-2131	#496-3371					
\$DEVVM	075212	#496-3409								
\$DOAGA	015040	107-2808	107-2810	#107-2831						
\$DOAGN	015010	#107-2820								
\$DOWN	055532	#398-705	398-706							
\$DTBL	073002	486-2887	#486-2918							
\$ECCDI	037466	#293-6355	499-3543							
\$ECCIN	037554	#293-6367	499-3545							
\$ECC1D	037522	#293-6361	499-3544							
\$ECC1I	037574	#293-6372	499-3546							
\$ENASB	037612	#293-6377	499-3557							
\$ENA1S	037646	#293-6383	499-3558							
\$ENDAD	015000	36-1087	52-1533	56-1581	#107-2816					
\$ENERG	037030	#285-6254	499-3498							
\$ENV	075154	52-1527	#496-3376							
\$ENVVM	075155	52-1520	52-1523	#496-3377						
\$EOP	014662	#107-2796								
\$ERFLG	002016	#38-1124	456-2162	*456-2168	*459-2215					
\$ERROR	066662	47-1459	#459-2213							
\$ERRTB	101372	462-2301	#548-4417							
\$ERRTY	067324	459-2249	#462-2289							
\$ERTTL	002754	#42-1337	79-2185	104-2774	107-2798	355-7620	355-7622	*459-2223	*459-2225	460-2267
\$ESCAP	002454	#38-1247	*456-2171	460-2259	460-2261					

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
SETABL		075154	#496-3375
SETEND		075236	#496-3432 496-3442
SEXHAL		043546	#335-7058
SFATAL		075136	*459-2241 #496-3368
SFILLC		003002	#42-1352 407-898 542-4290
SFILLS		002451	#38-1244 *359-7706 542-4292
SGTSWR		073160	#487-2959 499-3486
SHALT		067136	#460-2254
SHALT2		075312	#497-3468
SHIBTS		075236	#496-3437
SHIOCT		074434	344-7237 346-7276 355-7561 *491-3184 #491-3195
SILLUP		056134	397-646 400-712 #401-782 401-783
SINVAL		040510	#302-6519 499-3560
SITEMB		002017	#38-1125 *459-2232 462-2292
SKERNE		037020	#285-6250 499-3497
SKMAP		041016	#310-6628 499-3501
SLF		003011	#42-1356 490-3128
SLOADC		037076	#288-6275 499-3506
SLOAD1		037146	288-6279 #288-6288
SLPADR		002746	#42-1334 *47-1454 182-4337 *182-4347 182-4348 *182-4364 185-4371 *185-4381 185-4382
			*185-4398 *456-2166 *456-2169 456-2173
SLPERR	002750		#42-1335 *47-1455 182-4337 *182-4348 *182-4364 185-4371 *185-4382 *185-4398 456-2166
			*456-2170 460-2257
SMADR1		075166	#496-3395
SMADR2		075172	#496-3399
SMADR3		075176	#496-3402
SMADR4		075202	#496-3405
SMAIL		075134	#496-3366 496-3438 496-3442
SMAMS1		075164	98-2667 #496-3388
SMAMS2		075170	#496-3397
SMAMS3		075174	#496-3400
SMAMS4		075200	#496-3403
SMBADR		075240	#496-3438
SMNEW		074254	487-2961 #490-3144
SMSGAD		075150	#496-3373
SMSGLG		075152	#496-3374
SMSGTY		075134	*460-2264 #496-3367
SMSWR		074243	487-2959 #490-3143
SMTYP1		075165	#496-3389
SMTYP2		075171	#496-3398
SMTYP3		075175	#496-3401
SMTYP4		075201	#496-3404
SNOTRA		075306	#497-3467 499-3482 499-3484 499-3536 499-3537 499-3538 499-3539 499-3540 499-3541
			499-3561 499-3562 499-3563 499-3564 499-3565 499-3566 499-3567 499-3568
SNULL	002450		#38-1243 407-900
SNWTST	= 000001		#64-1689 64-1689 #64-1689 #65-1796 65-1796 #65-1796 #67-1833 67-1833 #67-1833
			#71-2031 71-2031 #71-2031 #99-2710 99-2710 #99-2710 #102-2730 102-2730 #102-2730
			#102-2740 102-2740 #102-2740
SOCNT	072572		*485-2821 *485-2850 #485-2863
SOCTVL	075116		495-3333 #495-3358 495-3361
SOCT8	= 075122		468-2443 #495-3361
SOMODE	072574		*485-2816 *485-2820 485-2825 *485-2828 *485-2839 #485-2865

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES							
\$OVER		066502	456-2148	456-2167	#456-2172					
\$PASS		075142	*52-1519	*92-2557	*107-2799	*107-2800	107-2806	107-2831	*107-2837	158-3889 158-3898
			158-3906	158-3916	160-3927	160-3936	162-3951	165-4092	#496-3370	
\$PASTM		075244	#496-3440							
\$PATMA		002014	#38-1122	401-774	401-775	*457-2178	*457-2187	*457-2190	*457-2191	457-2192 457-2193
			459-2218							
\$PER01		064574	#445-1883	499-3509						
\$PER02		064622	#445-1889	499-3510						
\$PER03		064650	#445-1895	499-3511						
\$PER04		064700	#445-1901	499-3512						
\$PER07		064762	#445-1917	499-3513						
\$PER10		065004	#445-1922	499-3514						
\$PER11		065034	#445-1927	499-3515						
\$PER12		065054	#445-1932	499-3516						
\$PER13		065076	#447-1938	499-3517						
\$PER14		065116	#447-1943	499-3518						
\$PER15		065140	#447-1948	499-3519						
\$PER16		065162	#447-1953	499-3520						
\$PER17		065202	#447-1958	499-3521						
\$PER20		065220	#447-1963	499-3522						
\$PER21		065236	#447-1968	499-3523						
\$PER22		065256	#447-1973	499-3524						
\$PER23		065274	#447-1978	499-3525						
\$PER24		065312	#447-1983	499-3526						
\$PER25		054570	#393-569	499-3527						
\$PER26		065502	#450-2020	499-3528						
\$PER27		065522	#450-2024	499-3529						
\$PER30		054762	#395-605	499-3530						
\$PER31		065720	#452-2064	499-3531						
\$PER32		066016	#454-2082	499-3532						
\$PER33		066064	#454-2091	499-3533						
\$PER34		066144	#454-2102	499-3534						
\$PER35		066176	#454-2111	499-3535						
\$PWRDN		055160	47-1463	#397-638	401-777					
\$PWRUP		055536	398-704	#400-711	405-849					
\$QUES		003007	#42-1354	487-3002	490-3121					
\$RAND		074720	387-438	#493-3302						
\$RDCHR		073602	#490-3053	499-3489						
\$RDDEC		074436	#491-3210	499-3492						
\$RDLIN		073722	#490-3080	499-3490						
\$RDOCT		074266	#491-3159	499-3491						
\$READC		037220	#290-6304	499-3507						
\$RESRE		074662	#492-3283	499-3495						
\$SAVRE		074624	#492-3272	499-3494						
\$SAVR6		056140	*398-702	400-714	*400-715	*400-716	#401-784	405-832		
\$SCOPE		066232	47-1457	#456-2129						
\$STN	=	000001	#64-1689	#65-1796	#67-1833	#71-2031	#99-2710	#102-2730	#102-2740	
\$SVLAD		066466	456-2156	456-2163	#456-2169					
\$SWR	=	163000	#7-131	64-1689	65-1796	67-1833	71-2031	99-2710	102-2730	102-2740
\$SWREG		075156	52-1525	#496-3379						
\$TESTN		075140	*459-2218	#496-3369						
\$TKB		002774	#42-1349	273-5958	273-5994	348-7371	348-7432	351-7465	351-7475	353-7521 353-7543

SYMBOL	CROSS REFERENCE VALUE	REFERENCES								
\$TKS	002772	355-7594	355-7612	357-7670	357-7692	487-2936	487-2966	489-3037	490-3057	490-3063
		#42-1348	273-5960	273-5993	348-7373	348-7430	351-7467	351-7473	353-7523	353-7542
\$TN	= 000010	355-7596	355-7611	357-7672	357-7691	487-2934	487-2964	489-3035	490-3055	490-3061
		#7-132	64-1689	64-1689	#64-1689	65-1796	65-1796	#65-1796	67-1833	67-1833
		#67-1833	71-2031	71-2031	#71-2031	99-2710	99-2710	#99-2710	102-2730	102-2730
		#102-2730	102-2740	102-2740	#102-2740					
\$TPB	003000	#42-1351	407-935							
\$TPFLG	002452	#38-1245	*52-1521	407-877						
\$TPS	002776	#42-1350	407-933							
\$TRAP	075252	47-1461	#497-3452							
\$TRAP2	075274	#497-3463	499-3478							
\$TRPAD	075314	497-3457	#499-3478							
\$TSTM	075242	#496-3439								
\$TSTRD	037310	#290-6322	499-3559							
\$TTYIN	074200	490-3082	490-3083	490-3101	490-3119	490-3133	#490-3137			
\$TYPDS	072576	#486-2877	499-3483							
\$TYPE	056752	#407-877	499-3479							
\$TYPEC	057076	407-897	407-904	#407-907	483-2768	487-2988				
\$TYPEX	057360	407-940	407-942	#407-945						
\$TYPOC	072374	#485-2819	499-3480							
\$TYPON	072410	485-2818	#485-2821							
\$TYPOS	072350	#485-2814	499-3481							
\$UNIT	075146	*107-2835	*456-2132	#496-3372						
\$UNITM	075246	#496-3441								
\$USWR	075160	107-2831	#496-3380							
\$VECT1	075204	#496-3406								
\$VECT2	075206	#496-3407								
\$WASDB	040146	#298-6441	499-3551							
\$WASSB	040002	#296-6406	499-3549							
\$WAS1D	040262	#298-6467	499-3552							
\$WAS1S	040116	#296-6432	499-3550							
\$XTSTR	066370	#456-2150								
\$ZAP42	014760	#107-2809	460-2271							
\$OFILL	072573	*485-2815	*485-2819	485-2829	#485-2864					

6-	90	OPERATIONAL SWITCH SETTINGS	
6-	91	;SWITCH REGISTER DEFINITIONS	
6-	92	;*	
6-	93	;	SWITCH
6-	94	;	USE
6-	95	;	-----
6-	96	;	15 HALT ON ERROR
6-	97	;	14 LOOP ON TEST
6-	98	;	13 INHIBIT ERROR TYPEOUTS
6-	99	;	12 INHIBIT RELOCATION
6-	100	;	11 QUICK VERIFY / INHIBIT MARGIN TESTING
6-	101	;	10 BELL ON ERROR
6-	102	;	9 LOOP ON ERROR
6-	103	;	8 HALT PROGRAM (UNRELOCATED & RESTORE LOADERS)
6-	104	;	7 DETAILED ERROR REPORTS
6-	105	;	6 PRINT CONFIGURATION MAP
6-	106	;	5 LIMIT MAX ERRORS PER BANK
6-	107	;	4 FAT TERMINAL (132 COLUMNS OR BETTER)
6-	108	;	3 TEST MODE - SEE DOCUMENT
6-	109	;	2 TEST MODE - SEE DOCUMENT
6-	110	;	1 TEST MODE - SEE DOCUMENT
6-	110	;	0 DETECT SINGLE BIT ERRORS
10-	136	DEFINE	TRAPS
12-	252	DEFINE	BASIC PDP11 STUFF
12-	334	DEFINE	CACHE REGISTERS
12-	342	DEFINE	CPU REGISTERS
12-	350	DEFINE	MEMORY MANAGEMENT REGISTERS
14-	480	DEFINE	UNIBUS MAP REGISTERS
14-	548	DEFINE	SOFTWARE SWITCH & DISPLAY REGISTERS
14-	552	DEFINE	MK11 REGISTERS
14-	555	DEFINE	PARAMETERS
14-	560	DEFINE	MASTER & SLAVE BITS
16-	568	MACRO	FATAL
16-	588	MACRO	TYPE
18-	606	MACRO	NEWTST
20-	656	MACRO	\$\$NEWTST
20-	677	MACRO	SUBTST
20-	696	MACRO	\$\$SUBTST
22-	711	MACRO	TYPOCT
24-	751	MACRO	TYPOCS
26-	807	MACRO	TYPDEC
27-	849	MACRO	BMOV
29-	915	MACRO	MAP
31-	954	MACRO	SUPERVISOR
31-	975	MACRO	USER
33-	997	MACRO	SET4 & RES4
35-	1039	MACRO	DLEFT
37-	1063	TRAP CATCHER	
37-	1071	ACT11 HOOKS	
37-	1090	APT11 HOOKS	
39-	1111	VARIABLES	INITIALIZED TO ZERO
41-	1294	MULTIPOINT VARIABLES	
43-	1314	VARIABLES	INITIALIZED TO NON ZERO
45-	1361	CONFIGURATION TABLE	
46-	1402	***** MAIN *****	
46-	1403	INITIALIZE VARIABLES TO ZERO	
46-	1420	CLEAR NON-PROGRAM SPACE	
48-	1443	INITIALIZE VARIABLES TO NON ZERO	

48- 1456	:INITIALIZE VECTORS	
51- 1479	CLEAR THE CONFIGURATION TABLE	
51- 1491	SIZE FOR A HARDWARE SWITCH REGISTER	
51- 1508	FIND OUT IF ANY LINE CLOCK IS PRESENT	
53- 1516	SETUP ACT, APT, & XXDP	
55- 1542	INITIALIZE PATTERNS	
55- 1567	SUBR PLUG IN NULL PATTERNS	
57- 1578	PROTECT PROGRAM & LOADERS	
57- 1584	MULTIPOINT DETERMINATION ROUTINE	
59- 1618	SLAVES START HERE	
61- 1630	DETERMINE MK11 ACCESS METHOD	
61- 1647	SETUP USER & SUPERVISOR STACK	
63- 1664	GET SOFTWARE SWITCH REGISTER IF NECESSARY	
63- 1674	SETUP UNIBUS MAP	
63- 1681	GET MEMORY MANAGEMENT READY	
65- 1689	T1 ANALIZE CONFIGURATION	
66- 1796	T2 TEST BANK 0 ACCESSES	
66- 1825	ENABLE MK11 FOR CORRECT TRAPS	
68- 1833	T3 TEST BANKS 1-167 (OCTAL) FOR ZEROS & ONES	
70- 1937	DETERMINE PROTECTED BANKS	
72- 2031	T4 MK11 PROTECTION POINTER TEST	
74- 2046	CHECK SYSTEM SIZE REGISTER	
82- 2201	MOVE LOADERS IF NECESSARY	
84- 2242	MULTIPOINT HOOK	
86- 2305	SLAVES WAIT FOR MASTER TO COUNT DOWN	
88- 2319	BOOT UP ALL EXISTING CPU'S	
93- 2512	CHECK FOR UNIQUE CPU ID NUMBERS	
93- 2532	PRINT MULTIPOINT DIRECTORY	
93- 2546	LOAD CSR'S FROM E-TABLE \$CDW1 & \$CDW2	
95- 2562	LEGAL CONFIGURATION CHECK	
97- 2626	PRINT CONFIGURATION DETAILS	
99- 2664	CHECK APT SIZING	
100- 2710	T5 DIAGNOSTIC MODE DISPATCH ROUTINE	
103- 2730	T6 UNIQUE BANK TEST	
103- 2740	T7 FORCE ADDRESS PARITY ERRORS	
105- 2784	FLUSH OUT DBE'S	
108- 2789	END OF PASS ROUTINE	
110- 2843	WRITE BACKGROUND PATTERNS	
112- 2857	MTEST MODES	
112- 2859	BANKS FORWARD,PATTERNS FORWARD	**RECURSIVE**
114- 2893	BANKS FORWARD,PATTERNS REVERSE	**RECURSIVE**
116- 2927	BANKS WORST FIRST,PATTERNS FORWARD	**RECURSIVE**
118- 2968	BANKS WORST FIRST,PATTERNS REVERSE	**RECURSIVE**
121- 3010	PATTERNS FORWARD,BANKS FORWARD	**RECURSIVE**
123- 3052	PATTERNS FORWARD,BANKS WORST FIRST	**RECURSIVE**
125- 3101	PATTERNS REVERSE,BANKS FORWARD	**RECURSIVE**
127- 3143	PATTERNS REVERSE,BANKS WORST FIRST	**RECURSIVE**
130- 3193	SUBR SETUP MEMORY TEST	
132- 3214	SUBR TEST MK11 CONTROLLER LOGIC DISPATCH	
134- 3284	IS ECC ON?	
136- 3306	CHECK FOR SBE FREE LOCATIONS	
138- 3384	SETUP INTERLEAVE INFO	
140- 3421	SUBR GET SIZE FROM CSR	
143- 3448	CSR PATTERN CASE STATEMENT	
145- 3490	DETERMINE ADDRESSING ACCORDING TO INTERLEAVE	
147- 3624	DETERMINE IF BANK BELONGS TO CORRECT SIDE	
150- 3653	SUBR MK11 TEST DISPATCH	

152-	3724	SUBR	MJ11 TEST DISPATCH
153-	3778		PATTERNS
153-	3780	MEMORY	TEST SETUP ROUTINES
153-	3781	MT0000	SETUP DATA PATTERN TEST
153-	3789	MT0001	SETUP ADDRESS TEST
153-	3798	MT0002	SETUP COMPLEMENT ADDRESS TEST
155-	3811	MT0003	SETUP 3 XOR 9 WORST CASE NOISE TEST
155-	3836	MT0004	SETUP ROTATING ZEROS TEST
155-	3847	MT0005	SETUP ROTATING ONES TEST
157-	3860	MT0006	SETUP INITIAL DATA TEST
157-	3867	MT0007	SETUP ADDRESS BIT TEST
157-	3877	MT0010	SETUP BYTE ADDRESSING TEST
159-	3886	MT0011	SETUP CREATE SINGLE BIT ERROR TEST
159-	3895	MT0012	SETUP WRITE BYTE CLEARS SBE TEST
159-	3904	MT0013	SETUP CREATE DOUBLE BIT ERROR TEST
159-	3913	MT0014	SETUP WRITE INHIBIT DURING DATIP WITH DBE
161-	3924	MT0015	SETUP WRITE INHIBIT OF BYTE WITH DBE
161-	3933	MT0016	SETUP WRITE INHIBIT OF WORD WITH DBE
161-	3942	MT0017	SETUP HOLDING 1'S & 0'S
163-	3949	MT0020	SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST
165-	4021	MT0021	SETUP MARCHING 0'S & 1'S TEST
165-	4046	MT0022	SETUP REFRESH & SHIFTING DIAGONAL TEST
165-	4054	MT0023	SHIFTING DIAGONAL TEST
166-	4064	MT0024	SETUP FAST GALLOPING PATTERN TEST
166-	4090	MT0025	SETUP INTERRUPT ENABLE TEST
168-	4100	MT0026	SETUP RANDOM DATA TEST
170-	4123	MT0027	UNIQUE BANK TEST
172-	4179	MT0030	SETUP FLUSH OUT DBE'S TEST
174-	4218	MT0031	SETUP SOB-A-LONG TEST
176-	4238	MT0033	SETUP WRITE RECOVERY TEST
178-	4276	MT0034	SETUP BRANCH GOBBLE TEST
178-	4297	MT0035	SOFT ERROR - BACKGROUND PATTERN TEST
180-	4320	MT0999	SETUP NULL TEST
180-	4325		CHECK FOR KAMIKAZE MODE
183-	4334	SUBR	EXECUTE PATTERN IN SUPERVISOR
189-	4403	MEMORY	TEST PATTERN ROUTINES
189-	4413	MTP000	BASIC DATA TEST
189-	4424	MTP001	ADDRESS TEST
189-	4436	MTP002	COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
192-	4451	MTPA03	3 XOR 9 WORST CASE NOISE TEST (WRITE)
192-	4476	MTPB03	3 XOR 9 WORST CASE NOISE TEST (READ)
194-	4494	MTPC03	TEST DATA SUBPROGRAM
194-	4503	MTPD03	TEST DATA SUBSUBPROGRAM
196-	4513	MTPA04	ROTATING ZEROS TEST
196-	4526	MTPB04	SUBR ROTATING BIT
196-	4535	MTP005	ROTATION ONES TEST
199-	4550	MTP006	INITIAL DATA TEST
202-	4594	MTP007	ADDRESS BIT TEST
204-	4634	MTP010	BYTE ADDRESSING TEST
207-	4671	MPT011	SINGLE BIT ERROR TEST
210-	4782	MTP012	WRITE BYTE CLEARS SBE TEST
213-	4860	MTP013	CREATE DOUBLE BIT ERROR TEST
218-	4929	MTP014	WRITE INHIBIT DURING DATIP WITH DBE TEST
221-	5010	MTP015	WRITE INHIBIT OF BYTE WITH DBE
224-	5087	MTP016	WRITE INHIBIT OF WORD WITH DBE
229-	5167	MTP017	HOLDING 1'S & 0'S TEST
232-	5201	MTP020	MARCHING 1'S & 0'S IN CHECK BITS TEST

237- 5279	MTPA20	SLOW MARCHING 1'S & 0'S IN CHECK BITS TEST
244- 5416	MTPA21	MARCHING 1'S & 0'S PATTERN TEST
249- 5487	MTP022	REFRESH & SHIFTING DIAGONAL TEST
251- 5559	SUBR	REFRESH DELAY
254- 5582	MTPA24	FAST GALLOPING PATTERN TEST
256- 5626	MTPB24	FAST GALLOP PART B
256- 5634	MTPC24	FAST GALLOP PART C
259- 5645	MTP025	INTERRUPT ENABLE TEST
264- 5738	MTPA26	RANDOM DATA (WRITE)
264- 5745	MTPB26	RANDOM DATA (READ)
264- 5763	RANDOM	NUMBER SUBPROGRAM
264- 5776	RANDOM	NUMBER SUBSUBPROGRAM
266- 5784	MTO030	FLUSH OUT DBE'S
266- 5790	MTP031	SOB-A-LONG TEST
268- 5816	MTPA32	SKEWED UP ADDRESS TEST (WRITE)
268- 5826	MTPB32	SKEWED UP ADDRESS TEST (READ)
268- 5839	MTP033	WRITE RECOVERY TEST
270- 5859	MTP034	BRANCH GOBBLE TEST
271- 5905	MISC	SUBROUTINES
271- 5907	SUBR	COPY R0 TO R4, R1 TO R3, & R2 TO R5
271- 5913	FLIP	WARNING CONSTANTS IN WORST CASE NOISE TESTS
271- 5940	SUBR	WRITE BACKGROUND
274- 5953	SUBR	PRINT CONFIGURATION MAP
277- 6002	SUBR	TYPE CONFIGURATION
282- 6154	TRAP	PARITY ERROR HANDLER
284- 6200	TRAP	NON-EXISTANT MEMORY (HOLES) HANDLER
284- 6222	TRAP	TIMEOUT (TRAP TO 4) HANDLER
284- 6226	TRAP	MEMORY MANAGEMENT (TRAP TO 250) HANDLER
284- 6229	TRAP	RESERVED INSTRUCTION HANDLER
284- 6233	UNEXPECTED	IIST TRAP HANDLER
284- 6237	FIND BAD	SP, PC, & PSW FROM STACK
286- 6245	TRAP	KERNEL TRAP HANDLER
286- 6253	TRAP	ENERGIZE TRAP HANDLER
286- 6257	TRAP	DEENERGIZE TRAP HANDLER
286- 6261	TRAP	CACHON TRAP HANDLER
286- 6266	TRAP	CACHOFF TRAP HANDLER
289- 6273	TRAP	LOAD CSR TRAP HANDLER
291- 6302	TRAP	READ CSR TRAP HANDLER
291- 6321	TRAP	TEST (R1) & READ CSR CAREFULLY
294- 6354	TRAP	ECC DISABLE ALL CSR'S TRAP HANDLER
294- 6360	TRAP	ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
294- 6366	TRAP	MK11 INITIALIZE ALL CSR'S TRAP HANDLER
294- 6371	TRAP	MK11 INITIALIZE 1 SELECTED CSR TRAP HANDLER
294- 6376	TRAP	ENABLE SBE PARITY TRAPS ON ALL CSR'S
294- 6382	TRAP	ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
294- 6388	TRAP	WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
294- 6395	TRAP	WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
297- 6405	TRAP	WAS THERE A SBE ON ANY CSR TRAP HANDLER
297- 6430	TRAP	WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
299- 6440	TRAP	WAS THERE A DBE ON ANY CSR TRAP HANDLER
299- 6465	TRAP	WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
301- 6479	TRAP	CLEAR ALL MK11 CSR'S TRAP HANDLER
301- 6483	TRAP	CLEAR 1 SELECTED MK11 CSR TRAP HANDLER
301- 6487	TRAP	ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
301- 6494	TRAP	ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
303- 6503	SUBR	WRITE IN ALL CSR'S
303- 6518	TRAP	INVALIDATE BACKGROUND PATTERN

306-	6528	SUBR	GENERATE CHECK BITS
311-	6595	SUBR	MAPPER
311-	6627	TRAP	MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
314-	6649		RELOCATE PROGRAM
318-	6718		UNRELOCATE PROGRAM
318-	6754		SETUP LOWER 16K OF UNIBUS MAP
320-	6767		MOVE BANKS
322-	6815	SUBR	MAP USER TO NEW BANK
322-	6835	SUBR	SETUP KERNEL PAR'S FOR NEW BANK
322-	6848	SUBR	SETUP KERNEL PAR'S FOR NEW LOADER BANK
325-	6861	SUBR	EXAMINE BANK
328-	6918	SUBR	BANK OK?
328-	6929	SUBR	INCREMENT PATTERN TESTING MK11'S
328-	6937	SUBR	SET HIGHEST PATTERN TESTING TYPE
328-	6941	SUBR	INCREMENT BANK & TEST
330-	6948	SUBR	INCREMENT MARGINS & TEST
330-	6976		DECREMENT MARGINS & TEST
332-	6989	SUBR	SET HIGHEST MARGIN
332-	7006	SUBR	REWRITE CORE HISTORY
334-	7024		BOOTSTRAP ROUTINE
336-	7053		HALT PROGRAM
336-	7062		SHUTDOWN DIAGNOSTIC
336-	7085		APT SHUTDOWN SEQUENCE
338-	7095		BLOCK MOVE SUBROUTINE
339-	7122		FIELD SERVICE MODE
339-	7124	SUBR	FIELD SERVICE COMMAND MODE
341-	7169	COMMAND 0	EXIT
341-	7187	FS	COMMAND 1 READ CSR
343-	7202	FS	COMMAND 2 LOAD CSR
345-	7231	FS	COMMAND 3 EXAMINE MEMORY
347-	7270	FS	COMMAND 4 MODIFY MEMORY
349-	7322	FS	COMMAND 5 SELECT BANK, MARGIN, & PATTERN
349-	7440	FS	COMMAND 6 TYPE CONFIGURATION MAP
352-	7447	FS	COMMAND 7 BATTERY BACKUP TEST
354-	7496	FS	COMMAND 8 SOB-A-LONG TEST
356-	7550	FS	COMMAND 9 SUPER TIGHT SCOPE LOOP
356-	7618	FS	COMMAND 10 ERROR SUMMARY
358-	7645	FS	COMMAND 11 REFRESH TEST
360-	7700	FS	COMMAND 12 SET FILL COUNT
360-	7710	FS	COMMAND 13 ENTER KAMIKAZE MODE
360-	7715	FS	COMMAND 14 EXIT KAMIKAZE MODE
360-	7721	FS	COMMAND 15 TURN CACHE OFF
360-	7727	FS	COMMAND 16 TURN CACHE ON
360-	7733	FS	COMMAND 17 RUN ONLY MULTIPOINT TESTS
362-	7739	FS	COMMAND 18 RESUME RUNNING BOTH SINGLE & MULTIPOINT TESTS
362-	7744	FS	COMMAND 19 TEST ONLY SELECTED BANKS
362-	7763	FS	COMMAND 20 RESUME TESTING ALL BANKS
364-	7775	SUBR	DETERMINE CORRECT CSR
366-	3		MULTIPROCESSOR SLAVE DISPATCHER
368-	66		MOVE PROGRAM TO NEW BANK
368-	80		WRITE ONLY PATTERN IN HI BYTE
370-	100		READ ONLY PATTERN IN HI BYTE
370-	130		SLAVE PATTERNS
372-	145		EXECUTION CONTENTION TEST
374-	164		PORT ARBITRATION SYMMETRY TEST
376-	219		STOP SLAVE PROGRAM
376-	226		WRITE ONLY ADDRESS PATTERN IN HI BYTE

376-	230	READ ONLY ADDRESS PATTERN IN HI BYTE
378-	262	RUN FIVE PATTERNS IN DESIGNATED 1/4 BANK
378-	279	WRITE & READ A SKEWED UP ADDRESS TEST
380-	301	SETUP FOR ASRB TEST
382-	338	RECOVER FROM TASK13 - TRAP FROM IIST
384-	356	RESET FROM ASRB TEST
384-	361	RUN SLAVE IN CSR TESTS
384-	365	SET MARGIN 6, CPU HAS I/O PRIORITY
384-	370	CREATE STALE DATA TEST
386-	385	CACHE FLUSH TEST
386-	422	CACHE FLUSH TEST (FOR PAR'S)
388-	434	SETUP EXECUTION CONTENTION TEST
390-	458	MULTIPOINT ASRB TEST (IN SUPERVISOR SPACE)
392-	541	BALANCE TEST
394-	568	ERROR DATA (SUPERVISOR) SETUP STUFF
394-	582	DATA WAS 3 WORDS
396-	614	GET DATA FROM ABORTED AREA IF POSSIBLE
398-	634	POWER FAIL AUTO RESTART
398-	635	ROUTINE POWER DOWN AND UP
404-	788	MULTIPLE CPU'S POWERING UP
404-	795	POWER FAIL WHILE RELOCATED
406-	820	POWER UP FROM BANK 0 TO RELOCATION
408-	858	IO SUBROUTINES
408-	860	ROUTINE TYPE
409-	949	MULTIPOINT SLAVE STARTER
411-	1032	HAVE SLAVES RUN CSR TESTS
413-	1064	MOVE SLAVES TO THERE OWN BANKS
415-	1111	RUN NORMAL MULTIPROCESSOR TESTS
417-	1187	INITIALIZE THE IIST
417-	1207	INIT IIST & DISABLE BOOTS & INTERRUPTS
419-	1221	SLAVES MAP
421-	1241	MULTIPOINT ACCESS PATHS TEST
424-	1306	MULTIPOINT ADDRESS TEST
426-	1369	MULTIPOINT ASYNCHRONOUS ADDRESS UNIQUENESS TEST
428-	1417	MULTIPOINT SKEWED UP ADDRESS TEST
430-	1458	MULTIPOINT PORT ARBITRATION SYMMETRY TEST
432-	1506	MULTIPOINT PORT I/O PRIORITY TEST
436-	1575	CHECK FOR PROPER SKEWING OF ACCESSES
436-	1600	WAIT FOR 5 SECONDS FOR A SLAVE
438-	1615	MULTIPOINT EXECUTION CONTENTION TEST
438-	1662	MULTIPOINT ASRB TEST
440-	1666	MULTIPOINT CACHE FLUSH TEST
442-	1717	STOP SLAVES
444-	1739	ERROR DATA SETUP
449-	1988	DATA WAS A WORD
449-	2000	DATA WAS A BYTE
451-	2013	DATA WAS A 7 BIT BYTE
451-	2028	DETERMINE XOR OF GOOD & BAD
453-	2037	LOG ERROR ON BAD BANK
457-	2120	ROUTINE SCOPE HANDLER
458-	2176	SUBR DISPLAY
460-	2199	ROUTINE ERROR HANDLER
463-	2282	ROUTINE ERROR MESSAGE TYPEOUT
476-	2581	SUBR WHO'S BOX IS THIS?
478-	2645	SUBR DETAILED ERROR REPORT
484-	2754	QUE MESSAGES FROM SLAVE CPU'S
486-	2789	ROUTINE BINARY TO OCTAL (ASCII) AND TYPE

487-	2867	ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
488-	2924	ROUTINE TTY INPUT
490-	3007	CONTROL T
490-	3033	CONTROL S & CONTROL Q
492-	3147	ROUTINE READ AN OCTAL NUMBER FROM THE TTY
492-	3196	ROUTINE READ A DECIMAL NUMBER FROM THE TTY
493-	3255	ROUTINE SAVE AND RESTORE R0-R5
494-	3291	ROUTINE RANDOM NUMBER GENERATOR
496-	3321	ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
497-	3363	TABLES
497-	3365	APT MAILBOX-ETABLE
498-	3444	ROUTINE TRAP DECODER
500-	3471	TRAP TABLE
502-	3571	ODT (CEMKA) DATA AREA
549-	4403	TABLE ERROR POINTER
561-	4699	ERROR DATA TAGS (DT)
563-	4728	ERROR DATA FORMATS (DF)
565-	4746	ERROR MESSAGES (EM)
567-	4806	ERROR DATA HEADERS (DH)
569-	4838	MESSAGES

1

000000

LSTSS=0

.TITLE CEMKABO 1170 MAIN MEMORY DIAG 1
.IDENT /X01.60/

IDENTIFICATION

PRODUCT CODE: AC-C644B-MC
PRODUCT NAME: CEMKABO 1170 MAIN MEMORY DIAG
PRODUCT DATE: 31 JUL 1979
MAINTAINER: PRODUCT ENHANCEMENT

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 DIGITALS 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) 1978,1979 DIGITAL EQUIPMENT CORPORATION

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

36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88

REVISION HISTORY
=====

REVISION	VERSION	DATE	AUTHOR	CHANGES
=====	=====	=====	=====	=====
CEMKAA	V03.00	24-FEB-78	KAY R. FISHER	NONE = NEW PROGRAM
CEMKAB	V09.00	9-OCT-78	KAY R. FISHER	1. ADD MULTIPROCESSOR CAPABILITIES 2. ADD SOFT ERROR TEST FOR ALPHA PARTICLES IN MOS CHIPS 3. ADD THE FOLLOWING FIELD SERVICE COMMANDS A. KAMIKAZE MODE B. CACHE ON/OFF C. SELECT ONLY MULTIPOINT TESTS D. TEST ONLY SELECTED BANKS 4. EXPAND DETAILED ERROR REPORT TO INCLUDE ALL CSR'S AND SP AT TIME OF ERROR 5. SPEED UP GALLOPING PATTERN BY EXECUTING OUT OF THE PAR'S 6. OPTIMIZE THE SHIFTING DIAGONAL PATTERN BY SHARING CODE WITH THE REFRESH PATTERN 7. ADD BRANCH GOBBLE PATTERN 8. ADD WRITE RECOVERY PATTERN 9. REPAIRED THE FOLLOWING BUGS A. UNDER APT FIELD SERVICE COMMANDS 1 & 2 B. UNDER APT USE OF ENVIRONMENTAL MODE DID NOT MEET APT SPEC C. FATAL PARITY ERRORS AT LOAD TIME THAT WERE CAUSED BY .BLKW STATEMENTS D. RESTART AT 202 CLEARS ERROR ACCOUNTING E. INTERMITTANT SINGLE BIT ERRORS IN CHECKBITS NOT DETECT F. ERROR COUNT WAS MOD 32K G. ODT HANGS AFTER 256 OPENS H. WRONG PC PRINTED ON SOME ERROR I. HIADRS LOADRS REG'S NOT CLEARE AFTER SIZING J. ACT MEMORY SIZE DEPENDENT BIT DEFINED WRONG K. UNABLE TO RUN WITHOUT A LINE CLOCK

	.SBTTL	OPERATIONAL SWITCH SETTINGS		
	.SBTTL	:SWITCH REGISTER DEFINITIONS		
	.SBTTL	:*		
	.SBTTL	:*	SWITCH	USE
	.SBTTL	:*	-----	-----
90	.SBTTL	:*	15	HALT ON ERROR
91	.SBTTL	:*	14	LOOP ON TEST
92	.SBTTL	:*	13	INHIBIT ERROR TYPEOUTS
93	.SBTTL	:*	12	INHIBIT RELOCATION
94	.SBTTL	:*	11	QUICK VERIFY / INHIBIT MARGIN TESTING
95	.SBTTL	:*	10	BELL ON ERROR
96	.SBTTL	:*	9	LOOP ON ERROR
97	.SBTTL	:*	8	HALT PROGRAM (UNRELOCATED & RESTORE LOADERS)
98	.SBTTL	:*	7	DETAILED ERROR REPORTS
99	.SBTTL	:*	6	PRINT CONFIGURATION MAP
100	.SBTTL	:*	5	LIMIT MAX ERRORS PER BANK
101	.SBTTL	:*	4	FAT TERMINAL (132 COLUMNS OR BETTER)
102	.SBTTL	:*	3	TEST MODE - SEE DOCUMENT
103	.SBTTL	:*	2	TEST MODE - SEE DOCUMENT
104	.SBTTL	:*	1	TEST MODE - SEE DOCUMENT
105	.SBTTL	:*	0	DETECT SINGLE BIT ERRORS
106	.SBTTL	:*		
107	.SBTTL	:*		
108	.SBTTL	:*		
109	.SBTTL	:*		
110	.SBTTL	:*		

113 000000
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131 163000
132 000001
133 000000

.ENABL ABS
.ENABL AMA
.DSABL GBL
:NOTE: CEMKAB.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
:THIS PROGRAM. ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
.MCALL SMACIT,..PUSH,..POP,..TAG,..BRAN,..EMIT,..EMITN,..EMITL,..EMITR
.MCALL .IFOPR,..IS,..GENBR,..OPADD,..OPSUB,CLEAR,SET,CLEARB,SETB
.MCALL RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
.MCALL IF,..OR,..IFARI,..LEAVE,..GOTO,OR,AND,THEN,ELSE,WHILE,CASE
.MCALL FOR,TO,DOWNTO,REPEAT,UNTIL,THRU,END,BEGIN
.MCALL \$\$END,LEAVE,JUMPTO,GOTO,PUSH,POP,LET
.MCALL .SIMPLE,..ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
.MCALL \$CALL,\$RETURN

.NLIST TTM ;I WANT FAT PAPER!
.LIST MC,SYM ;LIST MACRO CALLS, SYMBOL TABLE
.NLIST MD,CND,ME ;DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
:LST\$\$= 0 ;DEFINED TO LIST SUPERMAC EXPANSIONS
\$SWR= 163000 ;USE THESE SYSMAC SWITCHES
\$TN= 1 ;FIRST TEST NUMBER TO ONE(1)
SMACIT

DEFINE TRAPS

136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192

```

.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
:      IS 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           ;;TTY 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

```

```

193      104424      CACHOFF=104424      ;TURN OFF CACHE
194
195      104425      LOADCSR=104425      ;LOAD CORRECT CSR
196      104426      READCSR=104426      ;READ CORRECT CSR
197
198      104427      PERR01= 104427      ;PROGRAM DETECTED ERROR
199      104430      PERR02= 104430      ;PROGRAM DETECTED ERROR
200      104431      PERR03= 104431      ;PROGRAM DETECTED ERROR
201      104432      PERR04= 104432      ;PROGRAM DETECTED ERROR
202      104433      PERR07= 104433      ;PROGRAM DETECTED ERROR
203      104434      PERR10= 104434      ;PROGRAM DETECTED ERROR
204      104435      PERR11= 104435      ;PROGRAM DETECTED ERROR
205      104436      PERR12= 104436      ;PROGRAM DETECTED ERROR
206      104437      PERR13= 104437      ;PROGRAM DETECTED ERROR
207      104440      PERR14= 104440      ;PROGRAM DETECTED ERROR
208      104441      PERR15= 104441      ;PROGRAM DETECTED ERROR
209      104442      PERR16= 104442      ;PROGRAM DETECTED ERROR
210      104443      PERR17= 104443      ;PROGRAM DETECTED ERROR
211      104444      PERR20= 104444      ;PROGRAM DETECTED ERROR
212      104445      PERR21= 104445      ;PROGRAM DETECTED ERROR
213      104446      PERR22= 104446      ;PROGRAM DETECTED ERROR
214      104447      PERR23= 104447      ;PROGRAM DETECTED ERROR
215      104450      PERR24= 104450      ;PROGRAM DETECTED ERROR
216      104451      PERR25= 104451      ;PROGRAM DETECTED ERROR
217      104452      PERR26= 104452      ;PROGRAM DETECTED ERROR
218      104453      PERR27= 104453      ;PROGRAM DETECTED ERROR
219      104454      PERR30= 104454      ;PROGRAM DETECTED ERROR
220      104455      PERR31= 104455      ;PROGRAM DETECTED ERROR
221      104456      PERR32= 104456      ;PROGRAM DETECTED ERROR
222      104457      PERR33= 104457      ;PROGRAM DETECTED ERROR
223      104460      PERR34= 104460      ;PROGRAM DETECTED ERROR
224      104461      PERR35= 104461      ;PROGRAM DETECTED ERROR
225      104462      PERR36= 104462      ;PROGRAM DETECTED ERROR
226      104463      PERR37= 104463      ;PROGRAM DETECTED ERROR
227      104464      PERR40= 104464      ;PROGRAM DETECTED ERROR
228      104465      PERR41= 104465      ;PROGRAM DETECTED ERROR
229      104466      PERR42= 104466      ;PROGRAM DETECTED ERROR
230      104467      PERR43= 104467      ;PROGRAM DETECTED ERROR
231
232      104470      ECCDIS= 104470      ;DISABLE ECC ON ALL CSR'S
233      104471      ECC1DIS=104471      ;DISABLE ECC ON 1 SELECTED CSR
234      104472      ECCINIT=104472      ;INITIALIZE ALL MK11 CSR'S
235      104473      ECC1INIT=104473      ;INITIALIZE 1 SELECTED MK11 CSR
236      104474      CBCSR= 104474      ;WRITE GENERATED CHECKBITS IN ALL CSR'S
237      104475      CB1CSR= 104475      ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
238      104476      WASSBE= 104476      ;WAS THERE A SBE ON ANY CSR?
239      104477      WAS1SBE=104477      ;WAS THERE A SBE ON 1 SELECTED CSR?
240      104500      WASDBE= 104500      ;WAS THERE A DBE ON ANY CSR?
241      104501      WAS1DBE=104501      ;WAS THERE A DBE ON 1 SELECTED CSR?
242      104502      CLRCsr= 104502      ;CLEAR ALL CSR'S
243      104503      CLR1CSR=104503      ;CLEAR 1 SELECTED CSR
244      104504      CHKDIS= 104504      ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
245      104505      CHK1DIS=104505      ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
246      104506      ENASBE= 104506      ;ENABLE TRAPS ON SBE'S FROM ALL CSR'S
247      104507      ENA1SBE=104507      ;ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
248      104510      TSTREAD=104510      ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
249      104511      INVALID=104511      ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
    
```

DEFINE BASIC PDP11 STUFF

```

252          .SBTTL DEFINE BASIC PDP11 STUFF
253
254          ;*INITIAL ADDRESS OF THE STACK POINTER
255          002000 STACK= 2000          ;;FIRST ADDRESS OF THE STACK
256          002000 KERSTK= STACK        ;;KERNEL STACK
257          000740 SUPSTK= 740          ;;SUPERVISOR STACK
258          000700 USESTK= 700          ;;USER STACK
259          104000 ERROR=EMT            ;;BASIC DEFINITION OF ERROR CALL
260          000004 SCOPE=IOT            ;;BASIC DEFINITION OF SCOPE CALL
261          177776 PSW= 177776          ;;PROCESSOR STATUS WORD
262          ;STKLMT=177774              ;;STACK LIMIT REGISTER
263          ;PIRQ= 177772              ;;PROGRAM INTERRUPT REQUEST REGISTER
264          177570 DSWR= 177570         ;;HARDWARE SWITCH REGISTER
265          177570 DDISP= 177570        ;;HARDWARE DISPLAY REGISTER
266          177546 LKS= 177546         ;;LINE CLOCK (KW11-L) STATUS REGISTER
267
268          ;*MISCELLANEOUS DEFINITIONS
269          000011 HT= 11                ;;CODE FOR HORIZONTAL TAB
270          000012 LF= 12                ;;CODE LINE FEED
271          000015 CR= 15                ;;CODE CARRIAGE RETURN
272          000200 CRLF= 200            ;;CODE FOR CARRIAGE RETURN-LINE FEED
273
274          ;*GENERAL PURPOSE REGISTER DEFINITIONS
275          ;SP=R6                      ;;STACK POINTER
276          ;KSP=SP                    ;;KERNEL STACK POINTER
277          000006 SSP=SP               ;;SUPERVISOR STACK POINTER
278          000006 USP=SP               ;;USER STACK POINTER
279          ;PC=R7                      ;;PROGRAM COUNTER
280
281          ;*'SWITCH REGISTER' SWITCH DEFINITIONS
282          100000 SW15= 100000
283          040000 SW14= 40000
284          020000 SW13= 20000
285          010000 SW12= 10000
286          004000 SW11= 4000
287          002000 SW10= 2000
288          001000 SW9= 1000
289          000400 SW8= 400
290          000200 SW7= 200
291          000100 SW6= 100
292          000040 SW5= 40
293          000020 SW4= 20
294          000010 SW3= 10
295          000004 SW2= 4
296          000002 SW1= 2
297          000001 SW0= 1
298
299          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
300          100000 BIT15= 100000
301          040000 BIT14= 40000
302          020000 BIT13= 20000
303          010000 BIT12= 10000
304          004000 BIT11= 4000
305          002000 BIT10= 2000
306          001000 BIT9= 1000
307          000400 BIT8= 400
308          000200 BIT7= 200

```

```

309      000100      BIT6= 100
310      000040      BIT5= 40
311      000020      BIT4= 20
312      000010      BIT3= 10
313      000004      BIT2= 4
314      000002      BIT1= 2
315      000001      BIT0= 1
316
317      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
318      000004      ERRVEC= 4      ;:TIME OUT AND OTHER ERRORS
319      000010      RESVEC= 10     ;:RESERVED AND ILLEGAL INSTRUCTIONS
320      ;TBITVEC=14      ;:'T' BIT
321      ;TRTVEC= 14      ;:TRACE TRAP
322      ;BPTVEC= 14      ;:BREAKPOINT TRAP (BPT)
323      000020      IOTVEC= 20     ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
324      000024      PWRVEC= 24     ;:POWER FAIL
325      000030      EMTVEC= 30     ;:EMULATOR TRAP (EMT) **ERROR**
326      000034      TRAPVEC=34     ;:'TRAP' TRAP
327      000060      TKVEC= 60      ;:TTY KEYBOARD VECTOR
328      ;TPVEC= 64      ;:TTY PRINTER VECTOR
329      ;LKVEC= 100     ;:LINE CLOCK (KW11-L) VECTOR
330      000114      CACHVEC=114    ;:CACHE ERROR INTERRUPT VECTOR
331      000114      PARVEC=CACHVEC
332      ;PIRQVEC=240     ;:PROGRAM INTERRUPT REQUEST VECTOR
333      000250      MMVEC= 250     ;:MEMORY MANAGEMENT VECTOR
334      ;.SBTTL DEFINE CACHE REGISTERS
335      177740      LOADRS = 177740 ;:LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
336      177742      HIADRS = 177742 ;:UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
337      177744      MEMERR = 177744 ;:CACHE ERROR REGISTER
338      177746      CONTRL = 177746 ;:MEMORY CONTROL REGISTER
339      177750      MAINT = 177750  ;:MEMORY MAINTENANCE REGISTER
340      ;HITMIS = 177752 ;:HIT MISS REGISTER '1' IMPLIES HIT IN CACHE
341
342      ;.SBTTL DEFINE CPU REGISTERS
343      177760      SIZELO = 177760 ;:MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
344      ;SIZEHI = 177762 ;:TO GET TO THE LAST 32 WORDS OF MEMORY
345      ;:HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
346      ;:CURRENTLY ALL ZERO
347      177764      SYSTID = 177764 ;:SYSTEM ID REGISTER
348      177766      CPUERR = 177766 ;:CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
349
350      ;.SBTTL DEFINE MEMORY MANAGEMENT REGISTERS
351      ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
352      177572      MMR0= 177572
353      177574      MMR1= 177574
354      177576      MMR2= 177576
355      172516      MMR3= 172516
356
357      ;*USER "I" PAGE DESCRIPTOR REGISTERS
358      177600      UIPDR0= 177600
359      ;UIPDR1= 177602
360      ;UIPDR2= 177604
361      ;UIPDR3= 177606
362      ;UIPDR4= 177610
363      ;UIPDR5= 177612
364      ;UIPDR6= 177614
365      ;UIPDR7= 177616
    
```

```
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
```

```

; *USER 'D' PAGE DESCRIPTOR REGISTORS
:UDPDR0= 177620
:UDPDR1= 177622
:UDPDR2= 177624
:UDPDR3= 177626
:UDPDR4= 177630
:UDPDR5= 177632
:UDPDR6= 177634
177636 UDPDR7= 177636

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

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

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

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

; *SUPERVISOR 'I' PAGE ADDRESS REGISTERS
172240 SIPAR0= 172240
172242 SIPAR1= 172242
172244 SIPAR2= 172244
172246 SIPAR3= 172246 ;TEST AREA
```

```
423          :SIPAR4=          172250          :TEST AREA
424          :SIPAR5=          172252          :TEST AREA
425          :SIPAR6=          172254          :TEST AREA
426          :SIPAR7=          172256
427
428          :*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
429          172260 SDPAR0= 172260
430          :SDPAR1=          172262
431          :SDPAR2=          172264
432          :SDPAR3=          172266
433          :SDPAR4=          172270
434          :SDPAR5=          172272
435          :SDPAR6=          172274
436          172276 SDPAR7= 172276
437
438          :*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
439          172300 KIPDR0= 172300
440          :KIPDR1=          172302
441          :KIPDR2=          172304
442          :KIPDR3=          172306
443          :KIPDR4=          172310
444          :KIPDR5=          172312
445          :KIPDR6=          172314
446          :KIPDR7=          172316
447
448          :*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
449          :KDPDR0=          172320
450          :KDPDR1=          172322
451          :KDPDR2=          172324
452          :KDPDR3=          172326
453          :KDPDR4=          172330
454          :KDPDR5=          172332
455          :KDPDR6=          172334
456          172336 KDPDR7= 172336
457
458          :*KERNEL 'I' PAGE ADDRESS REGISTERS
459          172340 KIPAR0= 172340
460          :KIPAR1=          172342
461          :KIPAR2=          172344
462          :KIPAR3=          172346
463          172350 KIPAR4= 172350
464          172352 KIPAR5= 172352
465          172354 KIPAR6= 172354
466          172356 KIPAR7= 172356
467
468          :*KERNEL 'D' PAGE ADDRESS REGISTERS
469          172360 KDPAR0= 172360
470          :KDPAR1=          172362
471          :KDPAR2=          172364
472          :KDPAR3=          172366
473          :KDPAR4=          172370
474          :KDPAR5=          172372
475          :KDPAR6=          172374
476          172376 KDPAR7= 172376
477
```

```
480          .SBTTL DEFINE UNIBUS MAP REGISTERS
481          :*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
482          :*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
483          170200      MAPL0 = 170200
484          170202      MAPH0 = 170202
485          170204      MAPL1 = 170204
486          :MAPH1 = 170206
487          :MAPL2 = 170210
488          :MAPH2 = 170212
489          :MAPL3 = 170214
490          :MAPH3 = 170216
491          :MAPL4 = 170220
492          :MAPH4 = 170222
493          :MAPL5 = 170224
494          :MAPH5 = 170226
495          :MAPL6 = 170230
496          :MAPH6 = 170232
497          :MAPL7 = 170234
498          :MAPH7 = 170236
499          :MAPL10 = 170240
500          :MAPH10 = 170242
501          :MAPL11 = 170244
502          :MAPH11 = 170246
503          :MAPL12 = 170250
504          :MAPH12 = 170252
505          :MAPL13 = 170254
506          :MAPH13 = 170256
507          :MAPL14 = 170260
508          :MAPH14 = 170262
509          :MAPL15 = 170264
510          :MAPH15 = 170266
511          :MAPL16 = 170270
512          :MAPH16 = 170272
513          :MAPL17 = 170274
514          :MAPH17 = 170276
515          :MAPL20 = 170300
516          :MAPH20 = 170302
517          :MAPL21 = 170304
518          :MAPH21 = 170306
519          :MAPL22 = 170310
520          :MAPH22 = 170312
521          :MAPL23 = 170314
522          :MAPH23 = 170316
523          :MAPL24 = 170320
524          :MAPH24 = 170320
525          :MAPL25 = 170324
526          :MAPH25 = 170326
527          :MAPL26 = 170330
528          :MAPH26 = 170332
529          :MAPL27 = 170334
530          :MAPH27 = 170336
531          :MAPL30 = 170340
532          :MAPH30 = 170342
533          :MAPL31 = 170344
534          :MAPH31 = 170346
535          :MAPL32 = 170350
536          :MAPH32 = 170352
```

```
537      :MAPL33 = 170354
538      :MAPH33 = 170356
539      :MAPL34 = 170360
540      :MAPH34 = 170362
541      :MAPL35 = 170364
542      :MAPH35 = 170366
543      170370      MAPL36 = 170370
544      170372      MAPH36 = 170372
545      :MAPL37 = 170374
546      170376      MAPH37 = 170376
547
548      .SBTTL DEFINE SOFTWARE SWITCH & DISPLAY REGISTERS
549      000174      DISPREG=174
550      000176      SWREG= 176
551
552      .SBTTL DEFINE MK11 REGISTERS
553      172100      MK11ADD=172100
554
555      .SBTTL DEFINE PARAMETERS
556      060000      FIRST=60000      ;START OF THE 16K TEST PATTERN AREA
557      157776      LAST=157776      ;END OF THE 16K TEST PATTERN AREA
558      040000      SIZE=40000      ;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)
559      011661      SECONDS=11661      ;CONSTANT FOR LENGTH OF TIME FOR HUNG CPU TEST
560      .SBTTL DEFINE MASTER & SLAVE BITS
561      000020      MASTER=20
562      000040      SLAVE1=40
563      000100      SLAVE2=100
564      000200      SLAVE3=200
```

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

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

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

606
607
608
609
610
611
612
613
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

```
.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
```

656
657
658
659
660
661
662
663
664
665
666
667
668
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
707
708

```
.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
```

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

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

751
752
753
754
755
756
757
758
759
760
761
762
763
764
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

```

.SBTTL MACRO TYPOCS
***** TYPOCS *****
TYPOCS IS USED TO CHANGE A BINARY NUMBER TO AN OCTAL
NUMBER AND TYPE 1 TO 6 DIGITS
WITH OR WITHOUT LEADING ZEROS.

ARGUMENTS:
1) NUM NUMBER TO BE TYPED
2) REMARK ALLOWS A COMMENT TO BE MADE
3) N NUMBER OF DIGITS (1 TO 6) TO BE TYPED
4) Z BLANK=SUPPRESS LEADING ZEROS (TYPES SPACES)
NON-BLANK=TYPE LEADING ZEROS

ROUTINES REQUIRED
1) CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
2) TYPE AN ASCIZ STRING (.$TYPE)

EXAMPLES:
1) TYPOCS #12345,<TYPES '5'>,1
2) TYPOCS #004,<TYPES '04'>,2,X
3) TYPOCS #004,<TYPES '4'>,2
*****

.MACRO TYPOCS NUM,REMARK,N,Z
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM, -(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPOS ;;GO TYPE--OCTAL ASCII
.IF NB N
.BYTE N ;;TYPE N DIGIT(S)
.IFF
.BYTE 6 ;;TYPE 6 DIGITS
.ENDC
.IF NB Z
.BYTE 1 ;;TYPE LEADING ZEROS
.IFF
.BYTE 0 ;;SUPPRESS LEADING ZEROS
.ENDC
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPOCS
    
```

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

```
.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
:      MUNUS SIGN WILL BE TYPED.
:ARGUMENTS:
:1)   NUM   NUMBER TO BE TYPED
:2)   REMARK  ALLOWS A COMMENT TO BE MADE
:ROUTINES REQUIRED
:1)   CONVERT BINARY TO DECIMAL AND TYPE (.$TYPDEC)
:2)   TYPE AN ASCIZ STRING (.$TYPE)
:EXAMPLES
:1)   TYPDEC SIZE,<TYPE THE CONTENTS OF SIZE>
:2)   TYPDEC #-10.,<TYPE A MINUS TEN>
:*****

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

849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
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

```
.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
```

MACRO BMOV

906
907
908
909
910
911
912

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

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

```
.SBTTL MACRO MAP
***** MAP *****
: THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
: TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-157777)).
: 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
```

954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994

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

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

997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036

```
.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")  
: I USE THE SET4 AND RES4 MACROS AROUND CODE THAT I EXPECT TO TRAP TO 4  
: LIKE LOOKING FOR ALL POSSIBLE CSR'S AND ETC. WHENEVER CODE IS NOT  
: SURROUNDED BY SET4 AND RES4 THEN ANY TRAPS TO 4 WILL CAUSE AN ERROR  
: PRINTOUT THAT SAYS "UNEXPECTED TRAP TO 4" AND ALL THE ASSOCIATED REGISTER JUNK  
:*****
```

```
.MACRO SET4 ARG  
.NLIST  
.DSABL CRF  
.IIF DF LST$$ .LIST ME  
.ENABL CRF  
.LIST  
MOV ARG,4  
.DSABL CRF  
.IIF DF LST$$ .NLIST ME  
.ENABL CRF  
.ENDM SET4  
  
.MACRO RES4  
.NLIST  
.DSABL CRF  
.IIF DF LST$$ .LIST ME  
.ENABL CRF  
.LIST  
MOV #TIMEOUT,4  
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
```

1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060

```
.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
```

```

1063
1064
1065 000000 000000 000000
1066 000177
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086 000046
1087 000046 015000
1088 000052
1089 000052 040000
1090
1091 000024
1092 000024 000200
1093 000042
1094 000042 002000
1095 000044
1096 000044 075236
1097 000200
1098 000200 000437
1099 000202 000442
1100 000204 000446
1101 000300
1102 000300 005037 002752
1103 000304 000137 004352
1104 000310
      000310 012737 177777 002752
1105 000316 000137 004352
1106 000322
      000322 012737 177777 002756
1107 000330 000137 000300
1108 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

              ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$SEOP
              ;:2)SET LOC.52 TO INDICATE MEMORY SIZE DEPENDANT

              BIT14
              .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)
        BR      START4      ;'RUN MULTIPROCESSOR TESTS WITHOUT THE IIST' START

              =300

START1: CLR      RESTART
        JMP      START
START2: SET      RESTART
              JMP      START
START4: SET      NOIIST
              JMP      START1
              =STACK
    
```

```

MOV #-1,RESTART
MOV #-1,NOIIST
    
```

1111			.SBTTL	VARIABLES	INITIALIZED TO ZERO
1112			:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS		
1113			:*USED IN THE PROGRAM.		
1114	002000		\$CMTAG:		::START OF COMMON TAGS
1115	002000	000000	BADCSR:	0	::CSR DETERMINED TO HAVE AN ERROR BY BAD BOX ROUTINE
1116	002002	000000	SELONLY:	0	::SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
1117	002004	000000	MULTIONLY:	0	::RUN ONLY MULTIPOINT TESTS FLAG
1118	002006	000000	DIAGFLAG:	0	::SET FOR SHIFTING DIAGONAL TEST
1119	002010	000000	KAMIKAZE:	0	::SET FOR KAMIKAZE MODE TESTING
1120	002012	000000	SKIPKAMI:	0	::USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
1121			:NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER		
1122	002014	000	\$PATMAR:	.BYTE 0	::PATTERN NUMBER & MARGIN VALUE
1123	002015	000	\$BANK:	.BYTE 0	::BANK & SIGN = MARGIN INDICATOR
1124	002016	000	\$ERFLG:	.BYTE 0	::CONTAINS ERROR FLAG
1125	002017	000	\$ITEMB:	.BYTE 0	::CONTAINS ITEM CONTROL BYTE
1126	002020	000000	LASTERROR:	.WORD 0	::NUMBER OF ERRORS ON LAST PASS
1127	002022	000000	SBEFLAG:	.WORD 0	::A SINGLE BIT ERROR HAS OCCURED THIS PASS
1128	002024	000000	ERRPC:	.WORD 0	::CONTAINS PC OF ERROR FOR TYPEOUT
1129	002026	000000	BADPC:	.WORD 0	::CONTAINS PC OF ERROR
1130	002030	000000	ERRSP:	.WORD 0	::CONTAINS SP OF ERROR FOR TYPEOUT
1131	002032	000000	BADSP:	.WORD 0	::CONTAINS SP OF ERROR
1132	002034	000000	ERRPSW:	.WORD 0	::CONTAINS PSW OF ERROR FOR TYPEOUT
1133	002036	000000	BADPSW:	.WORD 0	::CONTAINS PSW OF ERROR
1134	002040	000000	ADDRESS:	.WORD 0	::CONTAINS ADDRESS OF 'BAD' DATA
1135	002042	000000	PADDRESS:	.WORD 0	::ADDRESS OF PARITY ERROR
1136	002044	000000 000000	PHYADD:	.WORD 0,0	::22 BIT PHYSICAL ADDRESS
1137	002050	000000	GOOD:	.WORD 0	::CONTAINS 'GOOD' DATA
1138	002052	000000	GOOD2:	.WORD 0	::CONTAINS 'GOOD2' DATA
1139	002054	000000	GOOD3:	.WORD 0	::CONTAINS 'GOOD3' DATA
1140	002056	000000	BAD:	.WORD 0	::CONTAINS 'BAD' DATA
1141	002060	000000	BAD2:	.WORD 0	::CONTAINS 'BAD2' DATA
1142	002062	000000	BAD3:	.WORD 0	::CONTAINS 'BAD3' DATA
1143	002064	000000	BADXOR:	.WORD 0	::XOR OF GOOD & BAD = BAD BITS!
1144	002066	000000	\$AUTO:	.WORD 0	::AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
1145	002070	000000	FATAL\$:	.WORD 0	::FATAL ERROR INDICATOR
1146	002072	000000	NEMCNT:	0	::NON-EXISTANT MEMORY COUNTER (HOLES)
1147	002074	000000	PARCNT:	0	::PARITY ERROR COUNTER
1148	002076	000000	PATERR:	0	::PATTERN ERROR COUNTER
1149	002100	000000	NOPAR:	0	::NO PARITY ERROR MODE INDICATOR
1150	002102	000000	NONEM:	0	::NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
1151	002104	000000	NOCLOCK:	0	::NO LINE CLOCK IS ON THIS SYSTEM
1152	002106	000000	BANK:	0	::MEMORY BANK UNDER TEST
1153	002110	000000	BANKINDEX:	0	::USED TO INDEX INTO CONFIG TABLE
1154	002112	000000	MARGIN:	0	::INDICATES WHICH MARGIN IS IN PROGRESS
1155	002114	000000	CPUBIT:	0	::CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
1156	002116	000000	MJT:	0	::MEMORY UNDER TEST FLAG
1157	002120	000000	PATTERN:	0	::PATTERN NUMBER UNDER TEST
1158	002122	000000	KPFLAG:	0	::BANK IS PROTECTED REGION OF MK11
1159	002124	000000	ACFLAG:	0	::BANK CAN BE ACCESSED BY THIS CPU
1160	002126	000000	MKFLAG:	0	::IF SET INDICATES MK11 UNDER TEST
1161	002130	000000	PFLAG:	0	::BANK IS IN PROGRAM SPACE
1162	002132	000000	RRFLAG:	0	::BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
1163	002134	000000	RLFLAG:	0	::PROGRAM IS RELOCATED FLAG
1164	002136	000000	BMFLAG:	0	::'BANK IS IDENTIFIED AS BAD MEMORY' FLAG
1165	002140	000000	TMFLAG:	0	::'TYPE OF MEMORY TO TEST' FLAG; 0 = MK11
1166	002142	000000	ABORTFLAG:	0	::'ABORT OCCURED' FLAG
1167	002144	000000 000000	CSR:	.WORD 0,0	::DATA TO OR FROM CSR

VARIABLES	INITIALIZED TO ZERO		
1217 002370	000000	OLDCACHE:0	:BACKED UP VALUE OF CACHE CONTROL REGISTER
1218 002372	000000	PARTHERE:0	:PARITY TRAPS SOMETIMES GO TO ADDRESS STORED HERE
1219 002374	000000	FSSTACK:0	:STACK SAVED HERE IF IN FIELD SERVICE MODE
1220 002376	000000	OLDMARGIN:0	:BACKED UP VALUE OF LAST MARGIN SETTING
1221 002400	000000	NEWBANK:0	:USED FOR RELOCATION TO A NEW BANK
1222 002402	000000	SOURCE: .WORD 0	:SOURCE OF DATA WORDS FOR CHECKBIT GENERATION SUBROUTINE
1223 002404	000000	CHECK: .WORD 0	:CHECKBITS TO BE LOADED INTO CSR
1224 002406	000000	PCBUMP: .WORD 0	:VALUE TO BUMP THE PC BY TO RECOVER AFTER A PARITY TRAP
1225 002410	000000	CSRFBANK:0	:FIRST BANK OWNED (AT LEAST IN PART) BY THIS CSR
1226 002412	000000	CSRFBANK:0	:LAST BANK OWNED (AT LEAST IN PART) BY THIS CSR
1227 002414	000000	CSRINC: 0	:VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR
1228 002416	000	CSRINDEX: .BYTE 0	:AN INDEX VARIABLE FOR A CSR LOOP
1229 002417	000	CSROUDEX: .BYTE 0	:AN INDEX VARIABLE FOR A CSR LOOP
1230 002420	000000	CSRLOOP:0	:LOOP CONTROL FOR CSR TESTING
1231 002422	000000	SIDE: 0	:INDEX VARIABLE FOR EACH SIDE (CONTROLLER) OF A CSR
1232 002424	000000	SUCCESS:0	:FLAG SET BY SUCCESSFUL TASK OR SUBROUTINE
1233 002426	000000	ZERO: 0	:FOR AID IN 'MOV' INSTRUCTIONS
1234 002430	000000	ZEROS: 0	:FOR AID IN 'MOV' INSTRUCTIONS
1235 002432	000000	SPECIAL:0	:FLAG FOR SPECIAL CASE OF NO INTERNAL INTERLEAVE
1236 002434	000000	TIME: 0	:SECONDS THAT BATTERIES SHOULD LAST
1237 002436	000000	NULLFLAG:0	:SET WHEN RUNNING NULL PATTERNS
1238 002440	000000	QVFLAG: 0	:FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
1239 002442	000000	ACTFLAG:0	:FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
1240 002444	000000	APTFLAG:0	:FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
1241 002446	000000	XXDPCHAIN:0	:FLAGS XXDP CHAIN MODE PROGRAMMING RULES
1242		:NOTE: THESE TWO BYTES MUST STAY TOGETHER	
1243 002450	000	\$NULL: .BYTE 0	:CONTAINS NULL CHARACTER FOR FILLS
1244 002451	000	\$FILLS: .BYTE 0	:CONTAINS # OF FILL CHARACTERS
1245 002452	000	\$TPFLG: .BYTE 0	: 'TERMINAL NOT AVAILABLE' FLAG
1246		.EVEN	
1247 002454	000000	\$ESCAPE:0	:ESCAPE ON ERROR ADDRESS
1248 002456	000000	EVEN: 0	:USED FOR ALTERNATE DATA PATTERNS
1249 002460	000000	STRIPES:0	:COUNTS DIAGONAL STRIPES
1250 002462	000000	COUNT: 0	:BACKED UP COPY OF STRIPES
1251 002464	000000	NOTAB: 0	:NO TABLE BEING PRINTED - NOW
1252 002466	000000	MJSIZE: 0	:SIZE OF MJ11 MEMORY IN K WORDS
1253 002470	000000	MKSIZE: 0	:SIZE OF MK11 MEMORY IN K WORDS
1254 002472	000000	TOOMANY:0	:FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
1255 002474	000000	NOECCFLAG:0	:ECC CAPABILITY DISABLED ON THE CONTROL PANEL
1256 002476	000000	GDSWITCH:0	:GOOD VALUE FOR SOME SWITCHES
1257 002500	000000	BDSWITCH:0	:BAD VALUE FOR SOME SWITCHES
1258 002502	000000	WRITEONLY:0	:FLAG TO PATTERNS TO WRITE ONLY
1259 002504	000000	READONLY:0	:FLAG TO PATTERNS TO READ ONLY
1260 002506	000000	TESTADD:0	:THE ADDRESS TO RUN CSR TESTS ON
1261 002510	000000	ALLCPUS:0	:CPUBIT'S FOR ALL CPU'S PRESENT
1262 002512	000000	HIBANK: 0	:HIGHEST BANK OF MEMORY
1263 002514	000000	HIACBANK: 0	:1ST INACCESSABLE BANK OF MEMORY
1264 002516	000000	UNITOP: 0	:HIGHEST ACCESSABLE BANK OF MEMORY THRU UNIBUS MAP
1265 002520	000000	STOPOK: 0	:FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
1266 002522	000000	APTCORE:0	:AMOUNT OF CORE MEMORY ACCORDING TO APT
1267 002524	000000	APTMO: 0	:AMOUNT OF MOS MEMORY ACCORDING TO APT
1268 002526	000000	NOFSMODE:0	:FLAG TO DISABLE FIELD SERVICE MODE
1269 002530	000000	NOERROR:0	: 'THIS IS NOT AN ERROR' FLAG
1270 002532	000000	LOADBANK:0	:BANK LOADERS ARE RELOCATED TO
1271 002534	000000	TEMP: 0	:USED FOR JUNK
1272 002536	000000	QUICK: 0	:QUICK STOP FLAG FOR APT POWER FAIL
1273 002540	000000	NOSCOPE:0	: 'NO SCOPE LOOP ALLOWED' FLAG

1274	002542	000000	F ^S SINFLAG:0	;'FIELD SERVICE - NO INTERNAL INTERLEAVE'' FLAG
1275	002544	000000	APTSIZE:0	;APT SIZING INFO FLAG
1276	002546	000000	FS7FLAG:0	;TRUE WHEN IN FIELD SERVICE COMMAND 7
1277	002550	000000	KI0: 0	;SAVED KIPAR0 TO PASS CORRECT VALUE TO SLAVE
1278	002552	000000	KI1: 0	;SAVED KIPAR1 TO PASS CORRECT VALUE TO SLAVE
1279	002554	000000	KI2: 0	;SAVED KIPAR2 TO PASS CORRECT VALUE TO SLAVE
1280	002556	000000	KI3: 0	;SAVED KIPAR3 TO PASS CORRECT VALUE TO SLAVE
1281	002560	000000	KI4: 0	;SAVED KIPAR4 TO PASS CORRECT VALUE TO SLAVE
1282	002562	000000	KI5: 0	;SAVED KIPAR5 TO PASS CORRECT VALUE TO SLAVE
1283	002564	000000	KI6: 0	;SAVED KIPAR6 TO PASS CORRECT VALUE TO SLAVE
1284	002566	000000	KI7: 0	;SAVED KIPAR7 TO PASS CORRECT VALUE TO SLAVE
1285	002570	000000	SLAERROR:0	;SLAVE HAD AN ERROR FLAG
1286	002572	000000	HUNGMSB:0	;USED FOR TIMES OVER 13 SECONDS
1287	002574	000000	HUNGTIME:0	;TIME TO ALLOW THE SYSTEM TO BE HUNG (MAX = 13. SEC)
1288	002576	000000	CONFERROR:0	;CONFIGURATION ERROR FLAG
1289	002600	000000	TWOCSRS:0	;TO FLAG LOADCSR TRAP TO LOAD TWO CSR'S
1290	002602	000000	I: 0	;USED FOR GENERAL PURPOSE INDEXING
1291	002604	000000	J: 0	;USED FOR GENERAL PURPOSE INDEXING

```

MULTIPOINT VARIABLES
1294 .SBTTL MULTIPOINT VARIABLES
1295 ;MULTIPOINT DIRECTORY
1296 ;PORTDIR:MASTER IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1297 ; SLAVE IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1298 ; SLAVE IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1299 ; SLAVE IIST ID NUMBER OR SYSTEM ID NUMBER IF IIST DISABLED BY OPERATOR
1300 002606 000000 000000 000000 PORTDIR:.WORD 0,0,0,0
1301 002614 000000
1301 002616 000000 000000 000000 TASK: .WORD 0,0,0,0
1302 002624 000000
1302 002626 000000 000000 000000 PTYFLAG:.WORD 0,0,0,0
1303 002634 000000
1303 002636 000000 000000 000000 PTYBUF: .WORD 0,0,0,0
1304 002644 000000
1304 002646 000000 000000 000000 PORTCOUNT:.WORD 0,0,0,0
1305 002654 000000
1305 002656 000000 000000 000000 SLFLAG: .WORD 0,0,0,0
1306 002664 000000
1306 002666 000000 MINDEX: 0 ;MASTER CPU'S INDEX # TO TABLES ABOVE
1307 002670 000000 LOCK: 0 ;MULTIPROCESSOR LOCK
1308 002672 000000 ACK: 0 ;ACKNOWLEDGE FLAG (USED FOR STARTING SLAVES)
1309 002674 000000 MASWR: 0 ;MASTER'S SWR
1310 002676 000000 MDISPLAY:0 ;MASTER CPU DISPLAY WHEN RUNNING MULTIPROCESSOR TESTS
1311 002700 $CMTGE: ;*END OF COMMON TAGS

```

Address	Hex	Dec	Label	Value	Description
1314					INITIALIZED TO NON ZERO
1315	002700	000001	CACHK:	1	:CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
1316	002702	000012	ERRMAX:	10	:MAX # OF ERRORS PER BANK WITH SW11
1317	002704	000031	SOBK:	25	:SOB CONSTANT
1318	002706	002000	KSTACK:	STACK	:STACK BEGINNING (USED TO KEEP SLAVES STACKS SEGREGATED)
1319	002710	000001	LOADHOME:	1	:HOME BANK OF LOADERS
1320	002712	177777	WORST:	177777	:SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
1321	002714	176543	SEEDHI:	176543	:WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
1322	002716	123456	SEEDLO:	123456	:WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
1323	002720	176543	MSEEDH:	176543	:MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
1324	002722	123456	MSEEDL:	123456	:MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
1325	002724	177777	HEADER:	177777	:USED TO PRINT HEADINGS ONLY ONCE
1326	002726	177777	ONES:	177777	:FOR AID IN 'MOV' INSTRUCTIONS
1327	002730	000000	SLAVES:	0	:# OF SLAVE CPU'S
1328	002732	000000	ONCE:	0	:FLAG FOR TYPING TITLE ONLY ONCE
1329	002734	000003	FLIPLOC:	3	:COUNTER FOR FLIPING DATA ON WORST CASE NOISE TEST
1330	002736	177500	IISTACR:	177500	:ADDRESS OF THE IIST ACR (WAS 166000)
1331	002740	177502	IISTADR:	177502	:ADDRESS OF THE IIST ADR (WAS 166002)
1332	002742	000260	IISTVEC:	260	:ADDRESS OF THE IIST VECTOR (WAS 170)
1333	002744	052525	SOFTPAT:	52525	:PATTERN FOR SOFT ERROR BACKGROUND TESTS
1334	002746	000000	\$LPADR:	.WORD 0	::CONTAINS SCOPE LOOP ADDRESS
1335	002750	000000	\$LPERR:	.WORD 0	::CONTAINS SCOPE RETURN FOR ERRORS
1336	002752	000000	RESTART:	0	:RESTART (START ADD 202) FLAG
1337	002754	000000	\$ERTTL:	.WORD 0	::CONTAINS TOTAL ERRORS
1338	002756	000000	NOIIST:	0	:NO IIST IS TO BE USED DURING MULTIPOINT TESTS
1339	002760	000000	PWLOCK:	0	:MULTIPROCESSOR LOCK FOR POWER FAILURE
1340					
1341					:***** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER *****
1342	002762	000377	BAKPAT:	.WORD 377	:BACKGROUND PATTERN *
1343	002764	177400	SWAPAT:	.WORD 177400	:SWAPPED BAKPAT *
1344					:*****
1345					
1346	002766	177570	SWR:	.WORD DSWR	::ADDRESS OF SWITCH REGISTER
1347	002770	177570	DISPLAY:	.WORD DDISP	::ADDRESS OF DISPLAY REGISTER
1348	002772	177560	\$TKS:	177560	::TTY KBD STATUS
1349	002774	177562	\$TKB:	177562	::TTY KBD BUFFER
1350	002776	177564	\$TPS:	177564	::TTY PRINTER STATUS REG. ADDRESS
1351	003000	177566	\$TPB:	177566	::TTY PRINTER BUFFER REG. ADDRESS
1352	003002	012	\$FILLC:	.BYTE 12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
1353	003003	207	\$BELL:	.ASCIZ <207><377><377>	::CODE FOR BELL
	003006	000			
1354	003007	077	\$QUES:	.ASCII /?/	::QUESTION MARK
1355	003010	015	\$CRLF:	.ASCII <15>	::CARRIAGE RETURN
1356	003011	012	\$LF:	.ASCIZ <12>	::LINE FEED
1357				.EVEN	
1358	003014	000000	MULTIPOINT:	0	:MULTIPROCESSING FLAG

1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395 003016 000170
 1398 004336 000000 000000 000000
 004344 000000 000000 000000
 1399
 1400

```

.SBTTL CONFIGURATION TABLE
:CONFIG:FIRST 16K CONFIGURATION WORDS (3 EACH)
      2ND 16K CONFIGURATION WORDS (3 EACH)
      167TH 16K CONFIGURATION WORDS (3 EACH)
:CONFIGURATION WORDS:
      LOW: BIT 0-2 INTERLEAVE FACTOR
           BIT 3 EXTERNAL INTERLEAVE
           BIT 4 MASTER CAN ACCESS (MEMORY EXISTS)
           BIT 5 SLAVE #1 CAN ACCESS (SHARED)
           BIT 6 SLAVE #2 CAN ACCESS (SHARED)
           BIT 7 SLAVE #3 CAN ACCESS (SHARED)
           BIT 8-10 BOX NUMBER (0-7)
           BIT 11 ERRORS PRESENT
           BIT 12 PROTECTED REGION OF A MK11 BOX
           BIT 13-14 MEMORY TYPE;0=MJ11;1=MK11
           BIT 15 PROTECTED (PROGRAM SPACE)
      MED: BIT 0-5 NOT USED
           BIT 6-8 MK11 CSR ADDRESS
           BIT 9-11 NOT USED
           BIT 12 INTERNAL INTERLEAVE
           BIT 13 'BACKGROUND PATTERN VALID' FLAG
           BIT 14 BANK SELECTED FOR TEST BY FIELD SERVICE MODE
           BIT 15 LOADERS HOME BANK
      HIGH: BIT 0 INDICATES BANK IS ADDRESSED BY CSR 172100
           BIT 1 INDICATES BANK IS ADDRESSED BY CSR 172104
           BIT 2 INDICATES BANK IS ADDRESSED BY CSR 172110
           BIT 3 INDICATES BANK IS ADDRESSED BY CSR 172114
           BIT 4 INDICATES BANK IS ADDRESSED BY CSR 172120
           BIT 5 INDICATES BANK IS ADDRESSED BY CSR 172124
           BIT 6 INDICATES BANK IS ADDRESSED BY CSR 172130
           BIT 7 INDICATES BANK IS ADDRESSED BY CSR 172134
           BIT 8-15 NUMBER OF ERRORS
:CONFIG: .REPT 170
:CONFIEND: .WORD 0,0,0,0,0,0 ;THESE WORDS SHOULD NEVER GET WRITTEN INTO
;IF THEY DO THE PROGRAM IS OVERFLOWING IT'S
;CONFIGURATION TABLE (SOFTWARE BUG)!
    
```

```

1402          .SBTTL ***** MAIN *****
1403 004352  START: SUBTST <<INITIALIZE VARIABLES TO ZERO>>
:*****
:*SUBTEST    INITIALIZE VARIABLES TO ZERO
:*****
1404 004352  SET      ACK          ;ACKNOWLEDGE THAT I HAVE STARTED (IN CASE I'M A SLAVE)
004352 012737 177777 002672          MOV  #-1,ACK
1405 004360 000005          RESET
1406 004362 013706 002706          MOV  KSTACK,SP      ;;SETUP THE STACK POINTER
1407 004366          IF MULTIPOINT EQ ONES AND CPUBIT NE #MASTER
004366 023737 003014 002726          CMP  MULTIPOINT,ONES
004374 001014          BNE  L0
004376 023727 002114 000020          CMP  CPUBIT,#MASTER
004404 001410          BEQ  L0
1408          ;WAIT FOR ACK TO CLEAR OR TIMEOUT & HALT
1409 004406 005004          CLR  R4
1410 004410          1$:      IF ACK IS TRUE
004410 005737 002672          TST  ACK
004414 001402          BEQ  L1
1411 004416 077404          SOB  R4,1$
1412 004420 000000          SHALT:  HALT          ;THE MASTER WOULD NOT ALLOW THIS SLAVE TO PROCEED
1413 004422          END ;OF IF ACK IS TRUE
004422          L1:::
1414 004422 000137 005752          JMP  BEGIN
1415 004426          END ;OF IF MULTIPOINT
004426          L0:::
1416 004426 012700 002000          1$:      MOV  #SCMTAG,R0      ;;FIRST LOCATION TO BE CLEARED
1417 004432 005020          CLR  (R0)+          ;;CLEAR MEMORY LOCATION
1418 004434 022700 002700          CMP  #SCMTGE,R0      ;;DONE?
1419 004440 001374          BNE  1$            ;LOOP BACK IF NO
1420 004442          SUBTST <<CLEAR NON-PROGRAM SPACE>>
:*****
:*SUBTEST    CLEAR NON-PROGRAM SPACE
:*****
1421          ;THIS ATTEMPS TO GET RID OF ANY PARITY ERRORS BY WRITING INTO
1422          ;EVERY LOCATION THAT IS NOT LOADED INTO BY THE PROGRAM OR ALLOCATED
1423          ;TO THE XXDP LOADERS
1424 004442 012737 000004 002100          MOV  #4,NOPAR      ;PARITY ACTION = COUNT & IGNORE
1425 004450 012700 001000          MOV  #1000,R0
1426 004454 012737 000015 177746          MOV  #15,CONTRL   ;DISABLE CACHE & ABORTS
1427 004462 012720 000000          9$:      MOV  #0,(R0)+      ;CLEAR THE STACK AREA
1428 004466 020027 002000          CMP  R0,#STACK
1429 004472 103773          BLO  9$
1430 004474 012700 117146          MOV  #END+2,R0
1431 004500 012720 000000          8$:      MOV  #0,(R0)+      ;CLEAR FROM THE END OF THE PROGRAM TO THE XXDP LOADE
1432 004504 020027 137776          CMP  R0,#157776-8192.
1433 004510 103773          BLO  8$
1434 004512 005037 002100          CLR  NOPAR        ;RESTORE DEFAULT PARITY ACTION
1435          ;
1436 004516 005237 002670          INC  LOCK          ;ONLY ALLOW ONE CPU AT A TIME TO GO THRU HERE
1437 004522          IF LOCK NE #1
004522 023727 002670 000001          CMP  LOCK,#1
004530 001402          BEQ  L2
1438 004532 000000          MHALT1: HALT
1439 004534 000776          BR   MHALT1
1440 004536          END ;OF IF LOCK
004536          L2:::
    
```

```

1443 004536          SUBTST <<INITIALIZE VARIABLES TO NON ZERO>>
:*****
:*SUBTEST          INITIALIZE VARIABLES TO NON ZERO
:*****
1444 004536          SET          WORST
1445 004536 012737 177777 002712          MOV          #-1,WORST
1446 004536 012737 000003 002734          SET          HEADER
1447 004536 012737 177777 002724          MOV          #-1,HEADER
1448 004560 012737 176543 002720          MOV          #176543,MSEEDH
1449 004566 012737 123456 002722          MOV          #123456,MSEEDL
1450 004574 013737 002720 002714          MOV          MSEEDH,SEEDHI ;PRIME THE RANDOM NUMBER GENERATOR
1451 004602 013737 002722 002716          MOV          MSEEDL,SEEDLO ;BOTH HIGH AND LOW WORDS
1452 004610 012737 000377 002762          MOV          #377,BAKPAT
1453 004616 012737 177400 002764          MOV          #177400,SWAPAT
1454 004624 012737 052525 002744          MOV          #52525,SOFTPAT
1455 004632 012737 005752 002746          MOV          #BEGIN,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1456 004640 012737 005752 002750          MOV          #BEGIN,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
1456 004646          SUBTST <<INITIALIZE VECTORS>>
:*****
:*SUBTEST          INITIALIZE VECTORS
:*****
1457 004646 012737 066232 000020          MOV          #$$SCOPE,IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1458 004654 012737 000340 000022          MOV          #340,IOTVEC+2 ;;LEVEL 7
1459 004662 012737 066662 000030          MOV          #$$ERROR,EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1460 004670 012737 000340 000032          MOV          #340,EMTVEC+2 ;;LEVEL 7
1461 004676 012737 075252 000034          MOV          #$$TRAP,TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1462 004704 012737 000340 000036          MOV          #340,TRAPVEC+2;LEVEL 7
1463 004712 012737 055160 000024          MOV          #$$PWRDN,PWRVEC ;;POWER FAILURE VECTOR
1464 004720 012737 000340 000026          MOV          #340,PWRVEC+2 ;;LEVEL 7
1465 004726 012737 036454 000114          MOV          #$$PARITY,PARVEC;GET READY FOR PARITY ERRORS
1466 004734 012737 000340 000116          MOV          #340,PARVEC+2
1467 004742 012737 036752 000010          MOV          #PDP1105,RESVEC;RESERVED INSTRUCTION TRAP
1468 004750 012737 000340 000012          MOV          #340,RESVEC+2
1469 004756 012737 036726 000004          MOV          #$$TIMEOUT,ERRVEC;SETUP TIMEOUT ERRORS
1470 004764 012737 000340 000006          MOV          #340,ERRVEC+2 ;;SET PRIORITY OF ERROR TRAPS
1471 004772 012737 036740 000250          MOV          #$$MMTRAP,MMVEC ;;VECTOR FOR MEMORY MANAGEMENT
1472 005000 012737 000340 000252          MOV          #340,MMVEC+2
1473 005006 012737 036764 000170          MOV          #$$IITRAP,170 ;;SET UP FOR UNEXPECTED IIST TRAPS
1474 005014 012737 000340 000172          MOV          #340,172
1475 005022 104423          CACHON          ;TURN CACHE ON
    
```

```

1479 005024
1480
1481
1482
1483 005024 005737 002752
      005024 001006
1484 005032 012700 003016
1485 005036 005020
1486 005040 022700 004336
1487 005044 001374
1488 005046
      005046
1489
1490 005046 012737 000020 002114
1491 005054
1492
1493
1494
1495 005054 012737 005110 000004
      005054
1496 005062 012737 177570 002766
1497 005070 012737 177570 002770
1498 005076 022777 177777 175662
      005076 001017
1499 005106 000403
1500 005110 012716 005116
1501 005114 000002
1502 005116 012737 036726 000004
      005116 005037 177766
1503 005130 012737 000176 002766
1504 005136 012737 000174 002770
1505 005144
      005144
1506
1507
1508 005144
1509 005144 012737 005160 000004
      005144
1510 005152 005737 177546
1511 005156 000403
1512 005160
    
```

```

SUBTST <<CLEAR THE CONFIGURATION TABLE>>
:*****
:*SUBTEST CLEAR THE CONFIGURATION TABLE
:*****
:THIS ZEROS (UNLESS WE STARTED AT ADDRESS 202) THE CONFIG TABLE
:WHICH IS FULLY DISCRIBED AT LOCATION 'CONFIG'.
.ENABLE LSB
IF RESTART IS FALSE
TST RESTART
BNE L3
1$: MOV #CONFIG,R0
    CLR (R0)+
    CMP #CONFIEND,R0
    BNE 1$
    END ;OF IF RESTART
L3:*****
.DSABL LSB
MOV #MASTER,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 #3$,4
.DSABL CRF
MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWITCH REGISTER
MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
IF #-1 EQ @SWR ;:IF NO TRAP FROM REFERENCE TO @SWR AND @SWR = #-1
CMP #-1,@SWR
BNE L4
3$: BR 2$ ;:BRANCH IF NO TIMEOUT
    MOV #2$,(SP) ;:SET UP FOR TRAP RETURN
    RTI
2$: RES4 ;:RESET TRAPS TO 4 TO DEFAULT
    MOV #TIMEOUT,4
    CLR CPUERR ;:CLEAR OUT THE CPU ERROR REGISTER BITS
    ;:THAT A EXPECTED TRAP COULD HAVE SET
.DSABL CRF
MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY
END ;OF IF #-1
L4:*****
.DSABL LSB
SUBTST <<FIND OUT IF ANY LINE CLOCK IS PRESENT>>
:*****
:*SUBTEST FIND OUT IF ANY LINE CLOCK IS PRESENT
:*****
SET4 #4$
MOV #4$,4
.DSABL CRF
TST LKS
BR 5$ ;:IF NO TRAP SKIP
4$: SET NOCLOCK ;:SET NO L'NE CLOCK FLAG
    
```

```
1513 005160 012737 177777 002104
      005166 012737 036726 000004
      005174 005037 177766
      SS: RES4 ;RESTORE TRAPS TO 4 TO DFFAULT MOV #-1,NOCLOCK
          MOV #TIMEOUT,4 ;CLEAR OUT THE CPU ERROR REGISTER BITS
          CLR CPUERR ;THAT A EXPECTED TRAP COULD HAVE SET
          .DSABL CRF
```

```

1516 005200          SUBAAB: SUBTST <<SETUP ACT, APT, & XXDP>>
:*****
:*SUBTEST          SETUP ACT, APT, & XXDP
:*****
1517                ;THIS SETS UP A BUNCH OF FLAGS TO TELL THE PROGRAM EVERYTHING
1518                ;IT CARES TO KNOW ABOUT APT, ACT, & XXDP.
1519 005200 005037 075142 CLR $PASS ;CLEAR PASS COUNT
1520 005204 132737 000040 075155 IFB #BIT5 SET.IN $ENVM
                                BITB #BIT5,$ENVM
                                BEQ L5
1521 005214 001403          SET $TPFLG ;INDICATE NO TERMINAL
                                MOV #-1,$TPFLG
1522 005214 012737 177777 002452 END ;OF IFB #BIT5
                                L5:*****
1523 005222          IFB #BIT7 SET.IN $ENVM
                                BITB #BIT7,$ENVM
                                BEQ L6
1524 005232          SET APTSIZE
                                MOV #-1,APTSIZE
1525 005240 012737 177777 002544 MOV #$$SWREG,SWR ;USE APT SWR
1526 005246 012737 075156 002766 END ;OF IFB #BIT7
                                L6:*****
1527 005246          IFB $ENV EQ #1
                                CMPB $ENV,#1
                                BNE L7
1528 005256          SET APTFLAG,QVFLAG,$AUTO,QUICK
                                MOV #-1,APTFLAG
                                MOV #-1,QVFLAG
                                MOV #-1,$AUTO
                                MOV #-1,QUICK
1529 005306 012737 043640 000024 MOV #APTDOWN,PWRVEC
1530 005314          ELSE
                                BR L10
1531 005316          IF 42 NE #STACK AND 42 NE #0
                                L7:*****
                                CMP 42,#STACK
                                BEQ L11
                                TST 42
                                BEQ L11
1532 005334          SET QVFLAG,$AUTO
                                MOV #-1,QVFLAG
                                MOV #-1,$AUTO
1533 005350          IF 42 EQ #SENDAD
                                CMP 42,#SENDAD
                                BNE L12
1534 005360          SET ACTFLAG
                                MOV #-1,ACTFLAG
1535 005366          ELSE
                                BR L13
1536 005370          SET XXDPCHAIN
                                L12:*****
                                MOV #-1,XXDPCHAIN
1537 005376          END ;OF IF 42
                                L13:*****
1538 005376          END ;OF IF 42
                                L11:*****
1539 005376          END ;OF IFB $ENV
    
```

005376

L10:.....



1542 005376

SUBTST <<INITIALIZE PATTERNS>>

 :*SUBTEST INITIALIZE PATTERNS

1543
 1544
 1545
 1546

: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.
 IF APTFLAG IS TRUE

1547 005376 005737 002444
 005376 001432
 005402 012700 075220
 1548 005404 012001
 1549 005410 012703 020124
 1550 005412 004737 005472
 1551 005416 012001
 1552 005422 004737 005472
 1553 005424 012001
 1554 005430 012703 021402
 1555 005432 004737 005472
 1556 005436 012001
 1557 005442 004737 005472
 1558 005444 012001
 1559 005450 012703 021600
 1560 005452 004737 005472
 1561 005456 012001
 1562 005462 004737 005472
 1563 005464 012001
 1564 005470 004737 005472

MOV #SDDW0,R0
 MOV (R0)+,R1
 MOV #MKCSRT,R3
 CALL PATPLUG
 MOV (R0)+,R1
 CALL PATPLUG
 MOV (R0)+,R1
 MOV #MKPAT,R3
 CALL PATPLUG
 MOV (R0)+,R1
 CALL PATPLUG
 MOV #MJPAT,R3
 CALL PATPLUG
 MOV (R0)+,R1
 CALL PATPLUG
 END ;OF IF APTFLAG IS TRUE

TST APTFLAG
 BEQ L14

1565 005470 000420
 1566
 1567 005472

BR SUBAAA

L14:::~::~

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

 :*SUBTEST SUBR PLUG IN NULL PATTERNS

1568 005472 012737 000001 002602
 005472 005500
 1569 005500 006001
 1570 005502 103402
 005502 012713 025772
 1571 005504 012713 025772
 1572 005510 005510
 1573 005510 062703 000002
 1574 005514 005514 005237 002602
 005520 023727 002602 000020
 005526 003764
 005530
 1575 005530 000207

FOR I := #1 TO #16.

ROR R1
 ON.NOERROR ;IF CARRY CLEAR
 MOV #MT0999,(R3)
 END ;OF ON.ERROR
 ADD #2,R3
 END ;OF FOR

MOV #1,I
 B0:::~::~

BCS L15

L15:::~::~

INC I
 CMP I,#16.
 BLE B0

E0:::~::~

RETURN


```

1603 005662 012777 100000 175046      .DSABL CRF
1604 005670 017700 175042      MOV #BIT15,@IISTACR ;IS THERE AN IIST?
1605 005674 072027 177770      MOV @IISTACR,R0     ;YES - GET ID
1606 005700 042700 177774      ASH #-8.,R0
1607 005704 010037 002606      BIC #^C3,R0
1608 005710 005777 175024      MOV R0,PORTDIR     ;IDENTIFY MASTER
1609 005714      TST @IISTADR       ;SEE IF 2ND IIST REGISTER TRAPS
1610 005714      END ;OF IF NOIIST
1610 005714      SET MULTIPOINT    ;MULTIPOINTING!
1611 005722 012737 177777 003014      MOV KSTACK,SP      ;MULTIPOINTING!
1612 005726 013706 002706 2$:      RES4                ;MULTIPOINTING!
1612 005726 012737 036726 000004      MOV #TIMEOUT,4     ;MULTIPOINTING!
1612 005734 005037 177766      CLR CPUERR         ;MULTIPOINTING!
1613 005740      .DSABL CRF
1613 005740 012737 177777 002732      SET ONCE           ;MULTIPOINTING!
1614 005746 005037 002670      CLR LOCK           ;MULTIPOINTING!
1615      .DSABL LSB
    
```

L23:;;;;;;

MOV #-1,MULTIPOINT

;RESET TRAPS TO 4 TO DEFAULT

;CLEAR OUT THE CPU ERROR REGISTER BITS
 ;THAT A EXPECTED TRAP COULD HAVE SET

MOV #-1,ONCE

```
1618 005752          BEGIN: SUBTST <<SLAVES START HERE>>
:*****
:*SUBTEST          SLAVES START HERE
:*****
1619 005752 000240          NOP
1620 005754 005237 002670          INC      LOCK          ;ONLY ONE CPU AT A TIME IS ALLOWED HERE
1621 005760          IF LOCK NE #1
      005760 023727 002670 000001          CMP LOCK,#1
      005766 001402          BEQ L24
1622 005770 000000          MHALT2: HALT          ;MULTIPLE CPU'S EXECUTING SAME CODE
1623 005772 000776          BR      MHALT2
1624 005774          END ;OF IF LOCK
      005774          IF CPUBIT NE #MASTER          ;IF I'M A SLAVE L24:*****
1625 005774 023727 002114 000020          CALL SLAVEMAP          CMP CPUBIT,#MASTER
      006002 001402          BEQ L25
1626 006004 004737 061244          CALL SLAVEMAP
1627 006010          END ;OF IF CPUBIT
      006010          L25:*****
```

```

1630 006010          SUBTST <<DETERMINE MK11 ACCESS METHOD>>
:*****
:*SUBTEST          DETERMINE MK11 ACCESS METHOD
:*****
1631                ;DETERMINE HOW WE ACCESS THE MK11 CSR'S.
1632                ;IF I CAN'T ACCESS THE CSR'S NORMALLY THEN I ASSUME THAT I MUST
1633                ;RELOCATE THRU THE UNIBUS MAP SO I SET THE 'CSR'S RELOCATED' FLAG (CSRREL)
1634                ;NOTE: I DON'T REALLY KNOW AT THIS POINT IF THERE IS ANY MK11 MEMORY YET
1635                ; I ONLY DETERMINE HOW I WILL HAVE TO ACCESS IT IF THERE IS ANY!
1636 006010          SET4   #5$           ;TRAPS TO 4 GOTO 5$
          006010 012737 006036 000004    MOV   #5$,4
          .DSABL CRF
1637 006016 012700 000010    MOV   #8,R0           ;TEST FOR 8 CSR'S
1638 006022 012702 172100    MOV   #MK11ADD,R2    ;R2 = CSR ADDRESS
1639 006026 005712          4$: TST   (R2)           ;ANSWER RO TRAP
1640 006030 005037 002254    CLR   CSRREL        ;ANSWER - CSR'S NOT RELOCATED
1641 006034 000410          BR    6$           ;BRANCH - OUT
1642 006036 062706 000004    5$: ADD  #4,SP       ;FIX STACK FROM TRAP
1643 006042 062702 000004    ADD  #4,R2         ;NEXT CSR = LAST CSR + 4
1644 006046 077011          SOB  R0,4$        ;LOOP TILL DONE
1645 006050          SET   CSRREL
          006050 012737 177777 002254    MOV  #-1,CSRREL
1646 006056          6$: RES4          ;RESET TRAPS TO 4 TO DEFAULT
          006056 012737 036726 000004    MOV  #TIMEOUT,4
          006064 005037 177766          CLR   CPUERR       ;CLEAR OUT THE CPU ERROR REGISTER BITS
          ;THAT A EXPECTED TRAP COULD HAVE SET
          .DSABL CRF
1647 006070          SUBTST <<SETUP USER & SUPERVISOR STACK>>
:*****
:*SUBTEST          SETUP USER & SUPERVISOR STACK
:*****
1648 006070 104421          DEENERGIZE        ;TURN OFF MEMORY MANAGEMENT
1649
1650                ;SET PREVIOUS MODE TO SUPERVISOR
1651 006072 042737 030000 177776    BIC  #BIT13!BIT12,PSW
1652 006100 052737 010000 177776    BIS  #BIT12,PSW
1653
1654 006106          PUSH  #SUPSTK
          006106 012746 000740          MOV  #SUPSTK,-(SP)
1655 006112 006606          MTPJ  SSP
1656
1657                ;SET PREVIOUS MODE TO USER
1658 006114 052737 030000 177776    BIS  #BIT13!BIT12,PSW
1659
1660 006122          PUSH  #USESTK
          006122 012746 000700          MOV  #USESTK,-(SP)
1661 006126 006606          MTPJ  USP
    
```

```

1664 006130          SUBTST <<GET SOFTWARE SWITCH REGISTER IF NECESSARY>>
:*****
:*SUBTEST          GET SOFTWARE SWITCH REGISTER IF NECESSARY
:*****
1665 006130          IF CPUBIT EQ #MASTER AND $AUTO IS FALSE ;IF MASTER & NOT (APT OR ACT)
006130 023727 002114 000020          CMP CPUBIT,#MASTER
006136 001010          BNE L26
006140 005737 002066          TST $AUTO
006144 001005          BNE L26
1666 006146          IF SWR EQ #SWREG          ;IF SOFTWARE SWITCH REG SELECTED
006146 023727 002766 000176          CMP SWR,#SWREG
006154 001001          BNE L27
1667 006156 104407          GTSWR          ;;GET SOFT-SWR SETTINGS
1668 006160          END ;OF IF SWR
1669 006160          END ;OF IF CPUBIT          L27:;;;;;
006160          L26:;;;;;
1670
1671 006160 012737 000020 172516      MOV #20,MMR3          ;SET 22 BIT MODE
1672 006166 005037 177750          CLR MAINT          ;CLEAR MARGINS
1673
1674 006172          SUBTST <<SETUP UNIBUS MAP>>
:*****
:*SUBTEST          SETUP UNIBUS MAP
:*****
1675 006172 012737 000077 170372      MOV #77,MAPH36          ;MAP36 IS USED TO ACCESS MK11 CSR'S ON PDP11-70'S
1676 006200 012737 160000 170370      MOV #160000,MAPL36      ;THAT DO NOT HAVE THE MULTIPROCESSOR MODIFICATIONS.
1677 006206          CLEAR MAPH0,MAPLO          ;MAP0 IS USED FOR APT TO LOOK AT THE MAILBOX THRU.
006206 005037 170202          CLR MAPH0
006212 005037 170200          CLR MAPLO
1678 006216 004737 042026          CALL LOWMAP          ;SETUP LOWER UNIBUS MAP
1679 006222 052737 000040 172516      BIS #BITS,MMR3          ;ENERGIZE UNIBUS MAP
1680
1681 006230          SUBTST <<GET MEMORY MANAGEMENT READY>>
:*****
:*SUBTEST          GET MEMORY MANAGEMENT READY
:*****
1682 006230          IF CPUBIT EQ #MASTER
006230 023727 002114 000020          CMP CPUBIT,#MASTER
006236 001001          BNE L30
1683 006240 104422          KMAP          ;MAP KERNEL SPACE 1 TO 1
1684 006242          END ;OF IF CPUBIT
1685 006242          MAP          ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1
006242 010346          MOV R3,-(SP)          L30:;;;;;
006244 012703 000170          MOV #120,R3
006250 004737 040700          CALL MAPPER
.DSABL CRF
006254 012603          MOV (SP)+,R3
1686 006256 104420          ENERGIZE          ;TURN ON MEMORY MANAGEMENT
    
```

1689 006260

NEWTST <<ANALIZE CONFIGURATION>>

 :*TEST 1 ANALIZE CONFIGURATION

1690 006260 000004
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702
 1703 006262 005037 002256
 1704 006266 013746 177746
 1705 006272 012737 006666 000004
 1706 006300 010637 006754
 1707 006304 006337 002334
 1708 006310 006337 002330
 1709 006314 104426
 1710 006316 052737 000001 002334
 1711 006324 005001
 1712 006326 013700 002146
 1713 006332 072027 177765
 1714 006336 042700 177770
 1715 006342 032700 000001
 006346 001403
 006350 052737 000400 002334
 006356
 1716 006356 010046
 006360 006316
 006362 004737 006406
 1717 006366 006756
 1718 006370 006760
 1719 006372 006760
 1720 006374 006766
 1721 006376 006766
 1722 006400 006774
 1723 006402 006766
 1724 006404 006774
 1725 006406 062616
 006410 013646
 006412 004736
 1726 006414 010137 002264
 1727 006420 004737 017736

TST1: SCOPE
 :THIS CODE DOES THE FOLLOWING FOR MK11 CSR'S
 :1. FINDS ANY EXISTING CSR'S - SETTING BITS IN LOC 'MKCSRS' FOR
 : EACH ONE PRESENT.
 :2. SETS A BIT IN LOC 'MKCSRS' FOR ANY MK11 THAT IS INTERNALLY INTERLEAVED
 :3. DETERMINES THE INTERLEAVE FACTOR & STARTING ADDRESS. USING THIS
 : WE DETERMINE THE FIRST & LAST BANK INCLUDED IN THIS CSR. USING THAT
 : THE CONFIGURATION TABLE IS UPDATED TO
 : A. IDENTIFY BANKS WITHIN THE RANGE AS MK11 MEMORY.
 : B. IDENTIFY BANKS WITHIN THE RANGE WITH INTERLEAVE FACTOR.
 : C. IF NO EXTERNAL INTERLEAVE IDENTIFY BANKS WITHIN THE RANGE
 : AS CONTROLLED BY THIS CSR
 : D. IDENTIFY BANKS WITHIN THE RANGE BY A SPECIFIC BIT POSITION
 : FOR THIS CSR
 CLR CSRNO ;1ST CSR IS 0
 PUSH CONTRL ;SAVE CACHE STATUS
 MOV CONTRL,-(SP)
 SET4 #CONFGT ;TRAPS TO 4 GOTO CONFGT
 MOV #CONFGT,4
 .DSABL CRF
 MOV SP,CONFSP ;SAVE STACK
 CONFG1: ASL MKCSRS
 ASL PENDBOX
 READCSR ;TRAP IF IT DOES NOT EXIST
 BIS #1,MKCSRS ;MARK CSR AS PRESENT
 CLR R1
 MOV CSR+2,R0
 ASH #-11,R0
 BIC #^C7,R0
 IF #BIT0 SET.IN R0 THEN LET MKCSRS := MKCSRS SET.BY #BIT8
 BIT #BIT0,R0
 BEQ L31
 BIS #BIT8,MKCSRS
 L31:;;;;;
 CASE R0
 MOV RO,-(SP)
 ASL @SP
 JSR PC,L32
 INTER0
 INTER1
 INTER2
 INTER3
 INTER4
 INTER5
 INTER6
 INTER7
 END ;OF CASE R0
 L32:;;;;;
 ADD (SP)+,@SP
 MOV @SP+,-(SP)
 JSR PC,@SP+
 MOV R1,INTERL ;SET INTERLEAVE FACTOR
 CALL GETSIZE

```

1728                                     ;THE NEXT 6 LINES SAVE THE READ STATUS OF THE BOX CAPACITY
1729 006424 013701 002256                MOV     CSRNO,R1
1730 006430 006201                       ASR     R1
1731 006432 013761 002144 002154        MOV     CSR,CSR0(R1)
1732 006440 013761 002146 002174        MOV     CSR+2,CSR2(R1)
1733 006446 013761 002260 002214        MOV     FIRSTB,CSRFB(R1)
1734 006454                                IF FIRSTB GT #167
                                006454 023727 002260 000167                CMP FIRSTB,#167
                                006462 003405                                BLE L33
1735 006464 052737 000001 002330        BIS #BIT0,PENDBOX
1736 006472 000137 006712                JMP     CONFIG5
1737 006476                                END ;OF IF FIRSTB
                                L33:::
1738 006476 013761 002262 002234        MOV     LASTB,CSRLB(R1)
1739 006504 013701 002260                MOV     FIRSTB,R1
1740 006510 070127 000006                MUL     #6,R1
                                ;R1 = BANK * 6
1741 006514 010137 002110                MOV     R1,BANKINDEX
1742 006520 013701 002110                MOV     BANKINDEX,R1
                                ;GET NEXT INDEX (SAME FIRST TIME)
1743 006524 053761 002264 003016        BIS     INTERL,CONFIG(R1)
                                ;SETUP INTERLEAVE BITS
1744 006532 052761 020000 003016        BIS     #BIT13,CONFIG(R1)
                                ;IDENTIFY AS MK11
1745 006540 042761 040000 003016        BIC     #BIT14,CONFIG(R1)
1746 006546                                IF #BIT8 SET.IN MKCSRS
                                006546 032737 000400 002334                BIT #BIT8,MKCSRS
                                006554 001403                                BEQ L34
1747 006556 052761 010000 003020        BIS     #BIT12,CONFIG+2(R1)
1748 006564                                END ;OF IF #BIT8
                                L34:::
1749 006564 013700 002256                MOV     CSRNO,R0
1750 006570 072027 177776                ASH     #-2,R0
                                ;GET CSR LSB'S
1751 006574                                IF #BIT13!BIT12 SET.IN CSR+2
                                ;MAKE (0-7)
                                ;ANY EXTERNAL INTERLEAVE?
                                006574 032737 030000 002146                BIT #BIT13!BIT12,CSR+2
                                006602 001404                                BEQ L35
1752 006604 052761 000010 003016        BIS     #BIT3,CONFIG(R1)
                                ;IDENTIFY IT
1753 006612                                ELSE
                                006612 000406                                BR L36
                                L35:::
1754 006614                                PUSH R0
                                ;MOVE TO BITS 10-8
                                006614 010046                                MOV R0,-(SP)
1755 006616 072027 000010                ASH     #8,R0
                                ;IDENTIFY AS CSR & BOX NUMBER
1756 006622 050061 003016                BIS     R0,CONFIG(R1)
1757 006626                                POP R0
                                MOV (SP)+,R0
                                006626 012600
1758 006630                                END ;OF IF #BIT13!BIT12
                                L36:::
1759 006630 012702 000001                MOV     #1,R2
1760 006634 072200                ASH     R0,R2
                                ;R2 = 1,2,4,10,20,40,100, OR 200
1761 006636 050261 003022                BIS     R2,CONFIG+4(R1)
                                ;SETUP CSR WORD
1762 006642 062737 000006 002110        ADD     #6,BANKINDEX
                                ;NEXT TABLE ENTRY
1763 006650 005237 002260                INC     FIRSTB
1764 006654 023737 002260 002262        CMP     FIRSTB,LASTB
                                ;DONE?
1765 006662 101716                BLOS   CONFIG3
                                ;NO - LOOP
1766 006664 000412                BR     CONFIG5
1767 006666 013706 006754                MOV     CONFIGSP,SP
                                ;FIX STACK FROM MULTIPLE TRAP
1768 006672                                IF #BIT4 OFF.IN CPUERR
                                006672 032737 000020 177766                BIT #BIT4,CPUERR
                                006700 001004                                BNE L37
1769 006702                                LET GOOD := #BIT4

```

```

006702 012737 000020 002050                                MOV #BIT4,GOOD
1770 006710 104037                                ERROR +37
1771 006712                                END ;OF IF CPUERR
006712                                L37:~~~~~
1772 006712 062737 000004 002256 CONFIG5: ADD #4,CSRNO ;BUMP CSR
1773 006720                                IF CSRNO NE #40 THEN JUMPTO CONFIG1
006720 023727 002256 000040                                CMP CSRNO,#40
006726 001402                                BEQ L40
006730 000137 006304                                JMP CONFIG1
006734                                L40:~~~~~
1774 006734                                RES4 ;RESET TRAPS TO 4 TO DEFAULT
006734 012737 036726 000004 MOV #TIMEOUT,4
006742 005037 177766 CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
;THAT A EXPECTED TRAP COULD HAVE SET
006746 .DSABL CRF
1775 006746 012637 177746 POP CONTRL ;RESTORE CACHE STATUS
006746 000413 BR SUBAAS MOV (SP)+,CONTRL
1776 006752 000000 CONFSP: 0 ;STACK SAVED HERE!
1777 006754 000000
1778
1779 ;TOTAL INTERLEAVE (0, 2-1, 4-1, OR 8-1)
1780 006756 000207 INTER0: RETURN ;NO INTERLEAVE
1781
1782 006760 INTER1:
1783 006760 012701 000001 INTER2: MOV #1,R1 ;2 WAY INTERLEAVE
1784 006764 000207 RETURN
1785
1786 006766 INTER3:
1787 006766 INTER4:
1788 006766 012701 000002 INTER6: MOV #2,R1 ;4 WAY INTERLEAVE
1789 006772 000207 RETURN
1790
1791 006774 INTER5:
1792 006774 012701 000004 INTER7: MOV #4,R1 ;8 WAY INTERLEAVE
1793 007000 000207 RETURN
  
```

```

1795 007002 104472
1796 007004
SUBAAS: ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
NEWST <<TEST BANK 0 ACCESSES>>
:*****
:*TEST 2 TEST BANK 0 ACCESSES
:*****
1797 007004 000004
1798
1799
1800
1801
1802
1803 007006 005037 002074
1804 007012 012737 000001 002100
1805 007020 005037 002072
1806 007024 012737 000001 002102
1807 007032
007032 012737 036646 000004
1808 007040 005000
1809 007042 012701 040000
1810 007046 104424
1811 007050 005720
1812 007052 077102
1813 007054 104423
1814
1815 007056 005737 002074
1816 007062 001403
1817 007064
007064 005237 002070
007070 104003
1818 007072 005737 002072
1819 007076 001406
1820 007100 162737 000002 002040
1821 007106
007106 005237 002070
007112 104004
1822 007114 053737 002114 003016
1823 007122
007122 012737 036726 000004
007130 005037 177766
1824
1825 007134
SUBTST <<ENABLE MK11 FOR CORRECT TRAPS>>
:*****
:*SUBTEST ENABLE MK11 FOR CORRECT TRAPS
:*****
1826 007134
007134 032777 000001 173624
007142 001003
007144 005737 002442
007150 001402
007152
1827 007152 104506
ENASBE ;TRAP ON SINGLE BIT ERRORS
BIT #SW0,@SWR
BNE L41
TST ACTFLAG
BEQ L42
L41:.....

```

1828 007154
007154 000401
007156
1829 007156 104472
1830 007160
007160

ELSE

ECCINIT
END ;OF IF #SWO

BR L43
L42:.....
;TRAP ON DOUBLE BIT ERRORS (NORMAL)
L43:.....

```

1833 007160
      007160 000004
1834
1835
1836
1837
1838
1839
1840
1841
1842 007162 005037 002106
1843 007166 012737 000001 002100
1844 007174 012737 000002 002102
1845 007202
      007202 012737 036646 000004
1846 007210
      007210 004537 043710
      007214 026376
1847 007216 012737 005237 177654
1848 007224 012737 002076 177656
1849 007232 005237 002106
1850 007236 022737 000170 002106
1851 007244 001450
1852 007246 013701 002106
1853 007252 070127 000006
1854 007256 010137 002110
1855 007262 005037 002076
1856 007266 005037 002074
1857 007272 005037 002072
1858 007276
      007276 010346
      007300 013703 002106
      007304 004737 040700
      007310 012603
1859 007312 005761 003016
1860 007316 100533
1861 007320 012737 000207 177644 WARN1:
1862 007326 012700 060000
1863 007332 010004
1864 007334 012701 040000
1865 007340 010103
1866 007342 005002
1867 007344 104424
1868 007346
      007346 052737 040000 177776
1869 007354 004737 177640
1870 007360 104417
1871 007362 104423
1872 007364 000412
1873 007366 005037 002106

```

```

NEWST <<TEST BANKS 1-167 (OCTAL) FOR ZEROS & ONES>>
:*****
:*TEST 3 TEST BANKS 1-167 (OCTAL) FOR ZEROS & ONES
:*****
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.
      :ANY ERROR HERE WILL FORCE THE CONFIGURATION TABLE TYPEOUT!
      :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
      MOV #NONEXIST,4
      .DSABL CRF
      BMOV MTP000 ;PUT IN FAST MEMORY
      JSR R5,BLOCK1
      MTP000
      .DSABL CRF
      MOV #005237,UIPAR6 ;PUT "INC " INSTRUCTION IN SUBROUTINE
      MOV #PATERR,UIPAR7 ;PUT IN ADDRESS OF OPERAND FOR ABOVE
      INC BANK
      CMP #120,BANK ;DONE?
      BEQ TAG2$ ;YES - SKIP TO NEXT TEST
      MOV BANK,R1
      MUL #6,R1 ;BANK*3*2
      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
      MOV R3,-(SP)
      MOV (SP)+,R3
      TST CONFIG(R1) ;IS THIS BANK PROTECTED?
      BMI TSTBANK ;YES - GO TEST BANK SPECIAL
      MOV #207,UIPAR2 ;PUT "RETURN" INSTRUCTION AFTER WRITE ROUTINE
      MOV #FIRST,R0
      MOV R0,R4
      MOV #SIZE,R1
      MOV R1,R3
      CLR R2 ;DATA IS ZEROS
      CACHOFF ;TURN CACHE OFF
      SUPERVISOR ;ENTER SUPERVISOR MODE
      BIS #BIT14,PSW ;DO IT TO IT!
      .DSABL CRF
      CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
      KERNEL ;ENTER KERNEL MODE
      CACHON ;TURN CACHE ON
      BR TAG3$ ;SKIP NEXT INSTRUCTION
      CLR BANK
TAG9$:
TAG2$:
TAG3$:

```

```

1874 007372      RES4      ;RESET TRAPS TO 4 TO DEFAULT
      007372 012737 036726 000004  MOV      #TIMEOUT,4
      007400 005037 177766      CLR      CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                      ;THAT A EXPECTED TRAP COULD HAVE SET

1875 007404      .DSABL CRF      ;INDICATE DEFAULT PARITY ACTION
      007410 005037 002100      CLR      NOPAR
1876 007410 000543      BR      SUBAAQ
1877 007412 005737 002072      TAG3$: TST      NEMCNT      ;ANY TRAPS?
1878 007416 001401      BEQ      1$          ;NO - SKIP
1879 007420 000704      BR      TAG9$       ;NOW - TRY NEXT BANK
1880 007422 104424      1$:  CACHOFF      ;TURN CACHE OFF
1881 007424 052737 040000 177776  SUPERVISOR ;ENTER SUPERVISOR MODE
      007424      BIS      #BIT14,PSW ;DO IT TO IT!

1882 007432 004737 177646      .DSABL CRF
1883 007436 104417      CALL     UIPAR3      ;FINISH PATTERN
1884 007440 104423      KERNEL      ;ENTER KERNEL MODE
1885 007442 005737 002076      CACHON      ;TURN CACHE ON
1886 007446 001031      TST      PATERR      ;ANY PATTERN ERRORS
1887 007450 005737 002074      BNE      2$          ;YES - SKIP
1888 007454 001026      TST      PARCNT      ;ANY PARITY ERRORS
1889 007456 005737 002072      BNE      2$          ;YES - SKIP
1890 007462 001023      TST      NEMCNT      ;ANY NON EXISTANT MEMORY
1891 007464 012700 060000      BNE      2$          ;YES - SKIP
1892 007470 010004      MOV      #FIRST,R0
1893 007472 012701 040000      MOV      R0,R4
1894 007476 010103      MOV      #SIZE,R1
1895 007500 013702 002726      MOV      R1,R3
1896 007504 012737 000240 177644  MOV      ONES,R2      ;DATA IS ONES
      007512 104424      MOV      #000240,UIPAR2 ;PUT 'NOP' INSTRUCTION BACK IN SUBROUTINE
1897 007512      CACHOFF      ;TURN CACHE OFF
1898 007514 052737 040000 177776  SUPERVISOR ;ENTER SUPERVISOR MODE
      007514      BIS      #BIT14,PSW ;DO IT TO IT!

1899 007522 004737 177640      .DSABL CRF
1900 007526 104417      CALL     FASTCITY    ;CALL TO THE USER INSTRUCTION PAR'S
1901 007530 104423      KERNEL      ;ENTER KERNEL MODE
1902 007532 013700 002110      CACHON      ;TURN CACHE ON
1903 007536 005737 002076      2$:  MOV      BANKINDEX,R0
1904 007542 001006      TST      PATERR      ;ANY PATTERN ERRORS?
1905 007544 005737 002074      BNE      3$          ;YES - SKIP
1906 007550 001003      TST      PARCNT      ;ANY PARITY ERRORS?
1907 007552 005737 002072      BNE      3$          ;YES - SKIP
1908 007556 001406      TST      NEMCNT      ;ANY HOLES?
1909 007560 052760 004000 003016 3$:  BEQ      4$          ;NONE - SKIP
1910 007566 012737 177777 002576  BIS      #BIT11,CONFIG(R0) ;SET ERROR BIT IN THIS BANK
      007566      SET      CONFGERROR ;FORCE PRINTING OF CONFIGURATION TABLE
                                      MOV #-1,CONFGERROR
1911 007574 053760 002114 003016 4$:  BIS      (PUBIT,CONFIG(R0) ;SET ACCESSED BIT
1912 007602 000137 007232      JMP      TAG9$
1913
1914      ;TEST A PROTECTED BANK
1915 007606      TSTBANK: PUSH     R1
      007606 010146      MOV      #1,NONEM      ;SET NON-EXISTANT MEMORY TO COUNT
1916 007610 012737 000001 002102  MOV      #FIRST,R0
1917 007616 012700 060000      MOV      #SIZE,R1
1918 007622 012701 040000      MOV      #SIZE,R1
1919 007626 104424      CACHOFF      ;TURN CACHE OFF
1920 007630      SUPERVISOR ;ENTER SUPERVISOR MODE
    
```

```

007630 052737 040000 177776      BIS    #BIT14,PSW          ;DO IT TO IT!
                                .DSABL  CRF
1921 007636 005720                4$:  TST    (R0)+
1922 007640 077102                SOB    R1,4$
1923 007642 104417                KERNEL
1924 007644 104423                CACHON          ;ENTER KERNEL MODE
1925 007646 012737 000002 002102  MOV    #2,NONEM      ;TURN CACHE ON
1926 007654                POP    R1          ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
                                MOV (SP)+,R1
1927 007656                IF PARCNT NE #0
                                TST PARCNT
                                BEQ L44
                                007656 005737 002074
                                007662 001406
1928 007664 052761 004000 003016  BIS    #BIT11,CONFIG(R1) ;ERROR BANK
1929 007672 012737 177777 002576  SET    CONFGERROR      ;FORCE PRINTING OF CONFIGURATION TABLE
                                MOV #-1,CONFGERROR
1930 007700                END ;OF IF PARCNT
                                L44:
1931 007700                IF NEMCNT EQ #0
                                TST NEMCNT
                                BNE L45
                                007700 005737 002072
                                007704 001003
1932 007706 053761 002114 003016  BIS    CPUBIT,CONFIG(R1) ;ACCESSED BANK
1933 007714                END ;OF IF NEMCNT
                                L45:
1934 007714 000137 007232      JMP    TAG9$
  
```

1937 007720

SUBAAQ: SUBST <<DETERMINE PROTECTED BANKS>>

 :*SUBTEST DETERMINE PROTECTED BANKS

1938
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1949

:THE MK11 INHIBITS ECC DISABLE (AND OTHER THINGS) ON THE BOTTOM
 :16K WORDS OF EACH CONTROLLER (2 PER BOX).
 :THIS IS CONSIDERED TO BE A PROTECTED BANK BY THE PROGRAM.
 :USUALLY (ON A SINGLE BOX SYSTEM WITH 4K MOS RAMS) THIS IS BANKS #0 & #2
 :HOWEVER IT MAY BE QUITE COMPLEX TO DETERMINE ON A GIVEN SYSTEM
 :CONFIGURATION (MIXED MK11,MJ11; MIXED 4K MOS, 16K MOS; MULTIPLE MK11
 :BOXES; MIXED INTERLEAVE SCHEMES) WHICH BANKS ARE PROTECTED.
 :SO
 :THIS ROUTINE ATTEMPS TO CREATE A DOUBLE BIT ERROR IN ADDRESS #0 & #2
 :OF EVERY MK11 BANK. MK11 HARDWARE WILL PREVENT THIS FROM HAPPENING
 :IN PROTECTED BANKS WHICH SHOULD ALWAYS INCLUDE BANK ZERO - WHERE THE
 :PROGRAM IS.

1950
 1951
 1952
 1953
 1954
 1955

: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!

1956 007720 000004
 1957 007722 104424
 1958 007724 005037 002122
 1959 007730 012701 060000
 1960 007734 005004
 1961 007736

SCOPE
 CACHOFF ;TURN CACHE OFF
 CLR KPFLAG
 MOV #FIRST,R1
 CLR R4
 FOR BANK := #0 TO #167

007736 005037 002106
 007742
 1962 007742 013703 002106
 1963 007746 070327 000006
 1964 007752 010337 002110
 1965 007756

CLR BANK
 BIT:::~::~
 MOV BANK,R3
 MUL #6,R3
 MOV R3,BANKINDEX
 IF CPUBIT SET.IN CONFIG(R3) ;IF ACCESSABLE
 BIT CPUBIT,CONFIG(R3)
 BEQ L46

007756 033763 002114 003016
 007764 001520
 1966 007766
 007766 032763 020000 003016
 007774 001514
 007776 032763 040000 003016
 010004 001110

IF #BIT13 SET.IN CONFIG(R3) AND #BIT14 OFF.IN CONFIG(R3) ;IF MK11
 BIT #BIT13,CONFIG(R3)
 BEQ L47
 BIT #BIT14,CONFIG(R3)
 BNE L47

1967 010006
 010006 010346
 010010 013703 002106
 010014 004737 040700

MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 MOV R3,-(SP)
 MOV BANK,R3
 CALL MAPPER
 .DSABL CRF
 MOV (SP)+,R3

010020 012603
 1968 010022 005037 002144
 1969 010026
 010026 052737 040000 177776

CLR CSR
 SUPERVISOR ;ENTER SUPERVISOR MODE
 BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF

1970 010034
 010034 013746 060000
 010040 013746 060002
 1971 010044 104504
 1972 010046 010411
 1973 010050 010461 000002

PUSH FIRST,FIRST+2 ;SAVE TWO TEST LOCATIONS
 MOV FIRST,-(SP)
 MOV FIRST+2,-(SP)
 CHKDIS ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
 MOV R4,(R1) ;WRITE CHECKBITS (ALL ZEROS)
 MOV R4,2(R1)

1974	010054	104502			CLRCR		;CLEAR MK11 CSRS
1975	010056	005711			TST (R1)		;READ CHECKBITS INTO REAL CSR
1976	010060	104500			WASDBE		;WAS THERE A DOUBLE BIT ERROR
1977							;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
1978	010062				ON.NOERROR ;1		
	010062	103441					BCS L50
1979	010064	005037	002144		CLR CSR		
1980	010070	104504			CHKDIS		;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
1981	010072	013711	002726		MOV ONES,(R1)		
1982	010076	013761	002726	000002	MOV ONES,2(R1)		
1983	010104	104502			CLRCR		;CLEAR MK11 CSRS
1984	010106	005711			TST (R1)		
1985	010110	104500			WASDBE		;WAS THERE A DOUBLE BIT ERROR
1986	010112				ON.NOERROR ;2		
	010112	103425					BCS L51
1987	010114	012737	007400	002144	MOV #7400,CSR		
1988	010122	104504			CHKDIS		;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
1989	010124	010411			MOV R4,(R1)		
1990	010126	010461	000002		MOV R4,2(R1)		
1991	010132	104502			CLRCR		;CLEAR MK11 CSRS
1992	010134	005711			TST (R1)		
1993	010136	104500			WASDBE		;WAS THERE A DOUBLE BIT ERROR
1994	010140				ON.NOERROR ;3		
	010140	103412					BCS L52
1995	010142	012737	074000	002144	MOV #74000,CSR		
1996	010150	104504			CHKDIS		;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
1997	010152	010411			MOV R4,(R1)		
1998	010154	010461	000002		MOV R4,2(R1)		
1999	010160	104502			CLRCR		;CLEAR MK11 CSRS
2000	010162	005711			TST (R1)		
2001	010164	104500			WASDBE		;WAS THERE A DOUBLE BIT ERROR
2002	010166				END ;OF ON.NOERROR ;3		
	010166						L52:::~::~
2003	010166				END ;OF ON.NOERROR ;2		
	010166						L51:::~::~
2004	010166				END ;OF ON.NOERROR ;1		
	010166						L50:::~::~
2005	010166				ON.NOERROR		
	010166	103405					BCS L53
2006					;MARK PROTECTED BANK		
2007	010170	013703	002110		MOV BANKINDEX,R3		
2008	010174	052763	010000	003016	BIS #BIT12,CONFIG(R3)		;IDENTIFY AS PROTECTED BY MK11
2009	010202				END ;OF ON.NOERROR		
	010202						L53:::~::~
2010	010202	104470			ECCDIS		;DISABLE ERROR CORRECTION
2011	010204	010411			MOV R4,(R1)		;CLEAR OUT DOUBLE BIT ERROR!
2012	010206	010461	000002		MOV R4,2(R1)		;CLEAR OUT DOUBLE BIT ERROR!
2013	010212	104502			CLRCR		
2014	010214				POP FIRST+2,FIRST		;RESTORE TWO TEST LOCATIONS
	010214	012637	060002				MOV (SP)+,FIRST+2
	010220	012637	060000				MOV (SP)+,FIRST
2015	010224	104417			KERNEL		;ENTER KERNEL MODE
2016	010226				END ;OF IF #BIT3		
	010226						L47:::~::~
2017	010226				END ;OF IF CPUBIT		
	010226						L46:::~::~
2018	010226				END ;OF FOR BANK		

	010226	005237	002106						
	010232	023727	002106	000167					INC BANK
	010240	003640							CMP BANK,#167
	010242								BLE B1
2019	010242				MAP			E1:::~::~	;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
	010242	010346							MOV R3,-(SP)
	010244	012703	000170		MOV #120.,R3				
	010250	004737	040700		CALL MAPPER				
					.DSABL CRF				
2020	010254	012603							MOV (SP)+,R3
2021	010256	005037	002106		CLR BANK				
2022	010262				IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE				
	010262	032777	000001	172476					BIT #SW0,@SWR
	010270	001003							BNE L54
	010272	005737	002442						TST ACTFLAG
	010276	001402							BEQ L55
	010300								
2023	010300	104506			ENASBE			L54:::~::~	;TRAP ON SINGLE BIT ERRORS
2024	010302				ELSE				
	010302	000401							BR L56
	010304								
2025	010304	104472			ECCINIT			L55:::~::~	;TRAP ON DOUBLE BIT ERRORS (NORMAL)
2026	010306				END ;OF IF #SW0				
	010306								L56:::~::~
2027									
2028	010306	104423			CACHON				;TURN CACHE ON

```
2031 010310 NEWTST <<MK11 PROTECTION POINTER TEST>>
:*****
:*TEST 4 MK11 PROTECTION POINTER TEST
:*****
TST4: SCOPE
;INSURE THAT BOTTOM OF MK11 WAS PROTECTED WITH PROTECTION POINTER CLEAR
BEGIN PROTECTIONTEST
FOR BANK := #0 TO #167
CALL EXBANK
IF MKFLAG IS TRUE AND ACFLAG IS TRUE
IF KPFLAG IS FALSE
ERROR +40 ;MK11 PROTECTION POINTER FAILURE - DID NOT PROTECT BANK
END ;OF IF KPFLAG
LEAVE PROTECTIONTEST
END ;OF IF MKFLAG
END ;OF FOR BANK
END PROTECTIONTEST

B2:*****
CLR BANK
B3:*****
TST MKFLAG
BEQ L57
TST ACFLAG
BEQ L57
TST KPFLAG
BNE L60
L60:*****
BR E2
L57:*****
INC BANK
CMP BANK,#167
BLE B3
E3:*****
E2:*****
```

010310 000004
2032 010312
2033 010312
2034 010312 005037 002106
010316
2035 010316 004737 042422
2036 010322 005737 002126
010326 001410
010330 005737 002124
010334 001405
2037 010336 005737 002122
010342 001001
2038 010344 104040
2039 010346
010346
2040 010346 000406
010346
2041 010350
010350
2042 010350 005237 002106
010354 023727 002106 000167
010362 003755
010364
2043 010364
010364

2046 C10364

SUBTST <<CHECK SYSTEM SIZE REGISTER>>

: *SUBTEST CHECK SYSTEM SIZE REGISTER

2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079

: THIS CHECKS THE SYSTEM SIZE REGISTER TO INSURE IT IS CORRECT
: AND PROGRAMS ANY MK11 BOXES ON LINE THAT ARE NOT INHIBITED.
:
: NOTE: IF THE SYSTEM SIZE REGISTER IS SET TO INDICATE A VERY SMALL
: MEMORY THE PROGRAM MAY NOT BE ABLE TO FUNCTION.
: IF THE MK11 BOX STARTING ADDRESS LOGIC MALFUNCTIONS THE PROGRAM
: WILL EITHER CRASH BECAUSE OF DUAL ADDRESSING PROBLEMS OR
: WILL HAVE HOLES IN MEMORY (NON-EXISTANT BANKS).
:
: THE TEST FUNCTIONS AS FOLLOWS
:
: 1. THE PROGRAM CHECKS FOR MEMORY ON 16K WORD (BANK) BOUNDRIES
: THRU THE UNIBUS MAP (SO THAT THE SYSTEM SIZE REGISTER IS BYPASSED).
: LOCATION 'UNITOP' IS ASSIGNED THE NUMBER OF THE LAST BANK THAT WAS
: ACCESSABLE VIA THE UNIBUS MAP.
:
: 2. LOCATION 'HIBANK' IS SET TO THE HIGHEST BANK THAT IS EITHER
: (1) DIRECTLY ACCESSABLE VIA THE MEMORY BUS OR
: (2) AN MK11 CSR CLAIMS THAT IT EXISTS.
:
: 3. LOCATION 'HIACBANK' IS ASSIGNED THE NUMBER OF THE HIGHEST BANK
: THAT IS DIRECTLY ACCESSABLE VIA THE MEMORY BUS.
:
: 4. IF THE HIGHEST ACCESSABLE BANK IS LESS THAN #170
: IF ANY BOX STARTING ADDRESSES WERE ABOVE BANK #167
: TRY TO PROGRAM THE STARTING ADDRESS OF THE PENDING BOX
: IF IT DOES PROGRAM DOWN AND THEREFORE ADD MEMORY TO THE SYSTEM
: RESTART THE PROGRAM
: NEXT PENDING BOX
:
: 5. R0 IS SET TO THE HIGHEST BANK NUMBER OF HIBANK OR UNITOP
:
: 6. IF THE SYSTEM SIZE REGISTER DOES NOT EQUAL R0'S BANK THEN PRINT THE ERROR

2082					:FIND UNITOP		
2083	010364	104424			CACHOFF		:TURN CACHE OFF
2084	010366	005037	170372		CLR MAPH36		:SET MAP TO BANK 1
2085	010372	012737	100000	170370	MOV #100000,MAPL36		
2086	010400				SET4 #2\$:TRAPS TO 4 GOTO 2\$
	010400	012737	010446	000004	MOV #2\$,4		
					.DSABL CRF		
2087	010406	012737	000004	002100	MOV #4,NOPAR		:PARITY ACTION = COUNT & IGNORE
2088	010414	012700	140000		MOV #140000,R0		:PAR6 SELECTED
2089	010420	005710			TST (R0)		:TRAP IF NON-EXISTANT
2090	010422	062737	100000	170370	ADD #100000,MAPL36		:BUMP 1 BANK
2091	010430	005537	170372		ADC MAPH36		
2092	010434	022737	000073	170372	CMP #73,MAPH36		:INTO 128K SHADOW MEMORY?
2093	010442	001366			BNE 1\$:NO - LOOP
2094	010444	000402			BR 3\$		
2095	010446	062706	000004		ADD #4,SP		:FIX STACK FROM TRAP
2096	010452	162737	100000	170370	SUB #100000,MAPL36		
2097	010460	005637	170372		SBC MAPH36		
2098	010464				RES4		:RESET TRAPS TO 4 TO DEFAULT
	010464	012737	036726	000004	MOV #TIMEOUT,4		
	010472	005037	177766		CLR CPUERR		:CLEAR OUT THE CPU ERROR REGISTER BITS :THAT A EXPECTED TRAP COULD HAVE SET
					.DSABL CRF		
2099	010476	005037	002100		CLR NOPAR		:RESTORE DEFAULT PARITY ACTION
2100	010502	006137	170370		ROL MAPL36		
2101	010506	006137	170372		ROL MAPH36		
2102	010512	013737	170372	002516	MOV MAPH36,UNITOP		
2103							
2104					:FIND HIBANK		
2105	010520	012737	000077	170372	MOV #77,MAPH36		:RESTORE MAP36
2106	010526	012737	160000	170370	MOV #160000,MAPL36		
2107	010534	104423			CACHON		:TURN CACHE ON
2108	010536				PUSH SLAVES		
	010536	013746	002730				MOV SLAVES,-(SP)
2109	010542	012737	000003	002730	MOV #3,SLAVES		:FAKE OUT THE EXBANK ROUTINE
2110	010550				BEGIN MEMSIZELOOP		
	010550						B4:::~::~
2111	010550				FOR BANK := #0 TO #167		
	010550	005037	002106				CLR BANK
	010554						B5:::~::~
2112	010554	004737	042422		CALL EXBANK		
2113	010560				IF ACFLAG IS FALSE AND MKFLAG IS FALSE		
	010560	005737	002124				TST ACFLAG
	010564	001004					BNE L61
	010566	005737	002126				TST MKFLAG
	010572	001001					BNE L61
2114	010574				LEAVE MEMSIZELOOP		
	010574	000406					BR E4
2115	010576				END ;OF IF ACFLAG		
	010576						L61:::~::~
2116	010576				END ;OF FOR BANK		
	010576	005237	002106				INC BANK
	010602	023727	002106	000167			CMP BANK,#167
	010610	003761					BLE B5
	010612						E5:::~::~
2117	010612				END MEMSIZELOOP		
	010612						E4:::~::~

2118	010612	013700	002106	MOV	BANK,RO
2119	010616	005300		DEC	RO
2120	010620	010037	002512	MOV	RO,HIBANK

```

2123                                     :FIND HIACBANK
2124 010624                               CLEAR  BANK,HIACBANK
      010624 005037 002106
      010630 005037 002514
2125 010634                               SET    ACFLAG
      010634 012737 177777 002124
2126 010642                               WHILE ACFLAG IS TRUE AND BANK LT #167
      010642
      010642 005737 002124
      010646 001411
      010650 023727 002106 000167
      010656 002005
2127 010660                               INC    BANK
2128 010664                               CALL  EXBANK
2129 010670                               END ;OF WHILE ACFLAG
      010670 000764
      010672
      010672
2130 010672                               POP    SLAVES
      010672 012637 002730 002514
2131 010676 013737 002106 002514       MOV    BANK,HIACBANK
2132
2133                                     :PROGRAM ON ANY BOXES SETUP FOR ALLOW PROGRAM
2134                                     :& HI (100) STARTING ADDRESSES
2135 010704                               IF HIACBANK LT #170
      010704 023727 002514 000170
      010712 002071
2136 010714                               IF PENDBOX IS TRUE
      010714 005737 002330
      010720 001466
2137 010722 013701 002330
2138 010726 005037 002256
      010732
2139 010732 106301
2140 010734                               ASLB  R1
      010734 103051                               ON.ERROR
2141 010736                               SET   TWOCSRS
      010736 012737 177777 002600
2142 010744 012737 000001 002144
2143 010752 013737 002514 002146
2144 010760 000241
2145 010762 006037 002146
2146 010766 005537 002146
2147 010772 052737 002000 002146
2148 011000 104425
2149 011002                               MOV   #1,CSR
      011002 005037 002144
      011006 005037 002146
      011012 005037 002600
2150 011016 012737 000010 002144
2151 011024 104425
2152 011026 104426
2153 011030 013702 002146
2154 011034 042702 177000
2155 011040 006302
2156 011042                               MOV   #BIT3,CSR
      LOADCSR
      READCSR
      MOV   CSR+2,R2
      BIC   #177000,R2
      ASL   R2
      IF R2 LT #167
    
```

```

CLR  BANK
CLR  HIACBANK
MOV  #-1,ACFLAG
B6:.....
TST  ACFLAG
BEQ  L62
CMP  BANK,#167
RGE  L62
    
```

```

BR  B6
L62:.....
E6:.....
MOV  (SP)+,SLAVES
    
```

```

CMP  HIACBANK,#170
BGE  L63
TST  PENDBOX
BEQ  L64
    
```

```

CLR  CSRNO
B7:.....
BCC  L65
MOV  #-1,TWOCSRS
    
```

:IF NOT 32K WORD BOUND LEAVE A WHOLE IN MEM
 :LOAD START ADD & INTERLEAVE BIT

```

CLR  CSR
CLR  CSR+2
CLR  TWOCSRS
    
```

:IF I CHANGED IT'S STARTING ADDRESS

011042 020227 000167
011046 002004
2157 011050 005037 002670
2158 011054 000137 004352
2159 011060
011060
2160 011060
011060
2161 011060
011060 062737 000004 002256
011066 023727 002256 000034
011074 003716
011076
2162 011076
011076
2163 011076
011076

CLR LOCK
JMP START
END ;OF IF R2
END ;OF ON.ERROR
END ;OF FOR CSRNO
END ;OF IF PENDBOX
END ;OF IF HIACBANK

CMP R2,#167
BGE L66
L66:::
L65:::
ADD #4,CSRNO
CMP CSRNO,#34
BLE B7
E7:::
L64:::
L63:::


```

011306
2196 011306 032700 000100 IF #BIT6 SET.IN R0 THEN LET R1 := R1 SET.BY #BIT5
011306 001402 BIT #BIT6,R0
011312 001402 BEQ L76
011314 052701 000040 BIS #BIT5,R1
011320
2197 011320 032700 000200 IF #BIT7 SET.: R0 THEN LET R1 := R1 SET.BY #BIT6
011320 001402 BIT #BIT7,R0
011324 001402 BEQ L77
011326 052701 000100 BIS #BIT6,R1
011332
2198 011332 000207 RETURN
L75:.....
L76:.....
L77:.....
```



```
011532
2231 011532
2232 011532 012700 172340
2233 011536 012701 002550
2234 011542 012705 000010
2235 011546 012021
2236 011550 077502
2237 011552
011552
2238
2239 011552
011552 012737 177777 002520

: COPY MAPPING FOR ANY SLAVES
MOV #KIPAR0,R0
MOV #KIO,R1
MOV #8,R5
1$: MOV (R0)+,(R1)+
SOB R5,1$
END ;OF IF CPUBIT

L104::::::
L100::::::
SET STOPOK
MOV #-1,STOPOK
```



```

011714
2279 011714 010260 002606          MOV     R2,PORTDIR(R0)          L112:.....
2280
2281          ;THIS IS WHERE THE SLAVES COME TO STOP
2282 011720 010003          STOPCPU:MOV     R0,R3
2283 011722          IF NOIIST IS TRUE
011722 005737 002756          TST NOIIST
011726 001417          BEQ L113
2284 011730          IF R0 EQ #0 THEN GOTO MP1      ;SKIP IF I'M THE MASTER
011730 005700          TST R0
011732 001456          BEQ MP1
2285 011734 052760 100000 002616    BIS     #BIT15,TASK(R0)
2286 011742          BMOV    WAITSTUFF
011742 004537 043710          JSR    R5,BLOCK1
011746 012034          WAITSTUFF
          .DSABL      CRF
2287 011750 005337 002670          DEC     LOCK                    ;ALLOW OTHERS TO RUN
2288 011754 062700 002636          ADD     #PTYBUF,R0              ;R0 POINTS TO COUNT DOWN LOCATION
2289
2290          ;WAIT FOR MASTER TO COUNT DOWN!
2291 011760 000137 177640          JMP     FASTCITY                ;CALL TO THE USER INSTRUCTION PAR'S
2292 011764          ELSE
011764 000423          BR     L114
011766          L113:.....
2293 011766 004737 061074          CALL   IISTINIT
2294 011772          IF R0 EQ #0 THEN GOTO MP1      ;SKIP IF I'M THE MASTER
011772 005700          TST R0
011774 001435          BEQ MP1
2295 011776 012777 000001 170732    MOV     #1,@IISTACR
2296 012004 012777 000004 170726    MOV     #BIT2,@IISTADR          ;ENABLE INTERRUPTS
2297 012012 000234          SPL     4
2298 012014 052760 100000 002616    BIS     #BIT15,TASK(R0)
2299 012022 005337 002670          DEC     LOCK                    ;ALLOW OTHERS TO RUN
2300 012026 000001          WAIT                    ;WAIT IF I'M A SLAVE
2301 012030 000137 050726          JMP     TASKDO
2302 012034          END ;OF IF NOIIST
012034

```

L114:.....

2305 012034

WAITSTUFF:SUBTST <<SLAVES WAIT FOR MASTER TO COUNT DOWN>>

:*SUBTEST SLAVES WAIT FOR MASTER TO COUNT DOWN

2306 012034 012702 000014
2307 012040 012701 000013
2308 012044 005302
2309 012046 005301
2310 012050 020210
2311 012052 001776
2312 012054 020110
2313 012056 001366
2314 012060 005701
2315 012062 001370
2316 012064 000137 061216

1\$: MOV #12.,R2 :V177640
MOV #11.,R1 :V177644
2\$: DEC R2 :V177650
DEC R1 :V177652
3\$: CMP R2,(R0) :V177654
BEQ 3\$:V177656
CMP R1,(R0) :V177660
BNE 1\$:V177662
4\$: TST R1 :V177664
BNE 2\$:V177666
JMP SLAVUP :V177670

2319 012070

```
MP1:  SUBTST <<BOOT UP ALL EXISTING CPU'S>>
:*****
:*SUBTEST  BOOT UP ALL EXISTING CPU'S
:*****
:ALL SLAVES ARE BOOTED AS FOLLOWS
:
:SETUP A SOFTWARE SWITCH REGISTER.
:IF WE ARE USING IIST'S.
:  BEGIN BOOTLOOP
:    FOR EACH OF 4 POSSIBLE CPU'S.
:      SETUP STACK VALUE LESS THAN PREVIOUS CPU.
:      CLEAR ACKNOWLEDGE.
:      POINT IIST INTERRUPT VECTOR TO START.
:MP2:  WAIT FOR THE MASTER'S IIST TO BE READY (SHOULD ALREADY BE).
:      INITIALIZE THE MASTER'S IIST (DISABLE BOOTS & INTERRUPTS).
:      INITIATE THE IIST BOOT OF THE SELECTED CPU.
:      WAIT FOR THE WINDOW IN THE BOOT THAT ALLOWS INTERRUPTS.
:      INITIATE THE IIST INTERRUPT OF THE SELECTED CPU.
:      WAIT FOR THE SLAVE TO ACKNOWLEDGE STARTING.
:      IF THE CPU DOES NOT ACKNOWLEDGE
:        IF THE CPU SELECTED IS NOT THE MASTER
:          LEAVE BOOTLOOP.
:        ELSE
:          GOTO MP2.
:      END OF IF THE CPU SELECTED IS NOT THE MASTER.
:    ELSE
:      WAIT FOR TASK DONE FROM SLAVE (TASK=START).
:      UPDATE SLAVE COUNT (THIS SLAVE IS THE HIGHEST # SO FAR).
:      IF THIS WAS THE 4TH CPU THEN LEAVE BOOTLOOP.
:    END OF IF THE CPU DOES NOT ACKNOWLEDGE.
:  END OF FOR EACH OF 4 POSSIBLE CPU'S.
:  END OF BOOTLOOP.
:  IF THERE WERE NO SLAVES
:    CLEAR THE MULTIPROCESSING FLAG 'MULTIPOINT'.
:    SET THE IIST VECTOR TO LOOK FOR UNEXPECTED TRAPS.
:  END OF IF THERE WERE NO SLAVES.
:ELSE
:  FOR EACH SLAVE
:    SETUP STACK VALUE LESS THAN PREVIOUS CPU.
:    TYPE "START SLAVE #N AT ADDRESS 200".
:    WAIT FOR TASK DONE FROM SLAVE (TASK=START).
:  END OF FOR EACH SLAVE.
:END OF IF WE ARE USING IIST'S.
:RESTORE THE MASTER TO THE NORMAL SWITCH REGISTER
:WAIT A SECOND.
:MAKE SURE I'M THE ONLY PROCESSOR RUNNING.
:SETUP STACK VALUE BACK TO MASTER'S DEFAULT.
:TURN CACHE ON.
```

2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363

```

2366
2367
2368 012070 005337 002670      .ENABL  LSB
DEC      LOCK                  ;ALLOW ONE SLAVE TO RUN
2369      :PASS PSEUDO SWR TO SLAVE
2370 012074 013737 002766 002674  MOV      SWR,MASWR           ;SAVE REAL SWR
2371 012102 012737 000176 002766  MOV      #SWREG,SWR         ;SETUP SOFTWARE SWR
2372 012110 017777 170560 170650  MOV      @MASWR,@SWR        ;FILL IT WITH REAL SWR STUFF
2373 012116
012116 005737 002756      IF NOIIST IS TRUE THEN JUMPTO MP4
012122 001402
012124 000137 012552
012130
2374 012130 012702 000400      MOV      #BIT8,R2           ;BOOT = BITS 11,10,9,8
2375 012134 012703 000001      MOV      #1,R3             ;INTERRUPT = BITS 3,2,1,0
2376 012140
012140
2377 012140
012140 012701 000001      FOR R1 := #1 TO #3
012144
2378 012144 162737 000200 002706  SUB      #200,KSTACK        ;SLAVES - PLEASE USE YOUR OWN STACK
2379 012152 013737 002706 000000  MOV      KSTACK,0          ;PUT IN LOC 0 FOR BOOTSTRAP
2380 012160 012700 000020      MOV      #MASTER,R0       ;SET FIRST CPU ID BIT
2381 012164 072001      ASH      R1,R0             ;SHIFT IT TO CORRECT SPOT
2382 012166 010037 002114      MOV      R0,CPUBIT        ;CPUBIT = BITS 5 OR BIT6 OR BIT7
2383 012172 010100      MOV      R1,R0            ;CPU # ---> R0
2384 012174 006300      ASL      R0                ;R0 := 2 OR 4 OR 6 ;INDEX VALUE
2385 012176 005037 002672      CLR      ACK              ;ALLOW ONLY ONE CPU AT A TIME
2386 012202 013704 002742      MOV      IISTVEC,R4
2387 012206 012724 004352      MOV      #START,(R4)+     ;INTERRUPT VECTOR TO START
2388 012212 012714 000340      MOV      #340,(R4)       ;HIGH PRIORITY
2389
2390      ;WAIT FOR IIST READY
2391 012216 012737 061165 002574  MOV      #5*SECONDS,HUNGTIME
2392 012224 012777 000001 170504  MP2:  MOV      #1,@IISTACR
2393 012232 004737 072154      CALL     QUE
2394 012236
012236 103004
2395 012240 104061
2396 012242 005037 002730      ERROR   +61
2397 012246
012246 000530      CLR     SLAVES
LEAVE    BOOTLOOP
2398 012250
012250      END ;OF ON.ERROR
2399 012250 032777 004000 170462  BIT      #BIT11,@IISTADR
2400 012256 001762      BEQ     MP2
2401
2402 012260 004737 061174      CALL     IISTOFF          ;INITIALIZE THE IIST & DISABLE BOOTS & INTERRUPTS
2403 012264 012777 000000 170444  MOV      #0,@IISTACR      ;SELECT THE ACCESS CONTROL REGISTER
2404 012272 010277 170442      MOV      R2,@IISTADR     ;BOOT SOME CPU
2405 012276 012777 000001 170434  MOV      #1,@IISTADR     ;NOW!
2406
2407      ;THIS DELAY IS GOOD ONLY FOR THE M9312 MULTIPROCESSOR BOOT
      ;THIS DELAY ALSO ASSUMES THAT CACHE IS OFF
2408 012304 012704 140000      MOV      #140000,R4      ;THIS DELAY TAKES APPROX 3/4 SECOND
2409 012310 000240
2410 012312 000240
2411 012314 000240
2412 012316 000240
    
```

2413 012320 005237 000110
2414
2415
2416 012324 077407

INC 110

SOB R4,3\$

:CHANGE MOVING LOCATION - SEE BOOT LISTING
:THIS STOPS THE BOOT FROM RUNNING THE MEMORY
:DIAGNOSTIC AND BLOWING UP MY SOFTWARE!

```

2418 012326 012777 000000 170402      MOV #0,@IISTACR      ;SELECT THE ACCESS CONTROL REGISTER
2419 012334 010377 170400      MOV R3,@IISTADR     ;INTERRUPT SOME CPU
2420 012340 012777 000001 170372      MOV #1,@IISTADR     ;NOW!
2421                                ;WAIT FOR ACK OR TIMEOUT
2422 012346 012705 000003      MOV #3,R5           ;THIS IS APPROX A 3 SECOND DELAY
2423 012352 005004      CLR R4
2424 012354 005737 002672      1$: IF ACK IS FALSE
                                TST ACK
                                BNE L117
2425 012362 005237 000110      INC 110             ;CHANGE MOVING LOCATION - SEE BOOT LISTING
                                ;THIS STOPS THE BOOT FROM RUNNING THE MEMORY
                                ;DIAGNOSTIC AND BLOWING UP MY SOFTWARE!
2426
2427
2428 012366 077406      SOB R4,1$
2429 012370 077507      SOB R5,1$
2430 012372      END ;OF IF ACK IS FALSE
2431 012372      IF ACK IS FALSE      ;IF THERE IS NO SLAVE AT THIS IISTID #
                                L117:::
                                TST ACK
                                BNE L120
2432 012400 006302      ASL R2
2433 012402 006303      ASL R3
2434 012404 020327 000020      IF R3 EQ #BIT4
                                CMP R3,#BIT4
                                BNE L121
2435 012412      LEAVE BOOTLOOP
                                BR E12
2436 012414      ELSE
                                BR L122
                                L121:::
2437 012416      GOTO MP2
                                BR MP2
2438 012420      END ;OF IF R3
                                L122:::
2439 012420      ELSE
                                BR L123
                                L120:::
2440 012422 005037 002672      CLR ACK             ;ALLOW SLAVE TO PROCEED
2441 012426 012737 000005 002572      MOV #5,HUNGMSB
2442 012434 012737 177777 002574      MOV #-1,HUNGTIME
2443 012442 004737 072154      2$: CALL QUE
2444 012446      ON.ERROR
                                BCC L124
2445 012450 162737 000001 002572      SUB #1,HUNGMSB
2446 012456      ON.ERROR
                                BCC L125
2447 012460 010137 002056      MOV R1,BAD
2448 012464 104060      ERROR +60
                                ;SLAVE IS HANGING SYSTEM!
2449 012466 005037 002730      CLR SLAVES
2450 012472      LEAVE BOOTLOOP
                                BR E12
2451 012474      END ;OF ON.ERROR
                                L125:::
2452 012474      END ;OF ON.ERROR
                                L124:::
2453 012474 005760 002616      TST TASK(R0)
2454 012500 100360      BPL 2$
2455 012502 010137 002730      MOV R1,SLAVES      ;COUNT OF SLAVES WAS IN R1
    
```

```

2456 012506          IF R3 EQ #BIT3 THEN LEAVE BOOTLOOP
      012506 020327 000010
      012512 001406
2457 012514          END ;OF IF ACK
      012514
2458 012514 006302
2459 012516 006303
2460 012520          ASL R2
      012520 005201          ASL R3
      012522 020127 000003          END ;OF FOR R1
      012526 003606
      012530          INC R1
2461 012530          END BOOTLOOP
      012530          CMP R1,#3
      012530          BLE B13
2462 012530          IF SLAVES EQ #0
      012530 005737 002730          E13:
      012534 001005          E12:
2463 012536 005037 003014          TST SLAVES
2464 012542 012777 036764 170172          BNE L126
2465 012550          CLR MULTIPOINT
      012550          MOV #IITRAP,@IISTVEC          ;RESTORE UNEXPECTED IIST TRAP
2466 012550 000462          END ;OF IF SLAVES
      012550          L126:
2467 012552          BR MP3
      012552 012701 000001          FOR R1 := #1 TO SLAVES
      012556          MOV #1,R1
      012556          B14:
2468 012556 012700 000020          MOV #BIT4,R0
2469 012562 072001          ASH R1,R0
2470 012564 010037 002114          MOV R0,CPUBIT          ;CPUBIT = BIT5 OR BIT6 OR BIT7
2471 012570 162737 000200 002706          SUB #200,KSTACK
2472 012576 010100          MOV R1,R0
2473 012600 006300          ASL R0          ;R0 = 2 OR 4 OR 6
2474 012602          TYPE MSG081          ;START SLAVE #
      012602 104401 115560          TYPEIT ,MSG081
2475 012606          .DSABL CRF
      012606 010146          TYPOCS R1,,1
      012610 104403          MOV R1,-(SP)          ;;SAVE R1 FOR TYPEOUT
      012612 001          TYPOS          ;;GO TYPE--OCTAL ASCII
      012613 000          .BYTE 1          ;;TYPE 1 DIGIT(S)
      012613          .BYTE 0          ;;SUPPRESS LEADING ZEROS
2476 012614          .DSABL CRF
      012614 104401 115577          TYPE MSG082          ;AT ADDRESS 200
      012614          TYPEIT ,MSG082
2477 012620 012737 000031 002572          .DSABL CRF
2478 012626 012737 177777 002574          MOV #25.,HUNGMSB
2479 012634 005037 002672          MOV #-1,HUNGTIME
2480 012640 004737 072154          CLR ACK          ;ALLOW SLAVE TO PROCEED
2481 012644          CALL QUE
      012644 103015          ON.ERROR
2482 012646 162737 000001 002572          SUB #1,HUNGMSB
2483 012654          ON.ERROR          BCC L127
      012654 103011          BCC L130
2484 012656 010137 002056          MOV R1,BAD
2485 012662 104060          ERROR +60          ;SLAVE IS HANGING SYSTEM!
2486 012664          CLEAR SLAVES,MULTIPOINT
      012664 005037 002730          CLR SLAVES
      012670 005037 003014          CLR MULTIPOINT
    
```

```

2487 012674 000137 012716          JMP      MP3
2488 012700          END ;OF ON.ERROR
2489 012700          END ;OF ON.ERROR
2490 012700 005760 002616          TST     TASK(R0)
2491 012704 100353          BPL     5$
2492 012706          END ;OF FOR R1
                INC R1
                CMP R1,SLAVES
                BLE B14
                E14:
2493          ;POINT THE MASTER BACK TO THE NORMAL SWR
2494 012716 013737 002674 002766 MP3:  MOV     MASWR,SWR
2495 012724 012737 000020 002114  MOV     #MASTER,CPUBIT
2496 012732 052737 000020 002510  BIS     #MASTER,ALLCPUS
2497          ;WAIT A SECOND
2498 012740 005000          CLR     R0
2499 012742 077001          SOB    R0,4$
2500          ;STOP ANY CONFUSED MASTER (ME)
2501 012744 005237 002670          INC     LOCK
2502 012750          IF LOCK NE #1
                CMP LOCK,#1
                BEQ L131
2503 012760 000000          MHALT3: HALT
2504 012762 000776          BR     MHALT3
2505 012764          END ;OF IF LOCK
                L131:
2506 012764 012737 000020 002114 MP5:  MOV     #MASTER,CPUBIT          ;BACK TO MASTER
2507 012772 012737 002000 002706  MOV     #STACK,KSTACK
2508 013000 104423          CACHON          ;TURN CACHE ON
2509          .DSABL  LSB
    
```

```
2512 013002          SUBTST <<CHECK FOR UNIQUE CPU ID NUMBERS>>
:*****
:*SUBTEST          CHECK FOR UNIQUE CPU ID NUMBERS
:*****
2513 013002 104424   CACHOFF          ;TURN CACHE OFF
2514 013004          BEGIN SAMESAME
2515 013004 013702 002730  MOV     SLAVES,R2
2516 013010 006302   ASL     R2
2517 013012 005000   FOR R0 := #0 TO R2 BY #2
                                CLR R0
2518 013014          FOR R1 := #0 TO R2 BY #2
                                B16:*****
2519 013016          IF R0 NE R1
                                CLR R1
2520 013022 020001   IF PORTDIR(R0) EQ PORTDIR(R1)
                                B17:*****
2521 013032 001006   ERROR     +41
2522 013034 104401 115617  TYPEIT   ,MSG084          ;PROCEEDING AS A SINGLE PORT TEST
                                .DSABL   CRF
2523 013040          LEAVE SAMESAME
2524 013042 000412   CLR     MULTIPOINT
2525 013046 005037 003014   END ;OF IF PORTDIR(R0)
                                BR E15
2526 013046          END ;OF IF R0
                                L133:*****
2527 013046 062701 000002   END ;OF FOR R1
                                L132:*****
2528 013056 062700 000002   END ;OF FOR R0
                                ADD #2,R1
2529 013066          END SAMESAME
                                CMP R1,R2
2530 013066 104423   CACHON          ;TURN CACHE ON
                                BLE B17
2531 013070          SUBTST <<PRINT MULTIPOINT DIRECTORY>>
2532 013070          :*****
:*SUBTEST          PRINT MULTIPOINT DIRECTORY
:*****
2533 013070 005737 003014   IF MULTIPOINT IS TRUE
                                E17:*****
2534 013076 104401 115706   TYPE     MSG086
                                TYPEIT   ,MSG086
                                .DSABL   CRF
2535 013102          TYPOCS          PORTDIR
                                ADD #2,R0
                                CMP R0,R2
                                BLE B16
                                E16:*****
                                E15:*****
                                TST MULTIPOINT
                                BEQ L134
```

013102	013746	002606		MOV	PORTDIR,-(SP)	::SAVE PORTDIR FOR TYPEOUT	
013106	104403			TYPOS		::GO TYPE--OCTAL ASCII	
013110	006			.BYTE	6	::TYPE 6 DIGITS	
013111	000			.BYTE	0	::SUPPRESS LEADING ZEROS	
2536	013112	112737	000060	115753	.DSABL	CRF	
2537	013120				MOVB	#'0,MSGAB7	
	013120	012700	000001		FOR R0 := #1 TO SLAVES		
	013124						MOV #1,R0
2538	013124	010001					B20::::::::::
2539	013126	006301			MOV R0,R1		
2540	013130	105237	115753		ASL R1		
2541	013134				INCB	MSGAB7	
	013134	104401	115745		TYPE	MSG087	
					TYPEIT	MSG087	
2542	013140				.DSABL	CRF	
	013140	016146	002606		TYPCS	PORTDIR(R1)	
	013144	104403			MOV	PORTDIR(R1),-(SP)	::SAVE PORTDIR(R1) FOR TYPEOUT
	013146	006			TYPOS		::GO TYPE--OCTAL ASCII
	013147	000			.BYTE	6	::TYPE 6 DIGITS
					.BYTE	0	::SUPPRESS LEADING ZEROS
2543	013150				.DSABL	CRF	
	013150	005200			END ;OF FOR R0		
	013152	020037	002730				INC R0
	013156	003762					CMP R0,SLAVES
	013160						BLE B20
2544	013160				END ;OF IF MULTIPORT		E20::::::::::
2545	013160						L134::::::::::
2546	013160				SUBAAR: SUBTST <<LOAD CSR'S FROM E-TABLE \$CDW1 & \$CDW2>>		
					:*****		
					:*SUBTEST LOAD CSR'S FROM E-TABLE \$CDW1 & \$CDW2		
					:*****		
2547	013160				IF \$CDW1 NE #0 OR \$CDW2 NE #0		
	013160	005737	075214				TST \$CDW1
	013164	001003					BNE L135
	013166	005737	075216				TST \$CDW2
	013172	001430					BEQ L136
	013174						L135::::::::::
2548	013174	013701	002334		MOV	MKCSRS,R1	
2549	013200	013737	075214	002144	MOV	\$CDW1,CSR	
2550	013206	013737	075216	002146	MOV	\$CDW2,CSR+2	
2551	013214				FOR CSRNO := #0 TO #34 BY #2		
	013214	005037	002256				CLR CSRNO
	013220						B21::::::::::
2552	013220	106301			ASLB	R1	
2553	013222				ON.ERROR		
	013222	103001					BCC L137
2554	013224	104425			LOADCSR		
2555	013226				END ;OF ON.ERROR		
	013226						L137::::::::::
2556	013226				END ;OF FOR CSRNO		
	013226	062737	000002	002256			ADD #2,CSRNO
	013234	023727	002256	000034			CMP CSRNO,#34
	013242	003766					BLE B21
	013244						E21::::::::::
2557	013244	005237	075142		INC	\$PASS	

2558 013250 000137 015050
2559 013254
013254

JMP APTHANG
END ;OF IF \$CDW1

;GO LOOP FOREVER KEEPING APT ALIVE

L136:;;;;;;

2562 013254

SUBTST <<LEGAL CONFIGURATION CHECK>>

 :SUBTEST LEGAL CONFIGURATION CHECK

2563 013254 023727 002114 000020

IF CPUBIT EQ #MASTER

CMP CPUBIT,#MASTER
 BNE L140

2564 013264 005037 002106

FOR BANK := #0 TO #167

CLR BANK
 B22:*****

2565 013270 004737 042422
 2566 013274 013700 002110

CALL EXBANK
 MOV BANKINDEX,R0

2567
 2568

:MAKE SURE THAT EVERYBODY CAN ACCESS EVERY BANK
 IF MKFLAG IS TRUE

2569 013300 005737 002126
 013300 001472

TST MKFLAG
 BEQ L141

2570

:IF ANYBODY CAN ACCESS
 IF #MASTER!SLAVE1!SLAVE2!SLAVE3 SET.IN CONFIG(R0)

2571 013306 032760 000360 003016
 013314 001466

BIT #MASTER!SLAVE1!SLAVE2!SL
 BEQ L142

2572 013316 013737 002510 002114
 2573 013324 004737 042422

MOV ALLCPUS,CPUBIT
 CALL EXBANK

2574 013330 012737 000020 002114
 2575

MOV #MASTER,CPUBIT
 :IF EVERYBODY CAN'T ACCESS
 IF ACFLAG IS FALSE AND CONFGERROR IS FALSE

2576 013336 005737 002124
 013342 001053

TST ACFLAG
 BNE L143
 TST CONFGERROR
 BNE L143

013344 005737 002576
 013350 001050

SET CONFGERROR

2577 013352 012737 177777 002576

CLEAR SLFLAG,SLFLAG+2,SLFLAG+4,SLFLAG+6

MOV #-1,CONFGERROR

2578 013360 005037 002656
 013364 005037 002660

CLR SLFLAG
 CLR SLFLAG+2
 CLR SLFLAG+4
 CLR SLFLAG+6

013370 005037 002662
 013374 005037 002664

IF #MASTER SET.IN CONFIG(R0) THEN LET SLFLAG := ONES

2579 013400 032760 000020 003016
 013406 001403

BIT #MASTER,CONFIG(R0)
 BEQ L144
 MOV ONES,SLFLAG

013410 013737 002726 002656
 013416

IF #SLAVE1 SET.IN CONFIG(R0) THEN LET SLFLAG+2 := ONES

2580 013416 032760 000040 003016
 013424 001403

L144:*****
 BIT #SLAVE1,CONFIG(R0)
 BEQ L145
 MOV ONES,SLFLAG+2

013426 013737 002726 002660
 013434

IF #SLAVE2 SET.IN CONFIG(R0) THEN LET SLFLAG+4 := ONES

2581 013434 032760 000100 003016
 013442 001403

L145:*****
 BIT #SLAVE2,CONFIG(R0)
 BEQ L146
 MOV ONES,SLFLAG+4

013444 013737 002726 002662
 013452

IF #SLAVE3 SET.IN CONFIG(R0) THEN LET SLFLAG+6 := ONES

2582 013452 032760 000200 003016
 013460 001403

L146:*****
 BIT #SLAVE3,CONFIG(R0)
 BEQ L147
 MOV ONES,SLFLAG+6

013462 013737 002726 002664
 013470

L147:*****

```

2583 013470 104063
2584 013472
      013472
2585 013472
      013472
2586 013472
      013472
2587
2588
2589 013472 004737 042422
2590 013476
      013476 005737 002124
      013502 001440
      013504 005737 002126
      013510 001435
2591 013512 016001 003016
2592 013516 042701 177770
2593 013522
      013522 032760 010000 003020
      013530 001001
2594 013532 006301
2595 013534
      013534
2596 013534
      013534 005701
      013536 001001
      013540 005201
      013542
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609 013542 005004
2610 013544 116002 003022
2611 013550 012703 000010
2612 013554 006002
2613 013556 005504
2614 013560 077303
2615
2616
2617 013562
      013562 020401
      013564 001407
      013566 005737 002576
      013572 001004
2618 013574
      013574 012737 177777 002576
2619 013602 104064
2620 013604
    
```

```

      ERROR +63
      END ;OF IF ACFLAG
      END ;OF IF #MASTER!SLAVE1!SLAVE2!SLAVE3
      END ;OF IF MKFLAG IS TRUE
;MAKE SURE (IF I CAN) THAT EACH BOX HAS ONE & ONLY ONE CSR
CALL      EXBANK
IF ACFLAG IS TRUE AND MKFLAG IS TRUE
      TST ACFLAG
      BEQ L150
      TST MKFLAG
      BEQ L150
MOV      CONFIG(R0),R1
BIC      #^C7,R1          ;R1 = 0 OR 1 OR 2 OR 4
IF #BIT12 OFF.IN CONFIG+2(R0) ;IF NO INTERNAL INTERLEAVE
      BIT #BIT12,CONFIG+2(R0)
      BNE L151
      ASL      R1          ;R1 = 0 OR 2 OR 4
      END ;OF IF #BIT3
      IF R1 EQ #0 THEN LEI R1 := R1 + #1
      TST R1
      BNE L152
      INC R1
      L151:
      L152:
;NOW
;INTERLEAVE
;-----
;4 TO 1 EXT & INTERNAL      4      8
;4 TO 1 EXT & NO INT        4      4
;2 TO 1 EXT & INTERNAL      2      4
;2 TO 1 EXT & NO INT        2      2
;      NO EXT & INTERNAL    1      2
;      NO EXT & NO INT      1      0
;SO
;R1 = 1 OR 2 OR 4 = "# OF CSR'S ON THIS BANK"
CLR      R4
MOVB     CONFIG+4(R0),R2
MOV      #8.,R3
ROR      R2          ;COUNT CSR'S IN THIS BANK
ADC      R4
SOB     R3,1$
;IF # OF CSR'S IS WRONG!
IF R4 NE R1 AND CONFGERROR IS FALSE
      CMP R4,R1
      BEQ L153
      TST CONFGERROR
      BNE L153
      SET      CONFGERROR
      MOV      #-1,CONFGERROR
      ERROR +64
      END ;OF IF R4
    
```

2621 013604 END ;OF IF ACFLAG
2622 013604 END ;OF FOR BANK
 013604 005237 002106
 013610 023727 002106 000167
 013616 003624
2623 013620 END ;OF IF CPUBIT
 013620

L153:.....

L150:.....

INC BANK
CMP BANK,#167
BLE B22

E22:.....

L140:.....

2626 013620
2627 013620 005037 002466
2628 013624 005037 002470
2629 013630 005001
2630 013632 033761 002114 003016
2631 013642 032761 060000 003016
2632 013656 032761 040000 003016
2633 013676 005237 002470
2634 013702
2635 013702
2636 013702 062701 000006
2637 013714 006337 002466
2638 013720 006337 002466
2639 013724 006337 002466
2640 013730 006337 002466
2641 013734 006337 002470
2642 013740 006337 002470
2643 013744 006337 002470
2644 013750 006337 002470
2645 013754 005237 002516
2646 013760 006337 002516
2647 013764 006337 002516
2648 013770 006337 002516
2649 013774 006337 002516
2650 014000 013737 002466 002602
2651 014014 023737 002602 002516
2652 014032 104401 003010

```
SUBSTST <<PRINT CONFIGURATION DETAILS>>
:*****
:*SUBTEST PRINT CONFIGURATION DETAILS
:*****
CLR MJSIZE
CLR MKSIZE
FOR R1 := #0 TO #167*3*2 BY #6
    CLR R1
    B23:*****
    IF CPUBIT SET.IN CONFIG(R1)
        BIT CPUBIT,CONFIG(R1)
        BEQ L154
        IF #BIT14!BIT13 OFF.IN CONFIG(R1) THEN LET MJSIZE := MJSIZE + #1
        BIT #BIT14!BIT13,CONFIG(R1)
        BNE L155
        INC MJSIZE
        L155:*****
        IF #BIT14 OFF.IN CONFIG(R1) AND #BIT13 SET.IN CONFIG(R1)
            BIT #BIT14,CONFIG(R1)
            BNE L156
            BIT #BIT13,CONFIG(R1)
            BEQ L156
            LET MKSIZE := MKSIZE + #1
            INC MKSIZE
        END ;OF IF MK11 MEMORY
    END ;OF IF ACCESSABLE BY CPU
END ;OF FOR ALL BANKS IN TABLE
    ADD #6,R1
    CMP R1,#167*3*2
    BLE B23
    E23:*****
    ASL MJSIZE
    ASL MJSIZE
    ASL MJSIZE
    ASL MJSIZE ;MJSIZE := MJSIZE * 16.
    ASL MKSIZE
    ASL MKSIZE
    ASL MKSIZE ;MKSIZE := MKSIZE * 16.
    INC UNITOP
    ASL UNITOP
    ASL UNITOP
    ASL UNITOP
    ASL UNITOP ;UNITOP := UNITOP * 16.
    LET I := MJSIZE + MKSIZE
    IF I LT UNITOP THEN LET I := UNITOP
    MOV MJSIZE,I
    ADD MKSIZE,I
    CMP I,UNITOP
    BGE L157
    MOV UNITOP,I
    L157:*****
TYPE $CRLF
TYPEIT $CRLF
.DSABL CRF
```

```

2653 014036      TYPDEC I
      014036 013746 002602  MOV I,-(SP) ;;SAVE I FOR TYPEOUT
      014042 104405      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
2654 014044      TYPE MSG070
      014044 104401 115246  TYPEIT ,MSG070
      .DSABL CRF
2655 014050      TYPDEC MJSIZE
      014050 013746 002466  MOV MJSIZE,-(SP) ;;SAVE MJSIZE FOR TYPEOUT
      014054 104405      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
2656 014056      TYPE MSG071
      014056 104401 115271  TYPEIT ,MSG071
      .DSABL CRF
2657 014062      TYPDEC MKSIZE
      014062 013746 002470  MOV MKSIZE,-(SP) ;;SAVE MKSIZE FOR TYPEOUT
      014066 104405      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
2658 014070      TYPE MSG072
      014070 104401 115304  TYPEIT ,MSG072
      .DSABL CRF
2659 014074      IF #SW6 SET.IN @SWR OR CONFGERROR IS TRUE
      014074 032777 000100 166664
      014102 001003
      014104 005737 002576
      014110 001402
      014112
2660 014112 004737 035402      CALL PCONFIG
2661 014116      END ;OF IF #SW6
      014116

```

```

BIT #SW6,@SWR
BNE L160
TST CONFGERROR
BEQ L161
L160:::
L161:::

```

2664 014116

SUBTST <<CHECK APT SIZING>>

 :*SUBTEST CHECK APT SIZING

IF APTFLAG IS TRUE AND APTSIZE IS TRUE

2665 014116 005737 002444
 014122 001505
 014124 005737 002544
 014130 001502
 2666 014132 005037 002534
 2667 014136 012700 075164
 2668 014142 005002
 014144
 2669 014144 105760 000001
 014150 001441
 2670 014152 111001
 2671 014154 042701 177400
 2672 014160 005760 000002
 014164 002002
 2673 014166 000261
 2674 014170 000401
 014172
 2675 014172 000241
 2676 014174
 014174
 2677 014174 006101
 2678 014176 005201
 2679 014200 006301
 2680 014202 006301
 2681 014204 006301
 2682 014206 006301
 2683 014210 163701 002534
 2684 014214 010137 002534
 2685 014220 126027 000001 000001
 014226 001002
 2686 014230 060137 002522
 2687 014234
 014234
 2688 014234 126027 000001 000004
 014242 001002
 2689 014244 060137 002524
 2690 014250
 014250
 2691 014250 062700 000004
 2692 014254
 014254
 2693 014254 005202
 014256 020227 000004
 014262 003730
 014264

CLR TEMP
 MOV #MAMS1,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
 ASL R1
 ASL R1
 ASL R1
 ASL R1
 SUB TEMP,R1
 MOV R1,TEMP
 IFB 1(R0) EQ #1

ADD R1,APTCORE
 END ;OF IFB 1(R0)

IFB 1(R0) EQ #4

ADD R1,APTMOS
 END ;OF IFB 1(R0)

ADD #4,R0
 END ;OF IFB 1(R0)

END ;OF FOR R2

TST APTFLAG
 BEQ L162
 TST APTSIZE
 BEQ L162

CLR R2
 B24:::~::~

TSTB 1(R0)
 BEQ L163

TST 2(R0)
 BGE L164

BR L165
 L164:::~::~

L165:::~::~

;TO COMPENSATE FOR 4 BANKS BEING (0-3)

CMPB 1(R0),#1
 BNE L166

L166:::~::~

CMPB 1(R0),#4
 BNE L167

L167:::~::~

L163:::~::~

INC R2
 CMP R2,#4
 BLE B24

E24:::~::~

```

2694 014264          IF APTCORE NE MJSIZE OR APTMOS NE MKSIZE
      014264 023737 002522 002466
      014272 001004
      014274 023737 002524 002470
      014302 001401
      014304
2695 014304 104051      ERROR +51
2696 014306          END ;OF IF APTCORE
      014306
2697
2698
2699 014306 013700 002730      ;CHECK FOR CORRECT NUMBER OF CPU'S
2700 014312 005200      MOV SLAVES,RO
2701 014314          INC RO ;RO = 1, 2, 3, OR 4
      014314 023700 075234      IF $DDW6 NE RO
      014320 001406
2702 014322 010037 002050
2703 014326 013737 075234 002056
2704 014334 104065
2705 014336          MOV RO,GOOD
      014336          MOV $DDW6,BAD
      014336          ERROR +65
2706 014336          END ;OF $DDW6
      014336          END ;OF IF APTFLAG
2707
2708 014336 004737 015104      CALL DOBACK ;WRITE BACKGROUND PATTERN

      CMP APTCORE,MJSIZE
      BNE L170
      CMP APTMOS,MKSIZE
      BEQ L171
L170:;;;;;;
L171:;;;;;;
      CMP $DDW6,RO
      BEQ L172
L172:;;;;;;
L162:;;;;;;

```

2710 014342
2711 014344 000004 002332
2712 014350 017700 166412
2713 014354 042700 177761
2714 014360 004770 014370
2715 014364 000137 014410
2716 014370 015204
2717 014372 015320
2718 014374 015434
2719 014376 015572
2720 014400 015730
2721 014402 016056
2722 014404 016246
2723 014406 016404
2724
2725

```
LOOP: NEWTST <<DIAGNOSTIC MODE DISPATCH ROUTINE>>
:*****
:*TEST 5      DIAGNOSTIC MODE DISPATCH ROUTINE
:*****
TST5:  SCOPE
      CLR      CONTFLAG
      MOV      @SWR,R0      ;GET SWITCHES
      BIC      #^C16,R0    ;MASK TO ONLY MODE BITS
      CALL     @DISPTBL(R0) ;DISPATCH TO ROUTINE THROUGH NEXT TABLE
      JMP      MEMDONE     ;GO TO NEXT TEST
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
      ;**NOTE** MARGINS FOLLOW PATTERNS (FORWARD IS (0,2,3,4,5))
      ;(REVERSE IS (5,4,3,2,0))
```

```

2729 014410 004737 015104 MEMDONE:CALL DOBACK ;CHECK BACKGROUND PATTERN
2730 014414 NEWTST<<UNIQUE BANK TEST>>
:*****
:*TEST 6 UNIQUE BANK TEST
:*****
2731 014414 000004 TST6: SCOPE
2732 :MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
2733 014416 005737 002002 :WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
014422 001015 IF SELONLY IS FALSE
2734 014424 SET HEADER,MUT TST SELONLY
014424 012737 177777 002724 BNE L173
014432 012737 177777 002116 MOV #-1,HEADER
2735 014440 004737 024370 MOV #-1,MUT
2736 014444 012737 177777 002724 CALL MT0027
014444 005037 002116 SET HEADER MOV #-1,HEADER
2737 014452 CLR MUT
2738 014456 END ;OF IF SELONLY
014456 L173:::::
2739
2740 014456 NEWTST <<FORCE ADDRESS PARITY ERRORS>>
:*****
:*TEST 7 FORCE ADDRESS PARITY ERRORS
:*****
2741 014456 000004 TST7: SCOPE
2742 :THIS MAKES SURE THAT ALL ACCESSABLE MEMORY LOCATIONS WILL GIVE A
2743 :PARITY TRAP IF I FORCE A MEMORY BUSS ADDRESS PARITY ERROR BY
2744 :USING THE MAINTENANCE REGISTER.
2745 :
2746 :WHEN DONE EVERY ADDRESS PARITY ERROR LED IN EVERY MEMORY BOX
2747 : (MK11'S, MKA11'S, AND MJ11'S) SHOULD BE LITE.
2748 014460 005737 002002 :YOU CAN ONLY TURN THEM OFF BY TURNING OFF POWER TO THE BOX'S.
014460 001040 IF SELONLY IS FALSE TST SELONLY
2749 014466 012700 060000 BNE L174
2750 014472 012737 000004 002100 MOV #FIRST,R0
2751 014500 004537 043710 MOV #4,NOPAR ;INDICATE PARITY ACTION
014504 014570 BMOV ADDPAR
JSR R5,BLOCK1
ADDPAR
.DSABL CRF
FOR BANK := #0 TO #167
2752 014506 005037 002106 CLR BANK
014506 004737 042422 B25:::::
014512 CALL EXBANK
2753 014512 005737 002124 IF ACFLAG IS TRUE
014522 001411 TST ACFLAG
2755 014524 005037 002074 BEQ L175
2756 014530 004737 026042 CLR PARCNT
2757 014534 023727 002074 000001 CALL SUPD01
014542 001401 IF PARCNT NE #1
014544 104035 CMP PARCNT,#1
2758 014546 BEQ L176
2759 014546 ERROR +35
END ;OF IF PARCNT
L176:::::
    
```

```
2760 014546          END ;OF IF ACFLAG          L175:.....
      014546
2761 014546          END ;OF FOR BANK
      014546 005237 002106
      014552 023727 002106 000167          INC BANK
      014560 003754          CMP BANK,#167
      014562          BLE B25
2762 014562 005037 002100          CLR NOPAR          ;INDICATE PARITY ACTION
2763 014566          END ;OF IF SELONLY          E25:.....
      014566          BR SUBAAU          L174:.....
2764 014566 000405
2765
2766          ;THESE 3 INSTRUCTIONS ARE MOVED TO THE USER PAR'S IN ORDER TO
2767          ;CAUSE ABORTS FROM THE BANK UNDER TEST VICE THE PROGRAM SPACE
2768 014570 012737 000002 177750 ADDPAR: MOV #BIT1,MAINT          ;V177640          ;CREATE ADDRESS PARITY ERROR
2769 014576 005710          TST (R0)          ;V177646          ;THIS WILL ABORT & INCREMENT PARCNT
2770 014600 000207          RETURN          ;V177650
```

```
2773 014602          SUBAAU: IF SBFLAG IS TRUE
      014602 005737 002022
      014606 001405
2774 014610          IF LASTERROR EQ $ERTTL
      014610 023737 002020 002754
      014616 001001
2775 014620 104050      ERROR      +50
2776 014622          END ;OF IF LASTERROR
      014622
2777 014622          END ;OF IF SBFLAG
      014622
2778
2779 014622          SET      WRITEONLY
      014622 012737 177777 002502
2780 014630 005137 002744
2781 014634 004737 015104
2782 014640 005037 002502
2783 014644          IF MULTIORT IS TRUE THEN JUMPTO MAST01
      014644 005737 003014
      014650 001402
      014652 000137 057372
2784 014656          FLUSH: SUBTST <<FLUSH OUT DBE'S>>
      014656          :*****
      :*SUBTEST      FLUSH OUT DBE'S
      :*****
2785 014656 004737 024754          CALL      MT0030
```

```
TST SBFLAG
BEQ L177
CMP LASTERROR,$ERTTL
BNE L200
L200:
L177:
MOV #-1,WRITEONLY
;RESTORE BACKGROUND PATTERN
TST MULTIORT
BEQ L201
JMP MAST01
L201:
```

```

2789          .SBTTL  END OF PASS ROUTINE
2790          :*****
2791          :*INCREMENT THE PASS NUMBER ($PASS)
2792          :*INDICATE END-OF-PROGRAM AFTER EACH PASSES THRU THE PROGRAM
2793          :*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
2794          :*IF THERES A MONITOR GO TO IT
2795          :*IF THERE ISN'T JUMP TO LOOP
2796 014662 005037 002022 $EOP: CLR SBFLAG
2797 014666 005037 002542 CLR FSINFLAG
2798 014672 013737 002754 002020 MOV $ERTTL, LASTERROR
2799 014700 005237 075142 INC $PASS ;: INCREMENT THE PASS NUMBER
2800 014704 042737 100000 075142 BIC #100000, $PASS ;: DON'T ALLOW A NEG. NUMBER
2801 014712 MSG077 ;: TYPE 'END PASS #'
      014712 104401 115414 TYPEIT ,MSG077
      .DSABL CRF
      IF #SW11 SET. IN @SWR OR QVFLAG IS TRUE
2802 014716
      014716 032777 004000 166042 BIT #SW11, @SWR
      014724 001003 BNE L202
      014726 005737 002440 TST QVFLAG
      014732 001404 SEQ L203
      014734 L202:*****
2803 014734 TYPE MSG035 ;: QV
      014734 104401 113707 TYPEIT ,MSG035
      .DSABL CRF
      CLR QVFLAG
      END ;: OF IF SW11
      L203:*****
2804 014740 005037 002440 TYPDEC $PASS
2805 014744 MOV $PASS, -(SP) ;: SAVE $PASS FOR TYPEOUT
      014744 013746 075142 TYPDS ;: GO TYPE--DECIMAL ASCII WITH SIGN
      014750 104405 .DSABL CRF
2807 014752 013700 000042 MOV 42, R0 ;: GET MONITOR ADDRESS
2808 014756 001430 BEQ $DOAGAIN ;: BRANCH IF NO MONITOR
2809 014760 022700 002000 $ZAP42: CMP #STACK, R0 ;: ARE WE UNDER RT11
2810 014764 001425 BEQ $DOAGAIN ;: YES - BRANCH
2811 ;: WE ARE UNDER (HEAVEN HELP US) XXDP!
2812 014766 010046 PUSH R0 ;: MOV R0, -(SP)
      014766 004737 043554 CALL SHUTUP
2813 014770 004737 043554 POP R0 ;: MOV (SP)+, R0
2814 014774 012600
2815 014776 000005 RESET ;: CLEAR THE WORLD
2816 015000 004710 $ENDAD: CALL (R0) ;: GO TO MONITOR
2817 015002 000240 NOP ;: SAVE ROOM
2818 015004 000240 NOP ;: FOR
2819 015006 000240 NOP ;: ACT11
2820 015010 $DOAGN: ;: UNDO SHUTUP STUFF
2821 ;: RESTORE STACK
2822 ;: ENERGIZE UNIBUS MAP & 22 BIT ADDRESSING
2823 ;: ENERGIZE MEMORY MANAGEMENT
2824 ;: PUT LOADERS BACK HOME
2825 015010 013706 002706 MOV KSTACK, SP
2826 015014 052737 000060 172516 BIS #BIT5!BIT4, MMR3
2827 015022 104420 ENERGIZE ;: TURN ON MEMORY MANAGEMENT
2828 015024 013700 002710 MOV LOADHOME, R0 ;: DESTINATION BANK
2829 015030 012701 000001 MOV #1, R1 ;: SOURCE BANK
2830 015034 004737 042072 CALL BANKMOV
    
```

```

2831 015040          $DOAGAIN:IF $USWR EQ $PASS          ;IF APT LIMITING # OF PASSES
      015040 023737 075160 075142          CMP $USWR,$PASS
      015046 001014          BNE L204
2832 015050 012701 000050          APTHANG: MOV #50,R1
2833 015054 077001          2$: SOB R0,2$
2834 015056 062737 000001 075144          ADD #1,$DEVCT
2835 015064 005537 075146          ADC $UNIT
2836 015070 077107          SOB R1,2$
2837 015072 005237 075142          INC $PASS
2838 015076 000764          BR APTHANG
2839 015100          END ;OF IF $USWR
      015100
2840 015100 000137 014342          JMP LOOP          ;RETURN

```

L204:::~::~

```

2843 015104      DOBACK: SUBTST <<WRITE BACKGROUND PATTERNS>>
:*****
:*SUBTEST      WRITE BACKGROUND PATTERNS
:*****
2844 015104 005037 002120      CLR      PATTERN
2845 015110      FOR BANK := #0 TO #167
      015110 005037 002106
      015114
2846 015114 004737 042422      CALL EXBANK
2847 015120      IF ACFLAG IS TRUE AND PFLAG IS FALSE
      015120 005737 002124
      015124 001420
      015126 005737 002130
      015132 001015
2848 015134      SET      HEADER,MUT
      015134 012737 177777 002724
      015142 012737 177777 002116
2849 015150 004737 021222      CALL      MKTEST      ;CALL MJTEST WOULD ALSO WORK
2850 015154 005037 002116      CLR      MUT
2851 015160      SET      HEADERR
      015160 012737 177777 002724
2852 015166      END ;OF IF ACFLAG
      015166
2853 015166      END ;OF FOR BANK
      015166 005237 002106
      015172 023727 002106 000167
      015200 003745
      015202
2854 015202 000207      RETURN
    
```

CLR BANK
 B26:::~::~

TST ACFLAG
 BEQ L205
 TST PFLAG
 BNE L205

MOV #-1,HEADER
 MOV #-1,MUT

MOV #-1,HEADERR

L205:::~::~

INC BANK
 CMP BANK,#167
 BLE B26

E26:::~::~

2857
2858
2859 015204

.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$:   CLR      MARGIN     ;SET MARGIN TO 0
      ;START OF MARGIN LOOP
3$:   CALL     MTEST      ;GO TEST CORRECT MEMORY
      ;TERMINATION OF MARGIN LOOP
      CALL     INCMAR     ;NEXT HIGHER MARGIN
      BNE      3$         ;IF NOT DONE - LOOP ON THIS MARGIN
      ;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$:   CALL     INCBNK     ;NEXT HIGHER BANK
      BNE      1$         ;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
```

2860 015204 005037 002106
2861
2862 015210 004737 042422
2863 015214 005737 002124
2864 015220 001417
2865 015222 005737 002132
2866 015226 001014
2867 015230 005037 002120
2868
2869 015234 005037 002112
2870
2871 015240 004737 016564
2872
2873 015244 004737 043006
2874 015250 001373
2875
2876 015252 004737 042746
2877 015256 001366
2878
2879 015260 004737 042772
2880 015264 001351
2881
2882 015266 005737 002134
2883 015272 001401
2884 015274 000207
2885 015276 004737 041130
2886 015302
 015302 103001
 015304 000207
 015306
2887
2888 015306 004737 015204
2889 015312 004737 041604
2890 015316 000207

BCC L206
RTS PC
L206:::~::~

```
:**NOTE** RECURSIVE CALL
CALL BAFPAF      ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN
```

2893 015320

BAFPAR: SUBTST <<BANKS FORWARD,PATTERNS REVERSE **RECURSIVE**>>
 :*****
 :*SUBTEST BANKS FORWARD,PATTERNS REVERSE **RECURSIVE**
 :*****

2894 015320 005037 002106

CLR BANK ;SET BANK TO 0

2895

;START OF BANK LOOP

2896 015324 004737 042422

1\$:

CALL EXBANK ;EXAMINE BANK

2897 015330 005737 002124

TST ACFLAG ;CAN WE ACCESS THIS BANK?

2898 015334 001417

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

2899 015336 005737 002132

TST RRFLAG ;RELOCATION REQUIRED?

2900 015342 001014

BNE 4\$;YES - GO TO BANK LOOP TERMINATION

2901 015344 004737 042762

CALL SETPAT ;SET HIGH PATTERN FOR CORRECT MEMORY

2902

;START OF PATTERN LOOP

2903 015350 004737 043212

2\$:

CALL SETMAR ;SET HIGH MARGINS (IF NOT INHIBITED)

2904

;START OF MARGIN LOOP

2905 015354 004737 016564

3\$:

CALL MTEST ;GO TEST CORRECT MEMORY

2906

;TERMINATION OF MARGIN LOOP

2907 015360 004737 043152

CALL DECMAR ;NEXT LOWER MARGIN

2908 015364 100373

BPL 3\$;IF NOT DONE - LOOP ON THIS MARGIN

2909

;TERMINATION OF PATTERN LOOP

2910 015366 005337 002120

DEC PATTERN ;IS THIS THE LAST PATTERN?

2911 015372 100366

BPL 2\$;NO - LOOP ON THIS PATTERN

2912

;TERMINATION OF BANK LOOP

2913 015374 004737 042772

4\$:

CALL INCBNK ;NEXT HIGHER BANK

2914 015400 001351

BNE 1\$;IF NOT DONE - LOOP ON THIS BANK

2915

;END OF LOOPS

2916 015402 005737 002134

TST RLFLAG ;HAVE WE BEEN RELOCATED?

2917 015406 001401

BEQ 5\$;NO - SKIP

2918 015410 000207

RETURN ;YES - RETURN

2919 015412 004737 041130

5\$:

CALL RELOCATE ;MOVE & MAP PROGRAM

2920 015416

ON.ERROR THEN \$RETURN

015416 103001

015420 000207

015422

2921

;**NOTE** RECURSIVE CALL

2922 015422 004737 015320

CALL BAFPAR ;CALL SELF

2923 015426 004737 041604

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

2924 015432 000207

RETURN

BCC L207
 RTS PC

L207:::~::~

2927 015434
 2928 015434 005037 002106
 2929
 2930 015440 004737 042422
 2931 015444 005737 002124
 2932 015450 001422
 2933 015452 005737 002136
 2934 015456 001417
 2935 015460 005737 002132
 2936 015464 001014
 2937 015466 005037 002120
 2938
 2939 015472 005037 002112
 2940
 2941 015476 004737 016564
 2942
 2943 015502 004737 043006
 2944 015506 001373
 2945
 2946 015510 004737 042746
 2947 015514 001366
 2948
 2949 015516 004737 042772
 2950 015522 001346
 2951
 2952 015524 005137 002712
 2953 015530 001003
 2954
 2955 015532 004737 015434
 2956 015536 000207
 2957 015540 005737 002134
 2958 015544 001401
 2959 015546 000207
 2960 015550 004737 041130
 2961 015554
 015554 103001
 015556 000207
 015560
 2962
 2963 015560 004737 015434
 2964 015564 004737 041604
 2965 015570 000207

```

BAWPAF: SUBST <<BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST     BANKS WORST FIRST,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     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$:      CLR     MARGIN        ;SET MARGIN TO 0
          ;START OF MARGIN LOOP
3$:      CALL    MTEST         ;GO TEST CORRECT MEMORY
          ;TERMINATION OF MARGIN LOOP
          CALL    INCMAR        ;NEXT HIGHER MARGIN
          BNE     3$            ;IF NOT DONE - LOOP ON THIS MARGIN
          ;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$:      CALL    INCBNK        ;NEXT HIGHER BANK
          BNE     1$            ;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
    
```

BCC L210
 RTS PC
 L210:::~::~

2968 015572
 2969 015572 005037 002106
 2970
 2971 015576 004737 042422
 2972 015602 005737 002124
 2973 015606 001422
 2974 015610 005737 002136
 2975 015614 001417
 2976 015616 005737 002132
 2977 015622 001014
 2978 015624 004737 042762
 2979
 2980 015630 004737 043212
 2981
 2982 015634 004737 016564
 2983
 2984 015640 004737 043152
 2985 015644 100373
 2986
 2987 015646 005337 002120
 2988 015652 100366
 2989
 2990 015654 004737 042772
 2991 015660 001346
 2992
 2993 015662 005137 002712
 2994 015666 001003
 2995
 2996 015670 004737 015572
 2997 015674 000207
 2998 015676 005737 002134
 2999 015702 001401
 3000 015704 000207
 3001 015706 004737 041130
 3002 015712
 015712 103001
 015714 000207
 015716
 3003
 3004 015716 004737 015572
 3005 015722 004737 041604
 3006 015726 000207

```

BAWPAR: SUBTST <<BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**>>
:*****
:*SUBTEST     BANKS WORST FIRST,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     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    SETPAT         ;SET HIGH PATTERN FOR CORRECT MEMORY
          ;START OF PATTERN LOOP
2$:      CALL    SETMAR         ;SET HIGH MARGINS (IF NOT INHIBITED)
          ;START OF MARGIN LOOP
3$:      CALL    MTEST          ;GO TEST CORRECT MEMORY
          ;TERMINATION OF MARGIN LOOP
          CALL    DECMAR        ;NEXT LOWER MARGIN
          BPL     3$             ;IF NOT DONE - LOOP ON THIS MARGIN
          ;TERMINATION OF PATTERN LOOP
          DEC     PATTERN        ;IS THIS THE LAST PATTERN?
          BPL     2$             ;NO - LOOP ON THIS PATTERN
          ;TERMINATION OF BANK LOOP
4$:      CALL    INCBNK         ;NEXT HIGHER BANK
          BNE     1$             ;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    BAWPAR         ;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    BAWPAR         ;CALL SELF
          CALL    UNRELOCATE     ;UNMOVE & UNMAP PROGRAM
          RETURN
    
```

BCC L211
 RTS PC
 L211:::~::~

3010 015730

PAFBAF: SUBST <<PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**>>
 :*****
 :*SUBTEST PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**
 :*****

3011 015730 005037 002120

CLR PATTERN ;SET PATTERN TO 0

3012

;START OF PATTERN LOOP

3013 015734 005037 002106

1\$: CLR BANK ;SET BANK TO 0

3014

;START OF BANK LOOP

3015 015740 004737 042422

2\$: CALL EXBANK ;EXAMINE BANK

3016 015744 004737 042730

CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?

3017 015750 001015

BNE 4\$;NO - GO TO BANK LOOP TERMINATOR

3018 015752 005737 002124

TST ACFLAG ;CAN WE ACCESS THIS BANK?

3019 015756 001412

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

3020 015760 005737 002132

TST RRFLAG ;RELOCATION REQUIRED?

3021 015764 001007

BNE 4\$;YES - GO TO BANK LOOP TERMINATION

3022 015766 005037 002112

CLR MARGIN ;SET MARGIN TO 0

3023

;START OF MARGIN LOOP

3024 015772 004737 016564

3\$: CALL MTEST ;GO TEST CORRECT MEMORY

3025

;TERMINATION OF MARGIN LOOP

3026 015776 004737 043006

CALL INCMAR ;NEXT HIGHER MARGIN

3027 016002 001373

BNE 3\$;IF NOT DONE - LOOP ON THIS MARGIN

3028

;TERMINATION OF BANK LOOP

3029 016004 004737 042772

4\$: CALL INCBNK ;NEXT HIGHER BANK

3030 016010 001353

BNE 2\$;IF NOT DONE - LOOP ON THIS BANK

3031

;TERMINATION OF PATTERN LOOP

3032 016012 004737 042746

CALL INCRPT ;NEXT HIGHER PATTERN

3033 016016 001346

BNE 1\$;OK - LOOP; ELSE CGNTINUE

3034

;END OF LOOPS

3035 016020 005137 002140

COM TMFLAG ;COMPLEMENT TYPE OF MEMORY

3036

;IS THIS AN EVEN NUMBER PASS?

3037 016024 001403

BEQ 5\$;YES - SKIP

3038

***NOTE** RECURSIVE CALL

3039 016026 004737 015730

CALL PAFBAF ;CALL SELF

3040 016032 000207

RETURN

3041 016034 005737 002134

5\$: TST RLFLAG ;HAVE WE BEEN RELOCATED?

3042 016040 001401

BEQ 6\$;NO - SKIP

3043 016042 000207

RETURN ;YES - RETURN

3044 016044 004737 041130

6\$: CALL RELOCATE ;MOVE & MAP PROGRAM

3045 016050

ON.ERROR THEN \$RETURN

016050 103001

016052 000207

016054

BCC L212
 RTS PC
 L212:::~::~

3046

***NOTE** RECURSIVE CALL

3047 016054 004737 015730

CALL PAFBAF ;CALL SELF

3048 016060 004737 041604

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

3049 016064 000207

RETURN

3052 016066

PAFBAW: SUBST <<PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**>>
 :*****
 :*SUBTEST PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**
 :*****

3053 016066 005037 002120

CLR PATTERN ;SET PATTERN TO 0

3054

;START OF PATTERN LOOP

3055 016072 005037 002106

1\$: CLR BANK ;SET BANK TO 0

3056

;START OF BANK LOOP

3057 016076 004737 042422

2\$: CALL EXBANK ;EXAMINE BANK

3058 016102 004737 042730

CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?

3059 016106 001020

BNE 4\$;NO - GO TO BANK LOOP TERMINATOR

3060 016110 005737 002124

TST ACFLAG ;CAN WE ACCESS THIS BANK?

3061 016114 001415

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

3062 016116 005737 002136

TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)

3063 016122 001412

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

3064 016124 005737 002132

TST RRFLAG ;RELOCATION REQUIRED?

3065 016130 001007

BNE 4\$;YES - GO TO BANK LOOP TERMINATION

3066 016132 005037 002112

CLR MARGIN ;SET MARGIN TO 0

3067

;START OF MARGIN LOOP

3068 016136 004737 016564

3\$: CALL MTEST ;GO TEST CORRECT MEMORY

3069

;TERMINATION OF MARGIN LOOP

3070 016142 004737 043006

CALL INCMAR ;NEXT HIGHER MARGIN

3071 016146 001373

BNE 3\$;IF NOT DONE - LOOP ON THIS MARGIN

3072

RETURN

3085 016200 005137 002712

000207

5\$: COM WORST ;4TH PASS?

3086 016204 001003

BNE 6\$;YES - SKIP

3087

***NOTE** RECURSIVE CALL

3088 016206 004737 016066

CALL PAFBAW ;CALL SELF

3089 016212 000207

RETURN

3090 016214 005737 002134

6\$: TST RLFLAG ;HAVE WE BEEN RELOCATED?

3091 016220 001401

BEQ 7\$;NO - SKIP

3092 016222 000207

RETURN ;YES - RETURN

3093 016224 004737 041130

7\$: CALL RELOCATE ;MOVE & MAP PROGRAM

3094 016230

ON.ERROR THEN \$RETURN

016230 103001

016232 000207

016234

BCC L213
 RTS PC
 L213:::~::~

3095

***NOTE** RECURSIVE CALL

3096 016234 004737 016066

CALL PAFBAW ;CALL SELF

3097 016240 004737 041604

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

3098 016244 000207

RETURN

```

3101 016246      PARBAF: SUBST  <<PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**>>
:*****
:*SUBTEST      PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**
:*****
3102 016246 004737 042762      CALL  HIPAT      ;SET HIGHEST PATTERNS
3103      ;START OF PATTERN LOOP
3104 016252 005037 002106      1$: CLR  BANK      ;SET BANK TO 0
3105      ;START OF BANK LOOP
3106 016256 004737 042422      2$: CALL  EXBANK   ;EXAMINE BANK
3107 016262 004737 042730      CALL  BANKOK    ;CORRECT MEMORY FOR THIS BANK?
3108 016266 001015      BNE  4$         ;NO - GO TO BANK LOOP TERMINATOR
3109 016270 005737 002124      TST  ACFLAG    ;CAN WE ACCESS THIS BANK?
3110 016274 001412      BEQ  4$         ;NO - GO TO BANK LOOP TERMINATION
3111 016276 005737 002132      TST  RRFLAG    ;RELOCATION REQUIRED?
3112 016302 001007      BNE  4$         ;YES - GO TO BANK LOOP TERMINATION
3113 016304 004737 043212      CALL  SETMAR   ;SET HIGH MARGINS (IF NOT INHIBITED)
3114      ;START OF MARGIN LOOP
3115 016310 004737 016564      3$: CALL  MTEST    ;GO TEST CORRECT MEMORY
3116      ;TERMINATION OF MARGIN LOOP
3117 016314 004737 043152      CALL  DECMAR   ;NEXT LOWER MARGIN
3118 016320 100373      BPL  3$         ;IF NOT DONE - LOOP ON THIS MARGIN
3119      ;TERMINATION OF BANK LOOP
3120 016322 004737 042722      4$: CALL  INCBNK  ;NEXT HIGHER BANK
3121 016326 001353      BNE  2$         ;IF NOT DONE - LOOP ON THIS BANK
3122      ;TERMINATION OF PATTERN LOOP
3123 016330 005337 002120      DEC  PATTERN   ;NEXT LOWER PATTERN
3124 016334 100346      BPL  1$         ;OK - LOOP; ELSE CONTINUE
3125      ;END OF LOOPS
3126 016336 005137 002140      COM  TMFLAG    ;COMPLEMENT TYPE OF MEMORY
3127      ;IS THIS AN EVEN NUMBER PASS?
3128 016342 001403      BEQ  5$         ;YES - SKIP
3129      ;**NOTE** RECURSIVE CALL
3130 016344 004737 016246      CALL  PARBAF   ;CALL SELF
3131 016350 000207      RETURN
3132 016352 005737 002134      5$: TST  RLFLAG   ;HAVE WE BEEN RELOCATED?
3133 016356 001401      BEQ  6$         ;NO - SKIP
3134 016360 000207      RETURN         ;YES - RETURN
3135 016362 004737 041130      6$: CALL  RELOCATE ;MOVE & MAP PROGRAM
3136 016366      ON.ERROR THEN $RETURN
      016366 103001
      016370 000207
      016372
      BCC L214
      RTS PC
      L214:::
3137      ;**NOTE** RECURSIVE CALL
3138 016372 004737 016246      CALL  PARBAF   ;CALL SELF
3139 016376 004737 041604      CALL  UNRELOCATE ;UNMOVE & UNMAP PROGRAM
3140 016402 000207      RETURN
    
```

3143 016404
 3144 016404 004737 042762
 3145
 3146 016410 005037 002106
 3147
 3148 016414 004737 042422
 3149 016420 004737 042730
 3150 016424 001020
 3151 016426 005737 002124
 3152 016432 001415
 3153 016434 005737 002136
 3154 016440 001412
 3155 016442 005737 002132
 3156 016446 001007
 3157 016450 004737 043212
 3158
 3159 016454 004737 016564
 3160
 3161 016460 004737 043152
 3162 016464 100373
 3163
 3164 016466 004737 042772
 3165 016472 001350
 3166
 3167 016474 005337 002120
 3168 016500 100343
 3169
 3170 016502 005137 002140
 3171
 3172 016506 001403
 3173
 3174 016510 004737 016404
 3175 016514 000207
 3176 016516 005137 002712
 3177 016522 001003
 3178
 3179 016524 004737 016404
 3180 016530 000207
 3181 016532 005737 002134
 3182 016536 001401
 3183 016540 000207
 3184 016542 004737 041130
 3185 016546
 016546 103001
 016550 000207
 016552
 3186
 3187 016552 004737 016404
 3188 016556 004737 041604
 3189 016562 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 SETMAR ;SET HIGH MARGINS (IF NOT INHIBITED)
;START OF MARGIN LOOP
3$: CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF MARGIN LOOP
CALL DECMAR ;NEXT LOWER MARGIN
BPL 3$ ;IF NOT DONE - LOOP ON THIS MARGIN
;TERMINATION OF BANK LOOP
4$: CALL INCBNK ;NEXT HIGHER BANK
BNE 2$ ;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
    
```

BCC L215
 RTS PC
 L215:::~::~

```

3193 016564      MTEST: SUBTST <<SUBR SETUP MEMORY TEST>>
:*****
:*SUBTEST      SUBR   SETUP MEMORY TEST
:*****
3194 016564 013700 002112      MOV    MARGIN,R0
3195 016570 006300              ASL    R0
3196 016572 010037 177750      MOV    R0,MAINT      ;SET MARGINS
3197 016576              SET    HEADER        ;INITIALIZE HEADER MESSAGE TYPEOUT
016576 012737 177777 002724      SET    MUT           MOV #-1,HEADER
3198 016604              SET    MUT           ;INDICATE THERE IS A MEMORY UNDER TEST
016604 012737 177777 002116      MOV    #-1,MUT
3199 016612 005037 002362      CLR    PASFLG
3200 016616 005737 002126      TST    MKFLAG        ;MK11?
3201 016622 001410      BEG    2$             ;NO - SKIP
3202              ;IF CONTFLAG IS FALSE THEN CALL MKCONTROL
3203 016624 005737 002332      TST    CONTFLAG
3204 016630 001002      BNE    1$
3205 016632 004737 016664      CALL   MKCONTROL
3206 016636 004737 021222      1$: CALL   MKTEST        ;YES - DO MK11 TESTS
3207 016642 000402      BR     3$
3208 016644 004737 021512      2$: CALL   MJTEST        ;DO MJ11 TESTS
3209 016650 005037 002116      3$: CLR    MUT           ;NOW - NO MEMORY UNDER TEST
3210 016654              SET    HEADER        ;ALLOW HEADERS NORMAL
016654 012737 177777 002724      MOV    #-1,HEADER
3211 016662 000207      RETURN

```

```

3214 016664 MKCONTROL:SUBTST <<SUBR TEST MK11 CONTROLLER LOGIC DISPATCH>>
:*****
:*SUBTEST SUBR TEST MK11 CONTROLLER LOGIC DISPATCH
:*****
3215 :THIS MODULE AND THE NEXT FEW PAGES SOLVE THE PROBLEM OF
3216 :HOW TO RUN THE CSR TESTS ON EACH MK11 MEMORY
3217 :
3218 :IN ORDER TO TEST EACH CONTROLLER (2 PER BOX) I MUST DO
3219 : EACH EXISTING BOX
3220 : IF ECC IS NOT DISABLED
3221 : EACH SIDE OF EACH BOX
3222 : FIND THE ADDRESS OF THIS SIDE (DEPENDS ON INTERLEAVE)
3223 : FIND A LOCATION THAT HAS NO SINGLE BIT ERRORS
3224 : RUN ALL CSR TESTS
3225 : NEXT SIDE
3226 : NEXT BOX
3227 : IF SELONLY IS TRUE THEN $RETURN

016664 005737 002002 TST SELONLY
016664 001401 BEQ L216
016672 000207 RTS PC
016674 L216:*****

3228 016674 PUSH BANK MOV BANK,-(SP)
016674 013746 002106 MOVB #200,CSRINDEX ;SET BIT FOR CONTROLLER
3229 016700 112737 000200 002416 MOVB #1,CSROUDEX ;SET BIT FOR CONTROLLER
3230 016706 112737 000001 002417 FOR CSRLOOP := #'0 TO #'7
3231 016714 012737 000060 002420 MOV #'0,CSRLOOP
016722 B27:*****
3232 016722 IFB CSRINDEX SET.IN MKCSRS BITB CSRINDEX,MKCSRS
016722 133737 002416 002334 BEQ L217
016730 001537
3233 016732 004737 017300 CALL ISECCON
3234 016736 012737 000060 002422 FOR SIDE := #'0 TO #'1
016744 MOV #'0,SIDE
3235 016744 004737 017610 CALL INTKRAP B30:*****
3236 016750 BEGIN CSRSTUFF B31:*****
016750
3237 016750 005037 002424 CLR SUCCESS
3238 016754 013737 002412 002106 FOR BANK := CSRLBANK DOWNT0 CSRFBANK
016762 MOV CSRLBANK,BANK
3239 016762 004737 042422 CALL EXBANK B32:*****
3240 016766 005737 002124 IF ACFLAG IS TRUE AND RRFLAG IS FALSE
016772 001467 TST ACFLAG
016774 005737 002132 BEQ L220
017000 001064 TST RRFLAG
3241 017002 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
017002 010346 MOV R3,-(SP)
017004 013703 002106 MOV BANK,R3
017010 004737 040700 CALL MAPPER
.DSABL CRF
017014 012603 MOV (SP)+,R3
3242 017016 104511 INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON "BANK"
3243 017020 000241 CLC
3244 017022 IF SPECIAL IS TRUE THEN $CALL SIDEOK
  
```

017022	005737	002432				TST SPECIAL
017026	001402					BEG L221
017030	004737	021066				JSR PC,SIDEOK
017034						L221:.....
3245	017034			ON.NOERROR		
017034	103446					BCS L222
3246	017036			FOR TESTADD := CSRFIRST TO #LAST BY CSRINC		MOV CSRFIRST,TESTADD
017036	013737	002336	002506			B33:.....
017044						
3247	017044	004737	017374	CALL SBETEST		
3248	017050			ON.NOERROR		
017050	103431					BCS L223
3249	017052	104424		CACHOFF		;TURN CACHE OFF
3250	017054			FOR I := #0 TO #37		
017054	005037	002602				CLR I
017060						B34:.....
3251	017060	005037	002100			;INDICATE PARITY ACTION
3252	017064			CLR NOPAR		
017064	012737	177777	002724	SET HEADER		MOV #-1,HEADER
3253	017072	005037	002362			
3254	017076			CLR PASFLG		
017076	013700	002602		LET R0 := I		MOV I,R0
3255	017102	004737	020114	CALL CSRCASE		
3256	017106			END ;OF FOR I		
017106	005237	002602				INC I
017112	023727	002602	000037			CMP I,#37
017120	003757					BLE B34
017122						E34:.....
3257	017122	104423		CACHON		;TURN CACHE ON
3258	017124			SET SUCCESS		
017124	012737	177777	002424			MOV #-1,SUCCESS
3259	017132			LEAVE CSRSTUFF		
017132	000415					BR E31
3260	017134			END ;OF ON.NOERROR		
017134						L223:.....
3261	017134			END ;OF FOR TESTADD		
017134	063737	002414	002506			ADD CSRINC,TESTADD
017142	023727	002506	157776			CMP TESTADD,#LAST
017150	003735					BLE B33
017152						E33:.....
3262	017152			END ;OF ON.NOERROR		
017152						L222:.....
3263	017152			END ;OF IF		
017152						L220:.....
3264	017152			END ;OF FOR BANK		
017152	005337	002106				DEC BANK
017156	023737	002106	002410			CMP BANK,CSRFBANK
017164	002276					BGE B32
017166						E32:.....
3265	017166			END CSRSTUFF		
017166						E31:.....
3266	017166			IF SUCCESS IS FALSE		
017166	005737	002424				TST SUCCESS
017172	001010					BNE L224
3267	017174	113737	002420	MOVB CSRLOOP,MSG34		;SETUP CSR NUMBER
3268	017202	113737	002422	MOVB SIDE,MSG34		;SETUP SIDE (0 OR 1)
3269	017210			TYPE MSG034		

```

017210 104401 113571          TYPEIT ,MSG034
                                .DSABL CRF
                                END ;OF IF SUCCESS
3270 017214
017214
3271 017214          END ;OF FOR SIDE
                                L224:.....
                                INC SIDE
                                CMP SIDE,#'1
                                BLE B30
017214 005237 002422
017220 023727 002422 000061
017226 003646
017230
3272 017230          END ;OF IFB CSRINDEX
                                E30:.....
                                L217:.....
3273 017230 000241          CLC
3274 017232 106037 002416      RORB CSRINDEX          ;CHANGE TO NEXT CONTROLLER
3275 017236 106337 002417      ASLB CSROUDEX         ;CHANGE TO NEXT CONTROLLER
3276 017242          END ;OF FOR CSRLOOP
                                INC CSRLOOP
                                CMP CSRLOOP,#'7
                                BLE B27
017242 005237 002420
017246 023727 002420 000067
017254 003622
017256
3277 017256 104472          ECCINIT          ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
3278 017260          SET          CONTFLAG
                                MOV #-1,CONTFLAG
017260 012737 177777 002332
3279 017266          POP          BANK
                                MOV (SP)+,BANK
017266 012637 002106
3280 017272 004737 042422      CALL          EXBANK
3281 017276 000207          RETURN
    
```

3284 017300

ISECCON:SUBTST <<IS ECC ON?>>

 :*SUBTEST IS ECC ON?

3285

:SETUP CSR NUMBER

3286 017300 005000
 3287 017302 113702 002417

CLR R0
 MOV B CSROUDEX,R2

:START A ZERO
 :GET CSR INDEX #2
 :GET CSR BIT IN CARRY (MAYBE)

3288 017306 106002
 3289 017310 103403

1\$:

ROR B R2
 BCS 2\$

:I GOT IT - SKIP
 :BUMP CSR ADDRESS LSB'S BY 4

3290 017312 062700 000004
 3291 017316 000773

ADD #4,R0
 BR 1\$

:LOOP TILL I FIND THE CSR
 :UPDATE CSR NUMBER

3292 017320 010037 002256
 3293 017324 104503

2\$:

MOV R0,CSRNO
 CLR1CSR

:CLEAR 1 SELECTED MK11 CSR

3294 017326 104426
 3295 017330

READCSR
 IF #BIT1 SET.IN CSR

BIT #BIT1,CSR
 BEQ L225

017330 032737 000002 002144
 017336 001413

:WARNING! ECC IS DISABLED ON BOX

3296 017340
 017340 104401 115430

TYPE MSG078
 TYPEIT ,MSG078
 .DSABL CRF

3297 017344 113737 002420 112134
 3298 017352

MOV B CSRLOOP,MSG015
 TYPE MSG015

017352 104401 112134

TYPEIT ,MSG015
 .DSABL CRF

3299 017356
 017356 012737 177777 002474

SET NOECCFLAG

MOV #-1,NOECCFLAG

3300 017364
 017364 000402

ELSE

BR L226

3301 017366 005037 002474
 3302 017372

CLR NOECCFLAG
 END ;OF IF #BIT1

L225:*****

017372
 3303 017372 000207

RETURN

L226:*****

```

3306 017374          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
3331 017374 013701 002506  MOV     TESTADD,R1
3332 017400          SUPERVISOR          ;ENTER SUPERVISOR MODE
      017400 052737 040000 177776  BIS     #BIT14,PSW      ;DO IT TO IT!
      .DSABL CRF
3333 017406 104424          CACHOFF          ;TURN CACHE OFF
3334 017410 104471          ECC1DIS          ;DISABLE ECC ON 1 SELECTED CSR
3335 017412          CLEAR (R1),2(R1)
      017412 005011          CLR (R1)
      017414 005061 000002          CLR 2(R1)
3336 017420 005711          TST (R1)
3337 017422 001061          BNE 1$
3338 017424 005761 000002          TST 2(R1)
3339 017430 001056          BNE 1$
3340
3341 017432 104503          CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
3342 017434          CLEAR (R1),2(R1)
      017434 005011          CLR (R1)
      017436 005061 000002          CLR 2(R1)
3343 017442 005711          TST (R1)
3344 017444 001050          BNE 1$
3345 017446 005761 000002          TST 2(R1)
3346 017452 001045          BNE 1$
3347
3348 017454 104510          TSTREAD          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
3349 017456          IF #BIT15!BIT4 SET.IN CSR THEN GOTO 1$
      017456 032737 100020 002144          BIT #BIT15!BIT4.CSR
      017464 001040          BNE 1$
3350
3351 017466 104471          ECC1DIS          ;DISABLE ECC ON 1 SELECTED CSR
    
```

```

3352 017470 005111          COM      (R1)
3353 017472 005161 000002   COM      2(R1)
3354 017476 023711 002726   CMP      ONES,(R1)
3355 017502 001031          BNE      1$
3356 017504 023761 002726 000002   CMP      ONES,2(R1)
3357 017512 001025          BNE      1$
3358
3359 017514 104503          CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
3360 017516 005011          CLR      (R1)
3361 017520 012761 100000 000002   MOV      #BIT15,2(R1)
3362 017526 005711          TST      (R1)
3363 017530 001016          BNE      1$
3364 017532 022761 100000 000002   CMP      #BIT15,2(R1)
3365 017540 001012          BNE      1$
3366
3367 017542 104510          TSTREAD          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
3368 017544          IF #BIT15!BIT4 SET.IN CSR THEN GOTO 1$
    017544 032737 100020 002144          BIT #BIT15!BIT4,CSR
    017552 001005          BNE 1$
3369
3370 017554 104417          KERNEL          ;ENTER KERNEL MODE
3371 017556 104473          ECC1INIT          ;INITIALIZE 1 SELECTED MK11 CSR
3372 017560 104423          CACHON          ;TURN CACHE ON
3373 017562          $RETURN NOERROR
    017562 000241          CLC
    017564 000207          RTS PC
3374
3375 017566 104503          1$: CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
3376 017570          CLEAR (R1),2(R1)
    017570 005011          CLR (R1)
    017572 005061 000002          CLR 2(R1)
3377 017576 104417          KERNEL          ;ENTER KERNEL MODE
3378 017600 104473          ECC1INIT          ;INITIALIZE 1 SELECTED MK11 CSR
3379 017602 104423          CACHON          ;TURN CACHE ON
3380 017604          $RETURN ERROR
    017604 000261          SEC
    017606 000207          RTS PC
3381
    .DSABL LSB
    
```

```

3384 017610      INTKRAP:SUBST <<SETUP INTERLEAVE INFO>>
:*****
:*SUBTEST      SETUP INTERLEAVE INFO
:*****
3385             :BASED ON THE STARTING ADDRESS, BOX CAPACITY, INTERLEAVE, AND SIDE
3386             :SETUP VARIABLES TO DETERMINE
3387             :(1)  THE STARTING ADDRESS OF THIS CONTROLLER
3388             :(2)  THE NUMBER TO ADD TO EACH ADDRESS TO STAY IN THIS CONTROLLER
3389             :(3)  THE LAST ADDRESS OF THIS CONTROLLER
3390 017610      PUSH      R0
3391 017610      010046             MOV R0,-(SP)
3392 017612      004737 017736      CALL  GETSIZE
3393 017616      013737 002260 002410  MOV  FIRSTB,CSRFBANK
3394 017624      013737 002262 002412  MOV  LASTB,CSRLBANK
3395 017632      005037 002432      CLR   SPECIAL
3396 017636      104426             READCSR ;READ CSR
3397 017640      013700 002146      MOV  CSR+2,R0
3398 017644      072027 177765      ASH  #-11,R0
3399 017650      042700 177760      BIC  #^C17,R0 ;R0 = (0..17) = BITS 14,13,12,11 OF CSR+2
3400 017654      010046             ;IF R0 = 0 CALL NEXT; IF R0 = 1 CALL NEXT+2, ETC.
3401 017656      006316             MOV R0,-(SP)
3402 017660      004737 017724      ASL  @SP
3403 017664      020234             JSR  PC,L227
3404 017666      020260             INT..0
3405 017670      020324             INT..1
3406 017672      020350             INT..2
3407 017674      020414             INT..3
3408 017676      020440             INT..4
3409 017700      020504             INT..5
3410 017702      020530             INT..6
3411 017704      020574             INT..7
3412 017706      020574             INT..10
3413 017710      020616             INT..11
3414 017712      020642             INT..12
3415 017714      020706             INT..13
3416 017716      020732             INT..14
3417 017720      020776             INT..15
3418 017722      021022             INT..16
3419 017724      END                INT..17
3420 017724      062616             ;OF CASE
3421 017726      013646             L227:.....
3422 017730      004736             ADD (SP)+,@SP
3423 017732      012600             MOV @ (SP)+,-(SP)
3424 017734      000207             JSR PC,@(SP)+
3425 017736      POP      R0             MOV (SP)+,R0
3426 017738      RETURN
    
```

3421 017736

GETSIZE:SUBTST <<SUBR GET SIZE FROM CSR>>

 :*SUBTEST SUBR GET SIZE FROM CSR

3422

:SETUP FIRSTB AND LASTB
 PUSH R0

3423 017736

010046

MOV R0,-(SP)

3424 017740

052737

000010

002144

BIS #BIT3,CSR

;GET SET TO READ CSR

3425 017746

042737

000006

002144

BIC #BIT2!BIT1,CSR

3426 017754

104425

LOADCSR

3427 017756

104426

READCSR

3428 017760

042737

000010

002144

BIC #BIT3,CSR

;RESTORE THIS FROM "SIZE" MODE

3429 017766

113700

002145

MOVB CSR+1,R0

;GET SIZE

3430 017772

006300

ASL R0

;R0 = # OF BANKS

3431 017774

110037

002262

MOVB R0,LASTB

3432 020000

013700

002146

MOV CSR+2,R0

3433

:CORRECT LAST BANK FOR EXTERNAL INTERLEAVE

3434 020004

032700

010000

IF #BIT12 SET.IN R0 THEN LET LASTB := LASTB L.SHIFT 1

BIT #BIT12,R0

BEQ L230

ASL LASTB

020010

001402

020012

006337

002262

020016

3435 020016

032700

020000

IF #BIT13 SET.IN R0 THEN LET LASTB := LASTB L.SHIFT 1

020022

001402

020024

006337

002262

020030

3436 020030

042700

177000

BIC #177000,R0

3437 020034

006300

ASL R0

3438 020036

010037

002260

MOV R0,FIRSTB

3439 020042

063737

002260

002262

ADD FIRSTB,LASTB

3440 020050

005337

002262

DEC LASTB

3441 020054

023727

002260

000167

IF FIRSTB HI #167 THEN LET FIRSTB := #170

CMP FIRSTB,#167

BLOS L232

MOV #170,FIRSTB

020062

101403

020064

012737

000170

002260

020072

3442 020072

023727

002262

000167

IF LASTB HI #167 THEN LET LASTB := #167

CMP LASTB,#167

BLOS L233

MOV #167,LASTB

020100

101403

020102

012737

000167

002262

020110

3443 020110

012600

POP R0

020110

000207

RETURN

MOV (SP)+,R0

3444 020112

000207

</

3448 020114
3449 020114 010046
020114 006316
020120 004737 020224

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

MOV RO,-(SP)
ASL @SP
JSR PC,L234

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

			:PAT	TIME	DISCRIPTION
3453	MKCSRT:		MT0006	:<1 SEC	INITIAL DATA TEST
3454	020124	022360	MT0010	:<1 SEC	BYTE ADDRESSING TEST
3455	020126	022456	MT0025	:<1 SEC	INTERRUPT ENABLE TEST
3456	020130	024172	MT0011	:<2 SEC	CREATE SINGLE BIT ERROR TEST
3457	020132	022512	MT0012	:<1 SEC	WRITE BYTE CLEARS SBE TEST
3458	020134	022570	MT0013	: 1 SEC	CREATE DOUBLE BIT ERROR TEST
3459	020136	022646	MT0014	: 1 SEC	WRITE INHIBIT DURING DATIP WITH DBE
3460	020140	022722	MT0015	: 1 SEC	WRITE INHIBIT OF BYTE WITH DBE
3461	020142	023000	MT0016	:<1 SEC	WRITE INHIBIT OF WORD WITH DBE
3462	020144	023056	MT0999	: 0 SEC	NULL TEST
3463	020146	025772	MT0999	: 0 SEC	NULL TEST
3464	020150	025772	MT0999	: 0 SEC	NULL TEST
3465	020152	025772	MT0999	: 0 SEC	NULL TEST
3466	020154	025772	MT0999	: 0 SEC	NULL TEST
3467	020156	025772	MT0999	: 0 SEC	NULL TEST
3468	020160	025772	MT0999	: 0 SEC	NULL TEST
3469	020162	025772	MT0999	: 0 SEC	NULL TEST
3470	020164	025772	MT0999	: 0 SEC	NULL TEST
3471	020166	025772	MT0999	: 0 SEC	NULL TEST
3472	020170	025772	MT0999	: 0 SEC	NULL TEST
3473	020172	025772	MT0999	: 0 SEC	NULL TEST
3474	020174	025772	MT0999	: 0 SEC	NULL TEST
3475	020176	025772	MT0999	: 0 SEC	NULL TEST
3476	020200	025772	MT0999	: 0 SEC	NULL TEST
3477	020202	025772	MT0999	: 0 SEC	NULL TEST
3478	020204	025772	MT0999	: 0 SEC	NULL TEST
3479	020206	025772	MT0999	: 0 SEC	NULL TEST
3480	020210	025772	MT0999	: 0 SEC	NULL TEST
3481	020212	025772	MT0999	: 0 SEC	NULL TEST
3482	020214	025772	MT0999	: 0 SEC	NULL TEST
3483	020216	025772	MT0999	: 0 SEC	NULL TEST
3484	020220	025772	MT0999	: 0 SEC	NULL TEST
3485	020222	025772	MT0999	: 0 SEC	NULL TEST

END ;OF CASE RO

L234:::~::~
ADD (SP)+,@SP
MOV @ (SP)+,-(SP)
JSR PC,@(SP)+

3486 020224 062616
020224 013646
020230 004736
3487 020232 000207

RETURN

3490 020234

SUBTST <<DETERMINE ADDRESSING ACCORDING TO INTERLEAVE>>
 :*****
 :*SUBTEST DETERMINE ADDRESSING ACCORDING TO INTERLEAVE
 :*****

3491
 3492
 3493
 3494
 3495
 3496
 3497
 3498
 3499 020234 012737 060000 002336
 3500 020242 012737 177777 002432
 3501 020250 012737 000004 002414
 3502 020256 000207
 3503
 3504
 3505 020260 013737 002260 002410
 3506 020266 123727 002422 000060
 3507 020276 012737 060000 002336
 3508 020304 000403
 3509 020306 012737 060004 002336
 3510 020314
 3511 020314 012737 000010 002414
 3512 020322 000207
 3513
 3514
 3515 020324 012737 060000 002336
 3516 020332 012737 177777 002432
 3517 020340 012737 000010 002414
 3518 020346 000207
 3519
 3520
 3521 020350 013737 002260 002410
 3522 020356 123727 002422 000060
 3523 020366 012737 060000 002336
 3524 020374

```

:CSRFBANK := THE FIRST BANK OF THIS CONTROLLER OF THIS CSR
:CSRLBANK := THE LAST BANK OF THIS CONTROLLER OF THIS CSR
:CSRFIRST := THE FIRST ADDRESS OF THIS CONTROLLER OF THIS CSR
:CSRINC := THE NUMBER TO ADD TO THE ADDRESS PAIR
:          TO RETURN TO THIS CONTROLLER OF THIS CSR
:SPECIAL := A FLAG SET IF THIS BOX IS NOT INTERNALLY INTERLEAVED

:NO INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE
INT..0: LET CSRFIRST := #FIRST
          SET SPECIAL
          LET CSRINC := #4
          RETURN
          MOV #FIRST,CSRFIRST
          MOV #-1,SPECIAL
          MOV #4,CSRINC

:INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE
INT..1: LET CSRFBANK := FIRSTB
          IFB SIDE EQ #'0
          LET CSRFIRST := #FIRST
          ELSE
          LET CSRFIRST := #FIRST+4
          END ;OF IFB
          MOV FIRSTB,CSRFBANK
          CMPB SIDE,#'0
          BNE L235
          MOV #FIRST,CSRFIRST
          BR L236
          L235:.....
          MOV #FIRST+4,CSRFIRST
          L236:.....
          MOV #10,CSRINC

:NO INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH EVEN DOUBLE WORDS SELECTED?)
INT..2: LET CSRFIRST := #FIRST
          SET SPECIAL
          LET CSRINC := #10
          RETURN
          MOV #FIRST,CSRFIRST
          MOV #-1,SPECIAL
          MOV #10,CSRINC

:INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH EVEN DOUBLE WORDS SELECTED?)
INT..3: LET CSRFBANK := FIRSTB
          IFB SIDE EQ #'0
          LET CSRFIRST := #FIRST
          ELSE
          MOV FIRSTB,CSRFBANK
          CMPB SIDE,#'0
          BNE L237
          MOV #FIRST,CSRFIRST
  
```



```

3557 020556          LET CSRFIRST := #FIRST+30          L243:.....
3558 020556 012737 060030 002336          END ;OF IFB          MOV #FIRST+30,CSRFIRST
3559 020564          LET CSRINC := #40          L244:.....
3560 020564 012737 000040 002414          RETURN          MOV #40,CSRINC
3561 020572 000207
3562          ;IMPOSSIBLE CONDITION
3563          ;NO INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3564 020574 INT.10: ;FALL THRU TO INT.11
3565          ;IMPOSSIBLE CONDITION
3566          ;INTERNAL INTERLEAVE, NO EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3567          INT.11: IF FSINFLAG IS FALSE
3568 020574 005737 002542          TST FSINFLAG
3569 020602 001005          BNE L245
3570 020602 012737 177777 002542          SET FSINFLAG          MOV #-1,FSINFLAG
3571 020610 104401 116137          TYPE MSG097
3572 020614          TYPEIT ,MSG097
3573 020614          .DSABL CRF
3574 020614          END ;OF IF FSINFLAG
3575 020614 000621          GOTO INT..1          L245:.....
3576 020616          ;NO INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3577 020616 012737 060004 002336 INT.12: LET CSRFIRST := #FIRST+4          MOV #FIRST+4,CSRFIRST
3578 020624          SET SPECIAL          MOV #-1,SPECIAL
3579 020624 012737 177777 002432          LET CSRINC := #10          MOV #10,CSRINC
3580 020632 012737 000010 002414          RETURN
3581 020640 000207          ;INTERNAL INTERLEAVE, 2-1 EXTERNAL INTERLEAVE (WITH ODD DOUBLE WORDS SELECTED?)
3582 020642          INT.13: LET CSRFBANK := FIRSTB          MOV FIRSTB,CSRFBANK
3583 020642 013737 002260 002410          IFB SIDE EQ #'0
3584 020650          LET CSRFIRST := #FIRST+4          MOV #FIRST+4,CSRFIRST
3585 020650 123727 002422 000060          ELSE
3586 020656 001004          LET CSRFIRST := #FIRST+14          MOV #FIRST+14,CSRFIRST
3587 020660 012737 060004 002336          BR L247          L246:.....
3588 020666 000403          LET CSRFIRST := #FIRST+14          MOV #FIRST+14,CSRFIRST
3589 020670          END ;OF IFB          L247:.....
3590 020670 012737 060014 002336          LET CSRINC := #20          MOV #20,CSRINC
3591 020676          RETURN
3592 020676 012737 000020 002414          ;NO INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=0,A02=1)
3593 020704 000207

```

3591	020706				INT.14: LET CSRFIRST := #FIRST+4	
	020706	012737	060004	002336		MOV #FIRST+4,CSRFIRST
3592	020714				SET SPECIAL	
	020714	012737	177777	002432		MOV #-1,SPECIAL
3593	020722				LET CSRINC := #20	
	020722	012737	000020	002414		MOV #20,CSRINC
3594	020730	000207			RETURN	
3595						
3596						
3597	020732				INT.15: ;INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=0,A02=1)	
	020732	013737	002260	002410	LET CSRFBANK := FIRSTB	MOV FIRSTB,CSRFBANK
3598	020740				IFB SIDE EQ #'0	
	020740	123727	002422	000060		CMPB SIDE,#'0
	020746	001004				BNE L250
3599	020750				LET CSRFIRST := #FIRST+4	
	020750	012737	060004	002336		MOV #FIRST+4,CSRFIRST
3600	020756				ELSE	
	020756	000403				BR L251
	020760					L250:::~::~
3601	020760				LET CSRFIRST := #FIRST+24	
	020760	012737	060024	002336		MOV #FIRST+24,CSRFIRST
3602	020766				END ;OF IFB	
	020766					L251:::~::~
3603	020766				LET CSRINC := #40	
	020766	012737	000040	002414		MOV #40,CSRINC
3604	020774	000207			RETURN	
3605						
3606						
3607	020776				INT.16: ;NO INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=1,A02=1)	
	020776	012737	060014	002336	LET CSRFIRST := #FIRST+14	MOV #FIRST+14,CSRFIRST
3608	021004				SET SPECIAL	
	021004	012737	177777	002432		MOV #-1,SPECIAL
3609	021012				LET CSRINC := #20	
	021012	012737	000020	002414		MOV #20,CSRINC
3610	021020	000207			RETURN	
3611						
3612						
3613	021022				INT.17: ;INTERNAL INTERLEAVE, 4-1 EXTERNAL INTERLEAVE (WITH A03=1,A02=1)	
	021022	013737	002260	002410	LET CSRFBANK := FIRSTB	MOV FIRSTB,CSRFBANK
3614	021030				IFB SIDE EQ #'0	
	021030	123727	002422	000060		CMPB SIDE,#'0
	021036	001004				BNE L252
3615	021040				LET CSRFIRST := #FIRST+14	
	021040	012737	060014	002336		MOV #FIRST+14,CSRFIRST
3616	021046				ELSE	
	021046	000403				BR L253
	021050					L252:::~::~
3617	021050				LET CSRFIRST := #FIRST+34	
	021050	012737	060034	002336		MOV #FIRST+34,CSRFIRST
3618	021056				END ;OF IFB	
	021056					L253:::~::~
3619	021056				LET CSRINC := #40	
	021056	012737	000040	002414		MOV #40,CSRINC
3620	021064	000207			RETURN	
3621						

```

3624 021066      SIDEOK: SUBTST <<DETERMINE IF BANK BELONGS TO CORRECT SIDE>>
:*****
:*SUBTEST      DETERMINE IF BANK BELONGS TO CORRECT SIDE
:*****
                ;RETURN ERROR IF NOT!

3625
3626
3627 021066 005037 002404      CLR      CHECK
3628 021072 013702 002336      MOV      CSRFIRST,R2
3629 021076 104475                CB1CSR          ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
3630 021100                SUPERVISOR      ;ENTER SUPERVISOR MODE
                BIS      #BIT14,PSW          ;DO IT TO IT!
                .DSABL CRF
3631 021106 104424                CACHOFF        ;TURN CACHE OFF
3632 021110 012722 000000      MOV      #0,(R2)+
3633 021114 012712 000000      MOV      #0,(R2)
3634 021120 104503                CLR1CSR        ;CLEAR 1 SELECTED MK11 CSR
3635 021122 005712                TST      (R2)
3636 021124 104426                READCSR
3637 021126 013701 002146      MOV      CSR+2,R1
3638
3639                ;ELIMINATE DBE
3640 021132 104471                ECC1DIS        ;DISABLE ECC ON 1 SELECTED CSR
3641 021134 013712 002430      MOV      ZEROS,(R2)
3642 021140 013742 002430      MOV      ZEROS,-(R2)
3643 021144 104503                CLR1CSR        ;CLEAR 1 SELECTED MK11 CSR
3644 021146 104417                KERNEL        ;ENTER KERNEL MODE
3645 021150 104423                CACHON        ;TURN CACHE ON
3646
3647 021152                IF #BIT9 OFF.IN R1 AND SIDE EQ #'0 THEN $RETURN NOERROR
                021152 032701 001000
                021156 001006
                021160 023727 002422 000060
                021166 001002
                021170 000241
                021172 000207
                021174
                L254:.....
3648 021174                IF #BIT9 SET.IN R1 AND SIDE EQ #'1 THEN $RETURN NOERROR
                021174 032701 001000
                021200 001406
                021202 023727 002422 000061
                021210 001002
                021212 000241
                021214 000207
                021216
                L255:.....
3649 021216                $RETURN ERROR
                021216 000261
                021220 000207
                SEC
                RTS PC
    
```

```

3653 021222 MKTEST: SUBTST <<SUBR MK11 TEST DISPATCH>>
:*****
:*SUBTEST SUBR MK11 TEST DISPATCH
:*****
IF MULTIONLY IS TRUE THEN $RETURN

3654 021222 005737 002004 TST MULTIONLY
021222 001401 BEQ L256
021230 000207 RTS PC
021232 L256:;

3655 021232 IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
021232 032777 000001 161526 BIT #SWO,@SWR
021240 001003 BNE L257
021242 005737 002442 TST ACTFLAG
021246 001402 BEQ L260
021250 L257:;

3656 021250 104470 ECCDIS ;DISABLE ERROR CORRECTION
3657 021252 ELSE
021252 000401 BR L261
021254 L260:;

3658 021254 104502 CLRCR ;CLEAR MK11 CSRS
3659 021256 END ;OF IF
021256 L261:;

3660 021256 012737 000002 002100 MOV #2,NOPAR ;INDICATE PARITY ACTION
3661 021264 012737 000002 002406 MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
3662 021272 013700 002120 MOV PATTERN,RO ;GET PATTERN NUMBER
3663 021276 006300 ASL RO ;MAKE IT A WORD ADDRESS
3664 021300 IF MKPAT(RO) NE #MT0035 AND MKPAT(RO) NE #MT0999
021300 026027 021402 025652 CMP MKPAT(RO),#MT0035
021306 001405 BEQ L262
021310 026027 021402 025772 CMP MKPAT(RO),#MT0999
021316 001401 BEQ L262

3665 021320 104511 INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON "BANK"
3666 021322 END ;OF IF MKPAT(RO)
021322 L262:;

3667 021322 004770 021402 CALL @MKPAT(RO) ;INDEX OFF TABLE
3668 021326 IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
021326 032777 000001 161432 BIT #SWO,@SWR
021334 001003 BNE L263
021336 005737 002442 TST ACTFLAG
021342 001413 BEQ L264
021344 L263:;

3669 021344 104476 WASSBE ;WAS THERE ANY SINGLE BIT ERRORS
3670 021346 ON.ERROR
021346 103011 BCC L265

3671 021350 SET SBFLAG MOV #-1,SBFLAG
021350 012737 177777 002022
3672 021356 013737 002144 002150 MOV CSR,SBECR
3673 021364 013737 002146 002152 MOV CSR+2,SBECR+2
3674 021372 END ;OF ON.ERROR
021372 L265:;

3675 021372 END ;OF IF #SWO
021372 L264:;

3676 021372 104472 ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
3677 021374 005037 002100 CLR NOPAR ;INDICATE PARITY ACTION
3678 021400 000207 RETURN
3679
3680 ;WARNING IF YOU CHANGE THIS TABLE ALSO
    
```

```
3681                                     :CHANGE '$DDW0' - '$DDW5' (THE PATTERN BIT MAP)
3682
3683
3684 021402                                MKPAT: :PAT      TIME      DISCRPTION
3685 021402 025652                        :NOTE MT0035 MUST BE FIRST & LAST
3686 021404 023134                        MT0035  :<1 SEC  :SOFT ERROR - BACKGROUND PATTERN TEST
3687 021406 022414                        MT0017  :<1 SEC  :HOLDING 1'S & 0'S TEST
3688 021410 021740                        MT0007  :<1 SEC  :ADDRESS BIT TEST
3689 021412 022000                        MT0001  :<1 SEC  :ADDRESS TEST
3690 021414 022216                        MT0002  :<1 SEC  :COMPLEMENT ADDRESS TEST
3691 021416 022300                        MT0004  : 1 SEC  :ROTATING ZEROS TEST
3692 021420 023612                        MT0005  : 1 SEC  :ROTATING ONES TEST
3693 021422 023156                        MT0021  : 1 SEC  :MARCHING 0'S & 1'S TEST
3694 021424 023734                        MT0020  :<1 SEC  :MARCHING 1'S & 0'S IN CHECK BITS
3695 021426 024240                        MT0022  :10 SEC  :REFRESH & SHIFTING DIAGONAL TEST
3696 021430 024032                        MT0026  :<1 SEC  :RANDOM DATA TEST
3697 021432 025214                        MT0024  :20 SEC  :FAST GALLOPING PATTERN TEST
3698 021434 025340                        MT0031  : 3 SEC  :SOB-A-LONG TEST
3699 021436 025530                        MT0033  :<1 SEC  :WRITE RECOVERY TEST
3700 021440 025652                        MT0034  :35 SEC  :BRANCH GOBBLE TEST
3701                                     MT0035  :<1 SEC  :SOFT ERROR - BACKGROUND PATTERN TEST
3702 021442 025772                        :NOTE MT0035 MUST BE FIRST & LAST
3703 021444 025772                        MT0999  : 0 SEC  :NULL TEST
3704 021446 025772                        MT0999  : 0 SEC  :NULL TEST
3705 021450 025772                        MT0999  : 0 SEC  :NULL TEST
3706 021452 025772                        MT0999  : 0 SEC  :NULL TEST
3707 021454 025772                        MT0999  : 0 SEC  :NULL TEST
3708 021456 025772                        MT0999  : 0 SEC  :NULL TEST
3709 021460 025772                        MT0999  : 0 SEC  :NULL TEST
3710 021462 025772                        MT0999  : 0 SEC  :NULL TEST
3711 021464 025772                        MT0999  : 0 SEC  :NULL TEST
3712 021466 025772                        MT0999  : 0 SEC  :NULL TEST
3713 021470 025772                        MT0999  : 0 SEC  :NULL TEST
3714 021472 025772                        MT0999  : 0 SEC  :NULL TEST
3715 021474 025772                        MT0999  : 0 SEC  :NULL TEST
3716 021476 025772                        MT0999  : 0 SEC  :NULL TEST
3717 021500 025772                        MT0999  : 0 SEC  :NULL TEST
3718 021502 025772                        MT0999  : 0 SEC  :NULL TEST
3719 021504 025772                        MT0999  : 0 SEC  :NULL TEST
3720 021506 025772                        MT0999  : 0 SEC  :NULL TEST
3721 021510 025772                        MT0999  : 0 SEC  :NULL TEST
```


3769	021664	025772	MT0999	:	0	SEC	:	NULL	TEST
3770	021666	025772	MT0999	:	0	SEC	:	NULL	TEST
3771	021670	025772	MT0999	:	0	SEC	:	NULL	TEST
3772	021672	025772	MT0999	:	0	SEC	:	NULL	TEST
3773	021674	025772	MT0999	:	0	SEC	:	NULL	TEST
3774	021676	025772	MT0999	:	0	SEC	:	NULL	TEST
3775	021700	025772	MT0999	:	0	SEC	:	NULL	TEST
3776	021702	025772	MT0999	:	0	SEC	:	NULL	TEST

3778
3779
3780
3781 021704

3782 021704 005037 002366
3783 021710 012700 060000
3784 021714 012701 040000
3785 021720 004737 035204
3786 021724
021724 004537 043710
021730 026376

3787 021732 004737 026042
3788 021736 000207
3789 021740

3790 021740 012737 000001 002366
3791 021746 012700 060000
3792 021752 012701 040000
3793 021756 005002
3794 021760 004737 035204
3795 021764
021764 004537 043710
021770 026422

3796 021772 004737 026042
3797 021776 000207
3798 022000

3799 022000 012737 000002 002366
3800 022006 012700 160000
3801 022012 012701 040000
3802 022016 012702 000001
3803 022022 010103
3804 022024 012704 060000
3805 022030 012705 100001
3806 022034
022034 004537 043710
022040 026454

3807 022042 004737 026042
3808 022046 000207

```
.SBTTL PATTERNS

.SBTTL MEMORY TEST SETUP ROUTINES
MT0000: SUBTST <<MT0000      SETUP DATA PATTERN TEST>>
:*****
:*SUBTEST      MT0000      SETUP DATA PATTERN TEST
:*****
      CLR      REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #FIRST,R0
      MOV      #SIZE,R1
      CALL     REGCOPY
      BMOV     MTP000
      JSR     R5,BLOCK1
      MTP000
      .DSABL   CRF
      CALL     SUPD01      ;DO IT IN SUPERVISOR MODE
      RETURN
MT0001: SUBTST <<MT0001      SETUP ADDRESS TEST>>
:*****
:*SUBTEST      MT0001      SETUP ADDRESS TEST
:*****
      MOV      #1,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #FIRST,R0
      MOV      #SIZE,R1
      CLR      R2
      CALL     REGCOPY
      BMOV     MTP001
      JSR     R5,BLOCK1
      MTP001
      .DSABL   CRF
      CALL     SUPD01      ;DO IT IN SUPERVISOR MODE
      RETURN
MT0002: SUBTST <<MT0002      SETUP COMPLEMENT ADDRESS TEST>>
:*****
:*SUBTEST      MT0002      SETUP COMPLEMENT ADDRESS TEST
:*****
      MOV      #2,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #LAST+2,R0
      MOV      #SIZE,R1
      MOV      #1,R2
      MOV      R1,R3
      MOV      #FIRST,R4
      MOV      #100001,R5
      BMOV     MTP002
      JSR     R5,BLOCK1
      MTP002
      .DSABL   CRF
      CALL     SUPD01
      RETURN
```

3811 022050

MT0003: SUBTST <<MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST>>

:SUBTEST MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST

3812 022050 012737 000003 002366
3813 022056 005037 002406
3814 022062 004737 035214
3815 022066 012701 060000
3816 022072 005000
3817 022074 160001
3818 022076 012703 020000
3819 022102 072327 177770
3820 022106 012702 000004
3821 022112 012705 000100
3822 022116
022116 004537 043744
022122 000021
022124 177636
022126 026506

MOV #3,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR PCBUMP ;TRAPS DO NOT ADD TO PC
1\$: CALL FLIPWARN ;SETUP WARNING CONSTANTS & R2
2\$: MOV #FIRST,R1 ;R1 <-- STARTING ADDRESS
CLR R0 ;INTERLEAVE ADDRESS CONSTANT
SUB R0,R1 ;CORRECT FOR ADD BEFORE USE MODE
MOV #SIZE/2,R3 ;INTERLEAVE SIZE (8K,4K,2K,OR 1K (DOUBLE WORDS))
ASH #-8,R3 ;R3 <-- R3 / 256.
MOV #4,R2 ;SMALL LOOP SIZE
MOV #64,R5 ;MEDIUM LOOP SIZE
BMOV MTPA03,UDPDR7,17.
JSR R5,BLOCK3
17.
UDPDR7
MTPA03
.DSABL CRF

3823 022130 104415
3824 022132 012737 177636 002360
3825 022140 004737 026220
3826 022144
022144 004537 043710
022150 026550

SAVREG
MOV #FASTCITY-2,SUPDOADD
CALL SUPDO3
BMOV MTPB03
JSR R5,BLOCK1
MTPB03
.DSABL CRF

3827 022152
022152 004537 043744
022156 000010
022160 172360
022162 026610

BMOV MTPC03,KDPAR0,8.
JSR R5,BLOCK3
8.
KDPAR0
MTPC03
.DSABL CRF

3828 022164
022164 004537 043744
022170 000010
022172 172260
022174 026630

BMOV MTPD03,SDPAR0,8.
JSR R5,BLOCK3
8.
SDPAR0
MTPD03
.DSABL CRF

3829 022176 104416
3830 022200 004737 026056
3831 022204 022737 000003 002734
3832
3833 022212 001323
3834 022214 000207
3835

RESREG
CALL SUPDO2
CMP #3,FLIPLOC ;DONE WITH 4 PATTERNS
;[(0,177777):(177777,0):(401,177777):(177777,401)]?
BNE 1\$;NO - LOOP
RETURN

3836 022216

MT0004: SUBTST <<MT0004 SETUP ROTATING ZEROS TEST>>

:SUBTEST MT0004 SETUP ROTATING ZEROS TEST

3837 022216 012737 000004 002366
3838 022224 012737 000004 002406
3839 022232 013702 002726
3840 022236 004737 035344
3841 022242 012700 060000
3842 022246 012701 040000
3843 022252

MOV #4,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV ONES,R2
CALL BACKGND ;WRITE BACKGROUND OF ONES
MOV #FIRST,R0
MOV #SIZE,R1
BMOV MTPA04

022252 004537 043710
 022256 026650
 3844 022260
 022260 004537 043744
 022264 000010
 022266 172360
 ,022270 026704
 3845 022272 004737 026056
 3846 022276 000207
 3847 022300

```

JSR R5,BLOCK1
MTPA04
.DSABL CRF
BMOV MTPB04,KDPAR0,8.
JSR R5,BLOCK3
8.
KDPAR0
MTPB04
.DSABL CRF
CALL SUPD02
RETURN
MT0005: SUBTST <<MT0005 SETUP ROTATING ONES TEST>>

```

3848 022300 012737 000005 002366
 3849 022306 012737 000004 002406
 3850 022314 005002
 3851 022316 004737 035344
 3852 022322 012700 060000
 3853 022326 012701 040000
 3854 022332
 022332 004537 043710
 022336 026724
 3855 022340
 022340 004537 043744
 022344 000010
 022346 172360
 022350 026704
 3856 022352 004737 026056
 3857 022356 000207

```

*****
: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
BMOV MTP005
JSR R5,BLOCK1
MTP005
.DSABL CRF
BMOV MTPB04,KDPAR0,8.
JSR R5,BLOCK3
8.
KDPAR0
MTPB04
.DSABL CRF
CALL SUPD02
RETURN

```

3860 022360
3861 022360 012737 000006 002366
3862 022366 012737 000004 002406
3863 022374 013701 002506
3864 022400 012737 026760 002360
3865 022406 004737 026220
3866 022412 000207
3867 022414

```
MT0006: SUBTST <<MT0006      SETUP INITIAL DATA TEST>>
:*****
:*SUBTEST      MT0006  SETUP INITIAL DATA TEST
:*****
      MOV      #6,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #4,PCBUMP      ;TRAPS ADD 4 TO PC
      MOV      TESTADD,R1
      MOV      #MTP006,SUPDOADD
      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
      RETURN
```

3868 022414 012737 000007 002366
3869 022422 005002
3870 022424 004737 035344
3871 022430 012701 060000
3872 022434 012702 C00001
3873 022440 050201
3874 022442 012737 027150 002360
3875 022450 004737 026220
3876 022454 000207
3877 022456

```
MT0007: SUBTST <<MT0007      SETUP ADDRESS BIT TEST>>
:*****
:*SUBTEST      MT0007  SETUP ADDRESS BIT TEST
:*****
      MOV      #7,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      CLR      R2
      CALL     BACKGND        ;OF ZEROS
      MOV      #FIRST,R1
      MOV      #1,R2
      BIS      R2,R1
      MOV      #MTP007,SUPDOADD
      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
      RETURN
```

3878 022456 012737 000010 002366
3879 022464 012737 000004 002406
3880 022472 013704 002506
3881 022476 012737 027250 002360
3882 022504 004737 026220
3883 022510 000207

```
MT0010: SUBTST <<MT0010      SETUP BYTE ADDRESSING TEST>>
:*****
:*SUBTEST      MT0010  SETUP BYTE ADDRESSING TEST
:*****
      MOV      #10,REALPAT     ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #4,PCBUMP      ;TRAPS ADD 4 TO PC
      MOV      TESTADD,R4
      MOV      #MTP010,SUPDOADD
      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
      RETURN
```

```
3886 022512 MT0011: SUBTST <<MT0011      SETUP CREATE SINGLE BIT ERROR TEST>>
:*****
:*SUBTEST      MT0011  SETUP CREATE SINGLE BIT ERROR TEST
:*****
3887 022512      IF NOECCFLAG IS TRUE THEN $RETURN
022512 005737 002474                                TST NOECCFLAG
022516 001401                                BEQ L270
022520 000207                                RTS PC
022522                                L270:*****
3888 022522      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
022522 005737 002442                                TST ACTFLAG
022526 001003                                BNE L271
022530 005737 002444                                TST APTFLAG
022534 001404                                BEQ L272
022536                                L271:*****
3889 022536      IF $PASS NE #0 THEN $RETURN
022536 005737 075142                                TST $PASS
022542 001401                                BEQ L273
022544 000207                                RTS PC
022546                                L273:*****
3890 022546      END ;OF IF ACTFLAG
022546                                L272:*****
3891 022546 012737 000011 002366  MOV #11,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3892 022554 012737 027356 002360  MOV #MTP011,SUPDOADD
3893 022562 004737 026220          CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
3894 022566 000207          RETURN
3895 022570 MT0012: SUBTST <<MT0012      SETUP WRITE BYTE CLEARS SBE TEST>>
:*****
:*SUBTEST      MT0012  SETUP WRITE BYTE CLEARS SBE TEST
:*****
3896 022570      IF NOECCFLAG IS TRUE THEN $RETURN
022570 005737 002474                                TST NOECCFLAG
022574 001401                                BEQ L274
022576 000207                                RTS PC
022600                                L274:*****
3897 022600      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
022600 005737 002442                                TST ACTFLAG
022604 001003                                BNE L275
022606 005737 002444                                TST APTFLAG
022612 001404                                BEQ L276
022614                                L275:*****
3898 022614      IF $PASS NE #0 THEN $RETURN
022614 005737 075142                                TST $PASS
022620 001401                                BEQ L277
022622 000207                                RTS PC
022624                                L277:*****
3899 022624      END ;OF IF ACTFLAG
022624                                L276:*****
3900 022624 012737 000012 002366  MOV #12,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3901 022632 012737 027766 002360  MOV #MTP012,SUPDOADD
3902 022640 004737 026220          CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
3903 022644 000207          RETURN
3904 022646 MT0013: SUBTST <<MT0013      SETUP CREATE DOUBLE BIT ERROR TEST>>
:*****
:*SUBTEST      MT0013  SETUP CREATE DOUBLE BIT ERROR TEST
:*****
3905 022646      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
```

```

022646 005737 002442
022652 001003
022654 005737 002444
022660 001404
022662
3906 022662 IF $PASS NE #0 THEN $RETURN
022662 005737 075142
022666 001401
022670 000207
022672
3907 022672 END ;OF IF ACTFLAG
022672
3908 022672 012737 000013 002366 MOV #13,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3909 022700 012737 030344 002360 MOV #MTP013,SUPDOADD
3910 022706 012737 000003 002100 MOV #3,NOPAR ;INDICATE PARITY ACTION
3911 022714 004737 026220 CALL SUPD03 ;DO IT IN SUPERVISOR MODE
3912 022720 000207 RETURN
3913 022722 MTO014: SUBTST <<MTO014 SETUP WRITE INHIBIT DURING DATIP WITH DBE>>
:*****
:*SUBTEST MTO014 SETUP WRITE INHIBIT DURING DATIP WITH DBE
:*****
3914 022722 IF NOECCFLAG IS TRUE THEN $RETURN
022722 005737 002474 TST NOECCFLAG
022726 001401 BEQ L303
022730 000207 RTS PC
022732
3915 022732 IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
022732 005737 002442 TST ACTFLAG
022736 001003 BNE L304
022740 005737 002444 TST APTFLAG
022744 001404 BEQ L305
022746
3916 022746 IF $PASS NE #0 THEN $RETURN
022746 005737 075142 TST $PASS
022752 001401 BEQ L306
022754 000207 RTS PC
022756
3917 022756 END ;OF IF ACTFLAG
022756
3918 022756 012737 000014 002366 MOV #14,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3919 022764 012737 030716 002360 MOV #MTP014,SUPDOADD
3920 022772 004737 026220 CALL SUPD03 ;DO IT IN SUPERVISOR MODE
3921 022776 000207 RETURN
    
```

```

TST ACTFLAG
BNE L300
TST APTFLAG
BEQ L301
L300:*****
    
```

```

TST $PASS
BEQ L302
RTS PC
L302:*****
    
```

```

L301:*****
    
```

```

;INDICATE PARITY ACTION
;DO IT IN SUPERVISOR MODE
    
```

```

*****
*****
    
```

```

TST NOECCFLAG
BEQ L303
RTS PC
L303:*****
    
```

```

TST ACTFLAG
BNE L304
TST APTFLAG
BEQ L305
L304:*****
    
```

```

TST $PASS
BEQ L306
RTS PC
L306:*****
    
```

```

L305:*****
    
```

3924 023000
3925 023000 005737 002474
023000 001401
023004 000207
023010
3926 023010 005737 002442
023010 001003
023014 005737 002444
023016 001404
023022
3927 023024 005737 075142
023024 001401
023030 000207
023032
023034
3928 023034
023034
3929 023034 012737 000015 002366
3930 023042 012737 031312 002360
3931 023050 004737 026220
3932 023054 000207
3933 023056
3934 023056 005737 002474
023056 001401
023062 000207
023064
023066
3935 023066 005737 002442
023066 001003
023072 005737 002444
023074 001404
023100
023102
3936 023102 005737 075142
023102 001401
023106 000207
023110
023112
3937 023112
023112
3938 023112 012737 000016 002366
3939 023120 012737 031700 002360
3940 023126 004737 026220
3941 023132 000207
3942 023134
3943 023134 012737 000017 002366

```
MT0015: SUBST <<MT0015      SETUP WRITE INHIBIT OF BYTE WITH DBE>>
:*****
:*SUBTEST      MT0015  SETUP WRITE INHIBIT OF BYTE WITH DBE
:*****
      IF NOECCFLAG IS TRUE THEN $RETURN
                                TST NOECCFLAG
                                BEQ L307
                                RTS PC
                                L307:*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
                                TST ACTFLAG
                                BNE L310
                                TST APTFLAG
                                BEQ L311
                                L310:*****
      IF $PASS NE #0 THEN $RETURN
                                TST $PASS
                                BEQ L312
                                RTS PC
                                L312:*****
      END ;OF IF ACTFLAG
                                L311:*****
      MOV      #15,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #MTP015,SUPDOADD
      CALL     SUPD03           ;DO IT IN SUPERVISOR MODE
      RETURN
MT0016: SUBST <<MT0016      SETUP WRITE INHIBIT OF WORD WITH DBE>>
:*****
:*SUBTEST      MT0016  SETUP WRITE INHIBIT OF WORD WITH DBE
:*****
      IF NOECCFLAG IS TRUE THEN $RETURN
                                TST NOECCFLAG
                                BEQ L313
                                RTS PC
                                L313:*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
                                TST ACTFLAG
                                BNE L314
                                TST APTFLAG
                                BEQ L315
                                L314:*****
      IF $PASS NE #0 THEN $RETURN
                                TST $PASS
                                BEQ L316
                                RTS PC
                                L316:*****
      END ;OF IF ACTFLAG
                                L315:*****
      MOV      #16,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #MTP016,SUPDOADD
      CALL     SUPD03           ;DO IT IN SUPERVISOR MODE
      RETURN
MT0017: SUBST <<MT0017      SETUP HOLDING 1'S & 0'S>>
:*****
:*SUBTEST      MT0017  SETUP HOLDING 1'S & 0'S
:*****
      MOV      #17,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
```

3944 023142 012737 032302 002360
3945 023150 004737 026220
3946 023154 000207

MOV #MTP017,SUPDOADD
CALL SUPDO3
RETURN

;DO IT IN SUPERVISOR MODE

```

3949 023156
3950 023156 005737 002442
      023156 001003
      023162 005737 002444
      023164 001404
      023170
      023172
3951 023172 005737 075142
      023172 001401
      023176 000207
      023200
      023202
3952 023202
      023202
3953 023202 012737 000020 002366
3954 023210 005037 002100
3955 023214 012737 032572 002360
3956 023222 013700 002334
3957 023226 005037 023320
      023226
      023232
3958 023232 013737 023320 002256
3959 023240 106300
3960 023242 103016
      023242
3961 023244 112737 000060 002422
3962 023252 004737 017610
3963 023256 005737 002432
      023256 001002
      023262 006237 002414
3964 023264
3965 023270
      023270
3966
3967 023270 010046
      023270 004737 023322
3968 023272
3969 023276 012600
      023276
3970 023300
      023300
3971 023300 062737 000004 023320
      023306 023727 023320 000034
      023314 003746
      023316
3972 023316 000207
3973 023320 000000
3974
3975 023322 013700 002336
3976 023326 010001
3977 023330 013702 002414
3978 023334 005003
3979 023336 012704 100000
    
```

```

MT0020: SUBTST <<MT0020          SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST>>
:*****
:*SUBTEST          MT0020  SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST
:*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE

                                TST ACTFLAG
                                BNE L317
                                TST APTFLAG
                                BEQ L320
                                L317:*****

      IF $PASS NE #0 THEN $RETURN

                                TST $PASS
                                BEQ L321
                                RTS PC
                                L321:*****

      END ;OF IF ACTFLAG

                                L320:*****
      MOV #20,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      CLR NOPAR                ;INDICATE PARITY ACTION
      MOV #MTPA20,SUPDOADD
      MOV MKCSRS,R0
      FOR MTV020 := #0 TO #34 BY #4

                                CLR MTV020
                                B35:*****

      MOV MTV020,CSRNO
      ASLB R0
      ON.ERROR

                                BCC L322

      MOVB #0,SIDE
      CALL INTKRAP
      IF SPECIAL IS FALSE

                                TST SPECIAL
                                BNE L323

      ASR CSRINC
      END ;OF IF SPECIAL

                                L323:*****
      ;NOW CSRFIRST & CSRINC ARE SETUP
      PUSH R0
                                MOV R0,-(SP)

      CALL MT020Z
      POP R0
                                MOV (SP)+,R0

      END ;OF ON.ERROR

                                L322:*****
      END ;OF FOR CSRNO

                                ADD #4,MTV020
                                CMP MTV020,#34
                                BLE B35
                                E35:*****

      RETURN
      MTV020: 0          ;VARIABLE FOR PAT 20

      MT020Z: MOV CSRFIRST,R0
              MOV R0,R1
              MOV CSRINC,R2
              CLR R3
              MOV #BIT15,R4
    
```

```

3980 023342 012705 160000      MOV      #LAST+2,R5
3981 023346 104503              CLR1CSR      ;CLEAR 1 SELECTED MK11 CSR
3982 023350              BMOV      MTP20A
      023350 004537 043710      JSR R5,BLOCK1
      023354 032360      MTP20A
      .DSABL      CRF
3983 023356 004737 026042      CALL     SUPD01
3984 023362              IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
      023362 032777 004000 157376      BIT #SW11,@SWR
      023370 001003      BNE L324
      023372 005737 002440      TST QVFLAG
      023376 001401      BEQ L325
      023400              L324:
3985 023400              GOTO MT020Y
      023400 000434      BR MT020Y
3986 023402              END ;OF IF #SW11
      023402              L325:
3987 023402              BMOV      MTP20B
      023402 004537 043710      JSR R5,BLOCK1
      023406 032376      MTP20B
      .DSABL      CRF
3988 023410 004737 026056      CALL     SUPD02
3989 023414              IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
      023414 032777 000001 157344      BIT #SW0,@SWR
      023422 001003      BNE L326
      023424 005737 002442      TST ACTFLAG
      023430 001402      BEQ L327
      023432              L326:
3990 023432 104477      WAS1SBE      ;WAS THERE ANY SINGLE BIT ERRORS ON 1 SELECTED CSR
3991 023434              ON.ERROR THEN GOTO MT020X
      023434 103463      BCS MT020X
3992 023436              END ;OF IF #SW0
      023436              L327:
3993 023436              BMOV      MTP20C
      023436 004537 043710      JSR R5,BLOCK1
      023442 032430      MTP20C
      .DSABL      CRF
3994 023444 004737 026056      CALL     SUPD02
3995 023450              IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
      023450 032777 000001 157310      BIT #SW0,@SWR
      023456 001003      BNE L330
      023460 005737 002442      TST ACTFLAG
      023464 001402      BEQ L331
      023466              L330:
3996 023466 104477      WAS1SBE      ;WAS THERE ANY SINGLE BIT ERRORS ON 1 SELECTED CSR
3997 023470              ON.ERROR THEN GOTO MT020X
      023470 103445      BCS MT020X
3998 023472              END ;OF IF #SW0
      023472              L331:
3999 023472 013701 002336      MT020Y: MOV      CSRFIRST,R1
4000 023476              BMOV      MTP20D
      023476 004537 043710      JSR R5,BLOCK1
      023502 032462      MTP20D
      .DSABL      CRF
4001 023504 004737 026056      CALL     SUPD02
4002 023510              IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
      023510 032777 000001 157250      BIT #SW0,@SWR
    
```


4021 023612

MT0021: SUBTST <<MT0021 SETUP MARCHING 0'S & 1'S TEST>>

:SUBTEST MT0021 SETUP MARCHING 0'S & 1'S TEST

4022 023612 012737 177777 002540
023612 012737 000021 002366
4023 023620 012737 000021 002366
4024 023626 013702 002762
4025 023632 004737 035344
4026 023636 010203
4027 023640 000303
4028 023642 012701 160000
4029 023646 010105
4030 023650 012704 060000
4031 023654
023654 004537 043710
023660 033234

SET NOSCOPE
MOV #21,REALPAT ;MOV #-1,NOSCOPE
MOV BAKPAT,R2 ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CALL BACKGND
MOV R2,R3
SWAB R3
MOV #LAST+2,R1
MOV R1,R5
MOV #FIRST,R4
BMOV MTPA21
JSR R5,BLOCK1
MTPA21
.DSABL CRF
CALL SUPD01
BMOV MTPB21
JSR R5,BLOCK1
MTPB21
.DSABL CRF
CALL SUPD02
MOV R4,R1
BMOV MTPC21
JSR R5,BLOCK1
MTPC21
.DSABL CRF
CALL SUPD02
BMOV MTPD21
JSR R5,BLOCK1
MTPD21
.DSABL CRF
CALL SUPD02
CLR NOSCOPE
RETURN

4032 023662 004737 026042
4033
4034 023666
023666 004537 043710
023672 033264
4035 023674 004737 026056
4036
4037 023700 010401
4038 023702
023702 004537 043710
023706 033320
4039 023710 004737 026056
4040
4041 023714
023714 004537 043710
023720 033354
4042 023722 004737 026056
4043 023726 005037 002540
4044 023732 000207
4045
4046 023734

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

:SUBTEST MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST

4047 023734 004737 026006
4048 023740
023740 103001
023742 000207
023744
4049 023744 012737 000022 002366
4050 023752 012737 033404 002360
4051 023760 004737 026220
4052 023764 000207
4053
4054 023766

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
BCC L336
RTS PC
L336:::~::~
MOV #22,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #MTP022,SUPD0ADD
CALL SUPD03 ;DO IT IN SUPERVISOR MODE
RETURN

MT0023: SUBTST <<MT0023 SHIFTING DIAGONAL TEST>>

:*SUBTEST MT0023 SHIFTING DIAGONAL TEST

4055 023766 004737 026006
4056 023772 103001
023772 000207
023774 000207
023776
4057 023776 012737 000023 002366
4058 024004 012737 033404 002360
4059 024012 012737 177777 002006
024012 004737 026220
4060 024020 005037 002006
4061 024024 000207
4062 024030 000207

CALL KAMITEST
ON.ERROR THEN \$RETURN

MOV #23,REALPAT
MOV #MTP022,SUPDOADD
SET DIAGFLAG

CALL SUPDO3
CLR DIAGFLAG
RETURN

;CHECK FOR KAMIKAZE MODE
;IF NOT IN KAMIKAZE MODE RETURN
BCC L337
RTS PC
L337:::~::~
;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
;IDENTIFY DIAGONAL TEST TO MTP022
MOV #-1,DIAGFLAG
;DO IT IN SUPERVISOR MODE

4064 024032
 4065 024032 004737 026006
 4066 024036 103001 024040 000207
 4067 024042 012737 177777 002540
 4068 024050 012737 000024 002366
 4069 024056 013702 002762
 4070 024062 004737 035344
 4071 024066 010203
 4072 024070 010304
 4073 024072 000304
 4074 024074 012701 060000
 4075 024100 012705 157776
 4076 024104 104415
 4077 024106 004537 043710
 024112 034060
 4078 024114 004537 043744
 024120 000010
 024122 172260
 024124 034120
 4079 024126 004537 043744
 024132 000010
 024134 172360
 024136 034134
 4080 024140 012737 172260 002360
 4081 024146 004737 026234
 4082
 4083
 4084 024152 104416
 4085 024154 000302
 4086 024156 000303
 4087 024160 004737 026234
 4088 024164 005037 002540
 4089 024170 000207
 4090 024172
 4091 024172 005737 002442
 024176 001003
 024200 005737 002444
 024204 001404
 024206
 4092 024206

```

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
                                BCC L340
                                RTS PC
                                L340:;;;;;

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
SAVREG
BMOV     MTPA24
        JSR R5,BLOCK1
        MTPA24
        .DSABL      CRF
BMOV     MTPB24,SDPAR0,8.
        JSR R5,BLOCK3
        8.
        SDPAR0
        MTPB24
        .DSABL      CRF
BMOV     MTPC24,KDPAR0,8.
        JSR R5,BLOCK3
        8.
        KDPAR0
        MTPC24
        .DSABL      CRF
MOV      #SDPAR0,SUPDOADD
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

                                TST ACTFLAG
                                BNE L341
                                TST APTFLAG
                                BEQ L342
                                L341:;;;;;

IF $PASS NE #0 THEN $RETURN
    
```

024206 005737 075142
024212 001401
024214 000207
024216
4093 024216
024216
4094 024216 012737 000025 002366
4095 024224 012737 034152 002360
4096 024232 004737 026220
4097 024236 000207

END ;OF IF ACTFLAG
MOV #25,REALPAT
MOV #MTP025,SUPDOADD
CALL SUPDO3
RETURN

TST \$PASS
BEQ L343
RTS PC
L343:.....
L342:.....
;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
;DO IT IN SUPERVISOR MODE

4100 024240

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

```

:*****
:*SUBTEST      MT0026  SETUP RANDOM DATA TEST
:*****
4101 024240 012737 000026 002366      MOV    #26,REALPAT
4102 024246 005037 002406              CLR    PCBUMP                ;TRAPS DO NOT ADD TO THE PC
4103 024252 013703 002716              MOV    SEEDLO,R3            ;INITIALIZE RANDOM NUMBERS
4104 024256 013702 002714              MOV    SEEDHI,R2
4105 024262 010305                      MOV    R3,R5
4106 024264 010204                      MOV    R2,R4
4107 024266 012701 060000              MOV    #FIRST,R1
4108 024272 012700 020000              MOV    #SIZE/2,R0
4109 024276 104415                      SAVREG
4110 024300                      BMOV   MTPA26                ;WRITE ROUTINE TO FAST MEMORY
      024300 004537 043710              JSR   R5,BLOCK1
      024304 034576                      MTPA26
      .DSABL          CRF
4111 024306                      BMOV   MTPC26,KDPAR0,8.      ;RANDOM SUBPROGRAM TO FAST MEMORY
      024306 004537 043744              JSR   R5,BLOCK3
      024312 000010                      8.
      024314 172360                      KDPAR0
      024316 034646                      MTPC26
      .DSABL          CRF
4112 024320                      BMOV   MTPD26,SDPAR0,8.      ;RANDOM SUBSUBPROGRAM TO FAST MEMORY
      024320 004537 043744              JSR   R5,BLOCK3
      024324 000010                      8.
      024326 172260                      SDPAR0
      024330 034666                      MTPD26
      .DSABL          CRF
4113 024332 004737 026042              CALL  SUPD01                ;WRITE RANDOM DATA
4114 024336 005037 034626              CLR    RANODD              ;FOR ERROR REPORTING
4115 024342                      BMOV   MTPB26                ;READ ROUTINE TO FAST MEMORY
      024342 004537 043710              JSR   R5,BLOCK1
      024346 034612                      MTPB26
      .DSABL          CRF
4116 024350 104416                      RESREG
4117 024352 004737 026042              CALL  SUPD01                ;READ RANDOM DATA
4118 024356 010337 002716              MOV    R3,SEEDLO           ;UPDATE FOR NEW RANDOM NUMBERS
4119 024362 010237 002714              MOV    R2,SEEDHI
4120 024366 000207                      RETURN
    
```

4123 024370
 4124
 4125
 4126 024370 012737 000027 002366
 4127 024376 005037 002112
 024402 005037 177750
 4128 024406 104502
 4129 024410
 024410 004537 043710
 024414 026376
 4130 024416 012737 000207 177644
 4131 024424 012737 177646 002360
 4132 024432
 024432 012737 177777 002526
 4133 024440
 024440 012737 000001 002602
 024446
 4134 024446
 024446 005037 002106
 024452
 4135 024452 004737 042422
 4136 024456
 024456 005737 002124
 024462 001430
 024464 005737 002132
 024470 001025
 4137 024472 104511
 4138 024474
 024474 013702 002106
 4139 024500 012700 060000
 4140 024504 010004
 4141 024506 012701 040000
 4142 024512 010103
 4143 024514
 024514 023727 002602 000001
 024522 001002
 4144 024524 004737 026042
 4145 024530
 024530
 4146 024530
 024530 023727 002602 000002
 024536 001002
 4147 024540 004737 026220
 4148 024544
 024544
 4149 024544
 024544
 4150 024544
 024544 005237 002106
 024550 023727 002106 000167
 024556 003735
 024560

```

MTO027: SUBTST <<MTO027          UNIQUE BANK TEST>>
:*****
:*SUBTEST      MTO027  UNIQUE BANK TEST
:*****
: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
CLEAR   MARGIN,MAINT        ;CLEAR MARGINS
                                CLR  MARGIN
                                CLR  MAINT
                                ;CLEAR MK11 CSRS
CLRCRSR
BMOV    MTP000
        JSR R5,BLOCK1
        MTP000
        .DSABL      CRF
WARN7:  MOV      #207,UIPAR2    ;PUT 'RETURN' INSTRUCTION AFTER WRITE ROUTINE
        MOV      #UIPAR3,SUPDOADD
        SET NOFSMODE
                                MOV  #-1,NOFSMODE
FOR I := #1 TO #2
                                MOV  #1,I
                                B36:*****
                                CLR  BANK
                                B37:*****
                                CALL EXBANK
                                IF ACFLAG IS TRUE AND RRFLAG IS FALSE
                                TST ACFLAG
                                BEQ L344
                                TST RRFLAG
                                BNE L344
                                ;INVALIDATE BACKGROUND PATTERN ON "BANK"
                                INVALIDATE
                                LET R2 := BANK
                                MOV BANK,R2
                                MOV      #FIRST,R0
                                MOV      R0,R4
                                MOV      #SIZE,R1
                                MOV      R1,R3
                                IF I EQ #1
                                CMP  I,#1
                                BNE L345
                                CALL SUPD01
                                END ;OF IF
                                L345:*****
                                IF I EQ #2
                                CMP  I,#2
                                BNE L346
                                CALL SUPD03
                                END ;OF IF
                                L346:*****
                                END ;OF IF
                                L344:*****
                                END :OF FOR BANK
                                INC  BANK
                                CMP  BANK,#167
                                BLE  B37
                                E37:*****
    
```

4151	024560			END ;OF FOR I	
	024560	005237	002602		INC I
	024564	023727	002602	000002	CMP I,#2
	024572	003725			BLE B36
	024574				E36:.....
4152	024574			IF FS7FLAG IS TRUE	
	024574	005737	002546		TST FS7FLAG
	024600	001403			BEQ L347
4153	024602	005037	002526	CLR NOFSMODE	
4154	024606	000207		RETURN	
4155	024610			END ;OF IF FS7FLAG	
	024610				L347:.....
4156	024610			FOR I := #1 TO #2	
	024610	012737	000001	002602	MOV #1,I
	024616				B40:.....
4157	024616			FOR BANK := #167 DOWNT0 #0	
	024616	012737	000167	002106	MOV #167,BANK
	024624				B41:.....
4158	024624	004737	042422	CALL EXBANK	
4159	024630			IF ACFLAG IS TRUE AND RRFLAG IS FALSE	
	024630	005737	002124		TST ACFLAG
	024634	001430			BEQ L350
	024636	005737	002132		TST RRFLAG
	024642	001025			BNE L350
4160	024644			LET R2 := BANK	
	024644	013702	002106		MOV BANK,R2
4161	024650	005102		COM R2	
4162	024652	012700	060000	MOV #FIRST,R0	
4163	024656	010004		MOV R0,R4	
4164	024660	012701	040000	MOV #SIZE,R1	
4165	024664	010103		MOV R1,R3	
4166	024666			IF I EQ #1	
	024666	023727	002602	000001	CMP I,#1
	024674	001002			BNE L351
4167	024676	004737	026042	CALL SUPD01	
4168	024702			END ;OF IF	
	024702				L351:.....
4169	024702			IF I EQ #2	
	024702	023727	002602	000002	CMP I,#2
	024710	001002			BNE L352
4170	024712	004737	026220	CALL SUPD03	
4171	024716			END ;OF IF	
	024716				L352:.....
4172	024716			END ;OF IF	
	024716				L350:.....
4173	024716			END ;OF FOR BANK	
	024716	005337	002106		DEC BANK
	024722	023727	002106	000000	CMP BANK,#0
	024730	002335			BGE B41
	024732				E41:.....
4174	024732			END ;OF FOR I	
	024732	005237	002602		INC I
	024736	023727	002602	000002	CMP I,#2
	024744	003724			BLE B40
	024746				E40:.....
4175	024746	005037	002526	CLR NOFSMODE	
4176	024752	000207		RETURN	

4179 024754

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

4180	024754	005037	002362		CLR PASFLG		
4181	024760	012737	000030	002366	MTA030: MOV #30,REALPAT		;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4182	024766				CLEAR MARGIN,MAINT		;CLEAR MARGINS
	024766	005037	002112				CLR MARGIN
	024772	005037	177750				CLR MAINT
4183	024776	012737	000001	002100	MOV #1,NOPAR		;INDICATE COUNT PARITY ERRORS
4184	025004				BMOV MTP030		
	025004	004537	043710		JSR R5,BLOCK1		
	025010	034704			MTP030		
					.DSABL CRF		
4185	025012	104470			ECCDIS		;DISABLE ERROR CORRECTION
4186	025014				SET NOFSMODE,NOSCOPE		
	025014	012737	177777	002526			MOV #-1,NOFSMODE
	025022	012737	177777	002540			MOV #-1,NOSCOPE
4187	025030				FOR BANK := #0 TO #167		
	025030	005037	002106				CLR BANK
	025034						B42:;;;;;
4188	025034	004737	042422		CALL EXBANK		
4189	025040				IF MKFLAG IS TRUE		
	025040	005737	002126				TST MKFLAG
	025044	001414					BEQ L353
4190	025046				IF ACFLAG IS TRUE AND RRFLAG IS FALSE		
	025046	005737	002124				TST ACFLAG
	025052	001411					BEQ L354
	025054	005737	002132				TST RRFLAG
	025060	001006					BNE L354
4191	025062	012701	040000		MOV #SIZE,R1		
4192	025066	012700	060000		MOV #FIRST,R0		
4193	025072	004737	026042		CALL SUPD01		
4194	025076				END ;OF IF ACFLAG		
	025076						L354:;;;;;
4195	025076				END ;OF IF MKFLAG		
	025076						L353:;;;;;
4196	025076				END ;OF FOR		
	025076	005237	002106				INC BANK
	025102	023727	002106	000167			CMP BANK,#167
	025110	003751					BLE B42
	025112						E42:;;;;;
4197	025112				IF PASFLG IS FALSE		
	025112	005737	002362				TST PASFLG
	025116	001030					BNE L355
4198	025120				SET PASFLG		
	025120	012737	177777	002362			MOV #-1,PASFLG
4199	025126	104502			CLRCR		;CLEAR MK11 CSRS
4200	025130	004737	041130		CALL RELOCATE		
4201	025134				ON.ERROR		
	025134	103006					BCC L356
4202	025136	104472			ECCINIT		;TRAP ON DOUBLE BIT ERRORS (NORMAL)
4203	025140				CLEAR NOFSMODE,NOSCOPE		
	025140	005037	002526				CLR NOFSMODE
	025144	005037	002540				CLR NOSCOPE
4204	025150	000207			RETURN		
4205	025152				END ;OF ON.ERROR		

```

025152
4206 025152 013737 002400 002106      MOV  NEWBANK,BANK
4207 025160 004737 042422              CALL EXBANK
4208 025164 004737 024760              CALL  MTA030
4209 025170 104472                      ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
4210 025172 004737 041604              CALL  UNRELOCATE
4211 025176 000207                      RETURN
4212 025200                                END ;OF IF PASFLG
                                L356:::
4213 025200 104472                      ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
4214 025202                                CLEAR  NOFSMODE,NOSCOPE
                                L355:::
                                CLR  NOFSMODE
                                CLR  NOSCOPE
4215 025212 000207                      RETURN
```

4218 025214

MT0031: SUBTST <<MT0031 SETUP SOB-A-LONG TEST>>

 :*SUBTEST MT0031 SETUP SOB-A-LONG TEST
 :*****

4219 025214 004737 026006
 4220 025220 103001
 025222 000207
 025224

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
 ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
 BCC L357
 RTS PC
 L357:::~::~

4221 025224 012737 177777 002540
 025224 012737 000031 002366
 4222 025232 012737 000031 002366
 4223 025240 005037 002100
 4224 025244 010346
 025246 013703 002106
 025252 004737 040700

SET NOSCOPE
 MOV #31,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 CLR NOPAR ;SETUP PARITY ACTION
 MAP BANK ;MAP FIRST SO BLOCK MOVE WORKS
 MOV R3,-(SP)
 MOV (SP)+,R3

4225 025256 012603
 025260 052737 040000 177776

SUPERVISOR ;ENTER SUPERVISOR MODE
 BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF

4226 025266 004537 043744
 025266 000027
 025272 060000
 025274 034714

BMOV MTP031,FIRST,SOBLENGTH/2
 JSR R5,BLOCK3
 SOBLENGTH/2
 FIRST
 MTP031
 .DSABL CRF

4227 025300 104417
 4228 025302 013702 002704
 4229 025306 010200
 4230 025310 012701 100776
 4231 025314 012705 060056
 4232 025320 012737 060002 002360
 4233 025326 004737 026234
 4234 025332 005037 002540
 4235 025336 000207

KERNEL ;ENTER KERNEL MODE
 MOV SOBK,R2
 MOV R2,R0
 MOV #100776,R1 ;COMPLEMENT OF INSTRUCTION "SOB R0,DOT"
 MOV #FIRST+SOBLENGTH,R5
 MOV #FIRST+2,SUPDOADD
 CALL SUPDO4
 CLR NOSCOPE
 RETURN

4238 025340

MT0033: SUBTST <<MT0033 SETUP WRITE RECOVERY TEST>>

 :*SUBTEST MT0033 SETUP WRITE RECOVERY TEST

4239 025340 004737 026006

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
 ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
 BCC L360
 RTS PC

4240 025344 103001
 025346 000207
 025350

L360:::~::~

4241 025350 012737 177777 002540

SET NOSCOPE ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 MOV #-1,NOSCOPE

4242 025356 012737 000033 002366

MOV #33,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 CLR NOPAR ;SETUP PARITY ACTION
 MAP BANK ;MAP FIRST SO THAT THE BLOCK MOVE WORKS

4243 025364 005037 002100

MOV R3,-(SP)

4244 025370 010346
 025372 013703 002106
 025376 004737 040700

MOV BANK,R3
 CALL MAPPER
 .DSABL CRF

4245 025402 012603
 025404 012700 010247

MOV #10247,R0 ;OPP CODE OF INSTRUCTION "MOV R2,-(PC)"
 MOV #177667,R1 ;OPP CODE OF COMPLEMENT OF INSTRUCTION "JMP (R0)"
 MOV #SIZE/2,R2 ;USED FOR 1/2 BANK LOOP
 MOV #FIRST,R3
 MOV #LAST+2,R4

4246 025410 012701 177667

4247 025414 012702 020000

4248 025420 012703 060000

4249 025424 012704 160000

4250
 4251 025430 052737 040000 177776

SUPERVISOR ;ENTER SUPERVISOR MODE
 BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF

4252
 4253 025436 010023

1\$: ;MOVE TEST TO MEMORY UNDER TEST

4254 025440 010144

MOV R0,(R3)+
 MOV R1,-(R4)
 SOB R2,1\$

4255 025442 077203
 4256
 4257

;MOVE LAST PART OF TEST TO FASTCITY

4258 025444 004537 043710
 025444 035054

BMOV MTP033
 JSR R5,BLOCK1
 MTP033
 .DSABL CRF

4259 025452 104417
 4260

KERNEL ;ENTER KERNEL MODE

4261 025454 012702 005141

MOV #5141,R2 ;OPP CODE OF INSTRUCTION "COM -(R1)"
 MOV #2\$,R0 ;ADDRESS TO RETURN TO IN R0
 MOV #LAST+2,R1 ;TOP OF BANK

4262 025460 012700 025526
 4263 025464 012701 160000 002360

4264 025470 012737 060000

MOV #FIRST,SUPDOADD

4265 025476 004737 026234

CALL SUPDO4

4266 025502 012703 020000

MOV #SIZE/2,R3

4267 025506 012705 000110

MOV #110,R5

4268 025512 012704 060000

MOV #FIRST,R4

4269 025516 004737 026056

CALL SUPDO2

4270 025522 005037 002540

CLR NOSCOPE

4271 025526 000207

2\$: RETURN ;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0033
 ;ALSO A RETURN FROM THE "CALL SUPDO4" ABOVE

4272
 4273

```

4276 025530          MT0034: SUBTST <<MT0034      SETUP BRANCH GOBBLE TEST>>
:*****
:*SUBTEST          MT0034 SETUP BRANCH GOBBLE TEST
:*****
4277 025530 004737 026006          CALL    KAMITEST          ;CHECK FOR KAMIKAZE MODE
4278 025534          ON.ERROR THEN $RETURN          ;IF NOT IN KAMIKAZE MODE RETURN
          025534 103001          BCC L361
          025536 000207          RTS PC
          025540          L361:;;;;;
4279 025540          SET      NOSCOPE
          025540 012737 177777 002540          MOV    #-1,NOSCOPE
4280 025546 012737 000034 002366          MOV    #34,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4281 025554 005037 002100          CLR   NOPAR          ;SETUP PARITY ACTION
4282 025560          MAP    BANK          ;MAP FIRST SO THAT BLOCK MOVE WORKS
          025560 010346          MOV    BANK,R3
          025562 013703 002106          CALL  MAPPER
          025566 004737 040700          .DSABL CRF
          MOV (SP)+,R3
          025572 012603
4283 025574          SUPERVISOR          ;ENTER SUPERVISOR MODE
4284 025574 052737 040000 177776          BIS    #BIT14,PSW          ;DO IT TO IT!
          .DSABL CRF
4285 025602          BMOV   MTP034,FIRST,GBLENGTH/2
          025602 004537 043744          JSR R5,BLOCK3
          025606 000037          GBLENGTH/2
          025610 060000          FIRST
          025612 035106          MTP034
          .DSABL CRF
4286 025614 104417          KERNEL          ;ENTER KERNEL MODE
4287
4288 025616 012705 060076          MOV    #FIRST+GBLENGTH,R5
4289 025622 012737 060004 002360          MOV    #FIRST+4,SUPDOADD
4290 025630 012701 060002          MOV    #FIRST+2,R1
4291 025634 012702 060003          MOV    #FIRST+3,R2
4292
4293 025640 004737 026234          CALL  SUPD04
4294 025644 005037 002540          CLR   NOSCOPE
4295 025650 000207          RETURN
4296
4297 025652          MT0035: SUBTST <<MT0035      SOFT ERROR - BACKGROUND PATTERN TEST>>
:*****
:*SUBTEST          MT0035 SOFT ERROR - BACKGROUND PATTERN TEST
:*****
4298 025652 012737 000035 002366          MOV    #35,REALPAT
4299 025660 012700 060000          MOV    #FIRST,R0
4300 025664 012701 040000          MOV    #SIZE,R1
4301 025670 013702 002744          MOV    SOFTPAT,R2
4302 025674 010103          MOV    R1,R3
4303 025676 013705 002106          MOV    BANK,R5
4304 025702 070527 000006          MUL   #6,R5
4305 025706 010004          MOV    R0,R4
4306 025710          BMOV   MTP000
          025710 004537 043710          JSR R5,BLOCK1
          025714 026376          MTP000
          .DSABL CRF
4307 025716 012737 000207 177644          MOV    #207,UIPAR2          ;PUT RETURN INSTRUCTION AFTER WRITE ROUTINE
    
```

```

4308 025724 012737 177646 002360      MOV    #UIPAR3,SUPDOADD
4309 025732      IF #BIT13 SET.IN CONFIG+2(R5) AND WRITEONLY IS FALSE
      025732 032765 020000 003020      BIT #BIT13,CONFIG+2(R5)
      025740 001406      BEQ L362
      025742 005737 002502      TST WRITEONLY
      025746 001003      BNE L362

4310      ;BACKGROUND PATTERN IS VALID
4311 025750 004737 026220      CALL SUPD03      ;READ IT
4312 025754      ELSE
      025754 000405
      025756

4313      ;BACKGROUND PATTERN HAS BEEN INVALIDATED
4314 025756 004737 026042      CALL SUPD01      ;WRITE IT
4315 025762 052765 020000 003020      BIS #BIT13,CONFIG+2(R5) ;VALIDATE IT
4316 025770      END ;OF IF #BIT13
      025770
4317 025770 000207      RETURN

L362:::
BR L363
L363:::

```

```
4320 025772          MT0999: SUBTST <<MT0999          SETUP NULL TEST>>
:*****
:*SUBTEST          MT0999 SETUP NULL TEST
:*****
4321 025772 005037 002366          CLR          REALPAT
4322 025776 012737 177777 002436  SET          NULLFLAG
4323 026004 000207          RETURN
4324
4325 026006          KAMITEST:SUBTST <<CHECK FOR KAMIKAZE MODE>>
:*****
:*SUBTEST          CHECK FOR KAMIKAZE MODE
:*****
4326 026006 005737 002010          IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE
026006 001006          TST KAMIKAZE
026012 005737 002442          BNE L364
026014 001003          TST ACTFLAG
026020 005737 002444          BNE L364
026022 001403          TST APTFLAG
026026 001403          BEQ L365
026030          L364:~~~~~
4327 026030          $RETURN NOERROR          ;RUN THE TEST
026030 000241          CLC
026032 000207          RTS PC
4328 026034          ELSE
026034 000402          BR L366
026036          L365:~~~~~
4329 026036          $RETURN ERROR          ;DON'T RUN THE TEST
026036 000261          SEC
026040 000207          RTS PC
4330 026042          END ;OF IF KAMIKAZE
026042          L366:~~~~~
```

```

4334 026042          SUPD01: SUBTST <<SUBR EXECUTE PATTERN IN SUPERVISOR>>
:*****
:*SUBTEST          SUBR EXECUTE PATTERN IN SUPERVISOR
:*****
4335 026042          MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
026042 010346          MOV R3,-(SP)
026044 013703 002106  CALL BANK,R3
026050 004737 040700  .DSABL MAPPER
                                CRF
                                MOV (SP)+,R3
4336 026054 012603          SUPD02: CALL GETDIS
4337 026056 004737 066514  PUSH $LPERR,$LPADR
                                MOV $LPERR,-(SP)
                                MOV $LPADR,-(SP)
026062 013746 002750
026066 013746 002746
4338 026072 010037 002266  MOV R0,SUPDR0
4339 026076 012700 002270  MOV #SUPDR1,R0
4340 026102 010120          MOV R1,(R0)+
4341 026104 010220          MOV R2,(R0)+
4342 026106 010320          MOV R3,(R0)+
4343 026110 010420          MOV R4,(R0)+
4344 026112 010520          MOV R5,(R0)+
4345 026114 010620          MOV SP,(R0)+
4346 026116 013700 002266  MOV SUPDR0,R0
4347 026122 012737 026136 002746  MOV #TAG4$, $LPADR
4348 026130 013737 002746 002750  MOV $LPADR,$LPERR
4349 026136 012700 002304  TAG4$: MOV #SUPDR6+2,R0
4350 026142 014006          MOV -(R0),SP
4351 026144 014005          MOV -(R0),R5
4352 026146 014004          MOV -(R0),R4
4353 026150 014003          MOV -(R0),R3
4354 026152 014002          MOV -(R0),R2
4355 026154 014001          MOV -(R0),R1
4356 026156 014000          MOV -(R0),R0
4357 026160          SUPERVISOR ;ENTER SUPERVISOR MODE
026160 052737 040000 177776  BIS #BIT14,PSW ;DO IT TO IT!
                                .DSABL CRF
4358 026166 012706 000740  MOV #SUPSTK,SSP
4359 026172 104424          CACHOFF ;TURN CACHE OFF
4360 026174 004737 177640  CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
4361 026200 104423          CACHON ;TURN CACHE ON
4362 026202 104417          KERNEL ;ENTER KERNEL MODE
4363 026204 000004          SCOPE
4364 026206          POP $LPADR,$LPERR
                                MOV (SP)+,$LPADR
                                MOV (SP)+,$LPERR
026206 012637 002746
026212 012637 002750
4365 026216 000207          RETURN
    
```


4403
 4404
 4405
 4406
 4407
 4408
 4409
 4410
 4411
 4412
 4413 026376

4414 026376 010220
 4415 026400 077102
 4416 026402 000240
 4417 026404 012401
 4418 026406 020102
 4419 026410 001402
 4420 026412 104430
 4421 026414 000240
 4422 026416 077307
 4423 026420 000207
 4424 026422

4425 026422 010220
 4426 026424 062702 000002
 4427 026430 077104
 4428 026432 000240
 4429 026434 012400
 4430 026436 020005
 4431 026440 001401
 4432 026442 104427
 4433 026444 062705 000002
 4434 026450 077307
 4435 026452 000207
 4436 026454

4437 026454 010540
 4438 026456 062705 000002
 4439 026462 077104
 4440 026464 000240
 4441 026466 162702 000002
 4442 026472 012401
 4443 026474 020102
 4444 026476 001401
 4445 026500 104430
 4446 026502 077307
 4447 026504 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
    
```

4451 026506

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

4452
 4453
 4454
 4455
 4456
 4457
 4458
 4459 026506 060001
 4460 026510 010421
 4461 026512 010421
 4462 026514 077204
 4463 026516 005104
 4464 026520 052704
 4465 026522 000401
 4466 026524 012702 000004
 4467 026530 077512
 4468 026532 005104
 4469 026534 052704
 4470 026536 000401
 4471 026540 012705 000100
 4472 026544 077320
 4473 026546 000207
 4474
 4475
 4476 026550

```

:RO = INTERLEAVE FACTOR
:R1 = ADDRESS
:R2 = SMALL LOOP CONSTANT
:R3 = NUM OF ADD TO TEST (LARGE LOOP)
:R4 = GOOD DATA
:R5 = MEDIUM LOOP CONSTANT
.ENABL LSB
1$: ADD R0,R1 ;V177636 ADD INTERLEAVE FACTOR
MOV R4,(R1)+ ;V177640
MOV R4,(R1)+ ;V177642
SOB R2,1$ ;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)
 :*****

4477
 4478 026550 000137 172360
 4479 026554 077203
 4480 026556 005104
 4481 026560 052704
 4482 026562 000401
 4483 026564 012702 000004
 4484 026570 077511
 4485 026572 005104
 4486 026574 052704
 4487 026576 000401
 4488 026600 012705 000100
 4489 026604 077317
 4490 026606 000207
 4491

```

.ENABL LSB
1$: JMP KDPAR0 ;V177640 GO TO V172360
SOB R2,1$ ;V177644
COM R4 ;V177646
BIS (PC)+,R4 ;V177650
WARN4: 401 ;V177652 WARNING LOCATION IS MODIFIED BEFORE 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
    
```

4494 026610
4495 026610 060001
4496 026612 020421
4497 026614 001401
4498 026616 104431
4499 026620 005141
4500 026622 005111
4501 026624 000137 172260
4502
4503 026630

```
MTPC03: SUBTST <<MTPC03 TEST DATA SUBPROGRAM>>  
:*****  
:*SUBTEST MTPC03 TEST DATA SUBPROGRAM  
:*****  
          ADD R0,R1 :V172360  
          CMP R4,(R1)+ :V172362  
          BEQ 1$ :V172364  
          PERR03 :V172366  
1$:      COM -(R1) :V172370  
          COM (R1) :V172372  
          JMP SDPAR0 :V174372 GO TO V172260
```

4504 026630 020421
4505 026632 001401
4506 026634 104431
4507 026636 005127
4508 026640 000000
4509 026642 001033
4510 026644 000137 177644

```
MTPD03: SUBTST <<MTPD03 TEST DATA SUBSUBPROGRAM>>  
:*****  
:*SUBTEST MTPD03 TEST DATA SUBSUBPROGRAM  
:*****  
          CMP R4,(R1)+ :V172260  
          BEQ 1$ :V172262  
          PERR03 :V172264  
1$:      COM (PC)+ :V172266  
          O :V172270  
          BNE +70 :V172272 GO TO V172362  
          JMP UIPAR2 :V172274 GO TO V177644
```

4513 026650

MTPA04: SUBTST <<MTPA04 ROTATING ZEROS TEST>>

:*SUBTEST MTPA04 ROTATING ZEROS TEST

4514 026650 012705 000010
4515 026654 010504
4516 026656 000241
4517 026660 000137 172360
4518 026664 016004 177776
4519 026670 103402
4520 026672 020204
4521 026674 001401
4522 026676 104432
4523 026700 077115
4524 026702 000207
4525
4526 026704

1\$: MOV #8,R5 ;V177640
MOV R5,R4 ;V177644
CLC ;V177646
JMP KDPAR0 ;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

MTPB04: SUBTST <<MTPB04 SUBR ROTATING BIT>>

:*SUBTEST MTPB04 SUBR ROTATING BIT

4527 026704 106110
4528 026706 077502
4529 026710 106120
4530 026712 106110
4531 026714 077402
4532 026716 106120
4533 026720 000137 177654
4534
4535 026724

1\$: ROLB (R0) ;V172360
SOB R5,1\$;V172362
ROLB (R0)+ ;V172364
2\$: ROLB (R0) ;V172366
SOB R4,2\$;V172370
ROLB (R0)+ ;V172372
JMP UIPAR6 ;V172374

MTP005: SUBTST <<MTP005 ROTATION ONES TEST>>

:*SUBTEST MTP005 ROTATION ONES TEST

4536 026724 012705 000010
4537 026730 010504
4538 026732 000261
4539 026734 000137 172360
4540 026740 016004 177776
4541 026744 103002
4542 026746 020204
4543 026750 001401
4544 026752 104432
4545 026754 077115
4546 026756 000207

1\$: MOV #8,R5 ;V177640
MOV R5,R4 ;V177644
SEC ;V177646
JMP KDPAR0 ;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

4550 026760

MTP006: SUBST <<MTP006 INITIAL DATA TEST>>

: *SUBTEST MTP006 INITIAL DATA TEST
: *****

: THIS TEST CHECKS THE DI/DO LINES BY
: SHIFTING A 1 THROUGH THE WORD.

```

4551
4552
4553 026760 012737 000001 002340      MOV #1,DATBUF      ;SET THE FIRST TEST BIT
4554 026766 005037 002342              CLR DATBUF+2       ;CLEAR 2ND WORD
4555 026772 013711 002340      1$: MOV DATBUF,(R1)  ;WRITE TEST WORD 1
4556 026776 013761 002342 000002      MOV DATBUF+2,2(R1) ;AND TEST WORD 2
4557 027004 011102              MOV (R1),R2
4558 027006 023702 002340      CMP DATBUF,R2     ;NOW READ THEM
4559 027012 001401              BEQ 2$            ;BR IF FIRST 16 OK
4560 027014 104433              PERR07           ;ERROR TRAP
4561
4562 027016 016102 000002      2$: MOV 2(R1),R2
4563 027022 023702 002342      CMP DATBUF+2,R2  ;NOW READ SECOND WORD
4564 027026 001401              BEQ 3$            ;BR IF OK
4565 027030 104434              PERR10           ;ERROR TRAP
4566
4567 027032 005737 002342      3$: TST DATBUF+2   ;HAS LAST BIT BEEN TESTED ?
4568 027036 100405              BMI 4$            ;MINUS MEANS BIT 31
4569 027040              DLEFT DATBUF     ;NO, SHIFT TEST BIT LEFT
         027040 006137 002340      ROL DATBUF
         027044 006137 002342      ROL DATBUF+2
         .DSABL CRF
4570 027050 000750              BR 1$            ;GO WRITE NEW TEST DATA
4571
4572 027052 012737 177776 002340  4$: MOV #177776,DATBUF ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
4573 027060 012737 177777 002342      MOV #-1,DATBUF+2 ;PUT A 0 IN BIT 0
4574 027066 013711 002340      5$: MOV DATBUF,(R1)  ;AND 1'S IN ALL OTHERS
4575 027072 013761 002342 000002      MOV DATBUF+2,2(R1) ;WRITE THE DATA
4576 027100 011102              MOV (R1),R2      ;2 WORDS WORTH
4577 027102 023702 002340      CMP DATBUF,R2   ;NOW READ FIRST WORD
4578 027106 001401              BEQ 6$            ;BR IF OK
4579 027110 104433              PERR07
4580
4581 027112 016102 000002      6$: MOV 2(R1),R2
4582 027116 023702 002342      CMP DATBUF+2,R2 ;NOW, READ SECOND WORD
4583 027122 001401              BEQ 7$            ;BR IF OK
4584 027124 104434              PERR10
4585
4586 027126 005737 002342      7$: TST DATBUF+2   ;TESTED BIT 31 YET?
4587 027132 100005              BPL 8$            ;BR IF YES, WE'RE DONE
4588 027134              DLEFT DATBUF
         027134 006137 002340      ROL DATBUF
         027140 006137 002342      ROL DATBUF+2
         .DSABL CRF
4589 027144 000750              BR 5$
4590 027146 000207      8$: RETURN       ;KEEP GOING

```

4594 027150

MTP007: SUBST <<MTP007 ADDRESS BIT TEST>>

:*SUBTEST MTP007 ADDRESS BIT TEST

4595
4596
4597
4598
4599 027150 111100
4600 027152 105700
4601 027154 001401
4602 027156 104435
4603
4604 027160 105111
4605 027162 111100
4606 027164 105700
4607 027166 001001
4608 027170 104436
4609
4610 027172 040201
4611 027174 006302
4612 027176 050201
4613 027200 011100
4614 027202 005700
4615 027204 001401
4616 027206 104437
4617
4618 027210 005111
4619 027212 011100
4620 027214 005700
4621 027216 001001
4622 027220 104440
4623
4624 027222 022702 100000
4625 027226 001407
4626 027230 022702 010000
4627 027234 001356
4628 027236 006302
4629 027240 012701 160000
4630 027244 000752
4631 027246 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

```

```

4634 027250      MTP010: SUBTST <<MTP010      BYTE ADDRESSING TEST>>
:*****
:*SUBTEST      MTP010  BYTE ADDRESSING TEST
:*****
4635      ;TEST 3 THIS TEST CHECKS FOR PROPER
4636      ;      BYTE ADDRESSING WITH ECC DISABLED
4637 027250 010402      MOV      R4,R2      ;R4 HAS LOWEST ADDRESS
4638 027252 010403      MOV      R4,R3      ;PUT IT IN R3 ALSO
4639 027254 062702 000004      ADD      #4,R2      ;POINT R2 TO LAST BYTE +1
4640 027260 012713 177777      MOV      #-1,(R3)    ;WRITE ALL ONES IN
4641 027264 012763 177777 000002      MOV      #-1,2(R3)  ;THE 4 TEST BYTES
4642 027272 105013      1$:      CLRB     (R3)    ;CLEAR A BYTE
4643 027274 010401      MOV      R4,R1      ;INITIALIZE R1 FOR EACH PASS
4644 027276 020201      2$:      CMP      R2,R1      ;IF EQUAL, JUST READ LAST BYTE
4645 027300 001420      BEQ      6$         ;BR IF EQUAL
4646 027302 020301      CMP      R3,R1      ;IS THIS THE BYTE OF ZEROS
4647 027304 001007      BNE      4$         ;BR IF NOT
4648 027306 111100      MOVB     (R1),R0
4649      ;WARNING IF YOU OPTOMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
4650 027310 022700 000000      CMP      #0,R0      ;IT IS, COMPARE FOR ZEROS
4651 027314 001401      BEQ      3$
4652 027316 104435      PERR11
4653
4654 027320 005201      3$:      INC      R1          ;NEXT BYTE
4655 027322 000765      BR       2$          ;RETURN
4656 027324 111100      4$:      MOVB     (R1),R0
4657 027326 122700 177777      CMPB     #-1,R0      ;ITS NOT THE BYTE OF 0'S, READ 1'S
4658 027332 001401      BEQ      5$
4659 027334 104436      PERR12
4660
4661 027336 005201      5$:      INC      R1          ;MOVE TO NEXT BYTE
4662 027340 000756      BR       2$
4663 027342 112713 177777      6$:      MOVB     #-1,(R3)   ;RESTORE 1'S TO BYTE JUST TESTED
4664 027346 005203      INC      R3          ;INC TO NEXT BYTE
4665 027350 020302      CMP      R3,R2      ;WAS THAT JUST THE LAST ONE?
4666 027352 001347      BNE      1$         ;BR IF NO
4667 027354 000207      RETURN
  
```

4671 027356

MTP011: SUBTST <<MPT011 SINGLE BIT ERROR TEST>>

4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688 027356 104503
4689
4690 027360 012737 000001 002340 1\$:
4691 027366 005037 002342
4692
4693 027372 012737 000001 002350 2\$:
4694 027400 005037 002352
4695
4696 027404 013737 002340 002344 3\$:
4697 027412 013737 002342 002346
4698 027420 105737 002362
4699 027424 001404
4700 027426 005137 002344
4701 027432 005137 002346
4702 027436 013702 002344 4\$:
4703 027442 013703 002346
4704 027446 012737 002344 002402
4705 027454 004737 040540
4706
4707
4708
4709 027460 013701 002350
4710 027464 074137 002344
4711 027470 013701 002352
4712 027474 074137 002346
4713 027500 013701 002506 5\$:
4714 027504 104471
4715 027506 013721 002344
4716 027512 104475
4717 027514 013711 002346
4718
4719
4720
4721 027520 104471
4722 027522 014100
4723 027524 020037 002344
4724 027530 001401

```

*****
*SUBTEST      MPT011  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 FORM 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 MK11 CSR
:BIG LOOP
MOV          #1,DATBUF      ;INITIAL DATA
CLR          DATBUF+2      ;32 BITS WORTH
:MEDIUM LOOP
MOV          #1,SBEMSK     ;INITIAL ERROR MASK
CLR          SBEMSK+2     ;32 BITS WORTH
:LITTLE LOOP
MOV          DATBUF,TSTDAT  ;
MOV          DATBUF+2,TSTDAT+2;TO SAVE ORIG DATA
TSTB        PASFLG       ;COMP DATA ON SECOND PASS ONLY
BEQ          4$           ;BR IF FIRST PASS
COM          TSTDAT        ;SECOND PASS, COMP BOTH WORDS
COM          TSTDAT+2
4$:          MOV          TSTDAT,R2
             MOV          TSTDAT+2,R3
             MOV          #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
             CALL        CHKGEN       ;GEN CHECKBITS ON TSTDAT
*****
** CREATE A SINGLE BIT ERROR **
*****
MOV          SBEMSK,R1
XOR          R1,TSTDAT
MOV          SBEMSK+2,R1
XOR          R1,TSTDAT+2
5$:          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 AND
             ;CHECK BITS. WE NOW HAVE CHECKBITS
             ;GENERATED ON DATBUF AND DATA WITH
             ;ONE BIT IN ERROR (AS PER SBEMSK).
             ;DISABLE ECC ON 1 SELECTED CSR
             ECC1DIS
             MOV          -(R1),R0
             CMP          R0,TSTDAT   ;READ THE LOW WORD (UNCORRECTED)
             BEQ          6$         ;BR IF OK

```

```

4725 027532 104455 PERR31
4726
4727 027534 062701 000002 6$: ADD #2,R1 ;POINT R1 TO SECOND 16 BITS
MOV (R1),R0
4728 027540 011100
4729 027542 020037 002346 CMP R0,TSTDAT+2 ;READ THE HIGH WORD (UNCORRECTED)
4730 027545 001401 BEQ 7$ ;BR IF OK
4731 027550 104455 PERR31
4732
4733 027552 104503 7$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
MOV -(R1),R0
4734 027554 014100 CMP R0,R2 ;SEE IF ITS BEEN CORRECTED
4735 027556 020002 BEQ 8$ ;IT SHOULD HAVE BEEN
4736 027560 001401 PERR32
4737 027562 104456
4738
4739 027564 104510 8$: TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
BCS 9$ ;BR IF IT IS SET
4740 027566 103411 SET HEADER ;ENABLE PRINTING OF ERROR HEADER INFO
4741 027570 012737 177777 002724 MOV #-1,HEADER
027570 010137 002040 MOV R1,ADDRESS
4742 027576 010137 002040 PERR34
4743 027602 104460 SET HEADER ;ENABLE PRINTING OF ERROR HEADER INFO
4744 027604 012737 177777 002724 MOV #-1,HEADER
027604 012737 177777 002724
4745
4746 027612 104503 9$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
ADD #2,R1 ;POINT TO HIGH WORD
4747 027614 062701 000002 MOV (R1),R0
4748 027620 011100 CMP R0,R3 ;SEE IF ITS BEEN CORRECTED
4749 027622 020003 BEQ 10$ ;BR IF OK
4750 027624 001401 PERR32
4751 027626 104456
4752
4753 027630 104510 10$: TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
BCS 11$ ;BR IF YES
4754 027632 103411 SET HEADER ;ENABLE PRINTING OF ERROR HEADER INFO
4755 027634 012737 177777 002724 MOV #-1,HEADER
027634 010137 002040 MOV R1,ADDRESS
4756 027642 010137 002040 PERR34
4757 027646 104460 SET HEADER ;ENABLE PRINTING OF ERROR HEADER INFO
4758 027650 012737 177777 002724 MOV #-1,HEADER
027650 012737 177777 002724
4759 027656 005737 002352 11$: TST SBEMSK+2 ;TEST FOR LAST MASK BIT
4760 027662 100405 BMI 12$ ;MINUS MEANS BIT 31
4761 027664 006137 002350 DLEFT SBEMSK
027664 006137 002352 ROL SBEMSK
ROL SBEMSK+2
.DSABL CRF
BR 3$
4762 027674 000643
4763 027676 032777 004000 12$: IF #SW11 SET.IN @SWR THEN GOTO 13$
027676 001013 BIT #SW11,@SWR
027704 001013 BNE 13$
4764 027706 IF QVFLAG IS TRUE THEN GOTO 13$
027706 005737 002440 TST QVFLAG
027712 001010 BNE 13$
4765 027714 005737 002342 TST DATBUF+2 ;LAST DATA BIT ?
4766 027720 100405 BMI 13$ ;WHICH IS BIT 31
4767 027722 006137 002340 DLEFT DATBUF
027722 006137 002342 ROL DATBUF
ROL DATBUF+2
.DSABL CRF

```

4768 027732 000617
 4769 027734 105737 002362
 4770 027740 001003
 4771 027742 105237 002362
 4772 027746 000604
 4773
 4774 027750 104471
 4775 027752 013701 002506
 4776 027756
 027756 005021
 027760 005011
 4777 027762 104503
 4778 027764 000207

13\$: BR 2\$
 TSTB PASFLG ;FIRST OR SECOND PASS ?
 BNE 14\$;NON ZERO MEANS WE'RE DONE
 INCB PASFLG ;NOT DONE, GO DO SECOND PASS
 BR 1\$
 ;CLEAR OUT ANY DBE'S OR SBE'S
 14\$: ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
 MOV TESTADD,R1
 CLEAR (R1)+,(R1)

CLR (R1)+
 CLR (R1)

CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
 RETURN

4782 027766

MTP012: SUBTST <<MTP012 WRITE BYTE CLEARS SBE TEST>>
:*****
:*SUBTEST MTP012 WRITE BYTE CLEARS SBE TEST
:*****

```

4783                                     :SINGLE BIT ERROR TEST TO INSURE THAT A WRITE
4784                                     :BYTE CLEARS SINGLE BIT ERRORS.
4785 027766 104503                       CLR1CSR                               :CLEAR 1 SELECTED MK11 CSR
4786 027770 012737 000001 002340         MOV #1,DATBUF                         :INITIAL DATA
4787 027776 005037 002342                 CLR DATBUF+2                          :32 BITS WORTH
4788 030002 012737 000001 002350 1$:    MOV #1,SBEMSK                          :INITIAL ERROR MASK
4789 030010 005037 002352                 CLR SBEMSK+2                          :32 BITS WORTH
4790 030014 013737 002340 002344 2$:    MOV DATBUF,TSTDAT                      :SAVE ORIGINAL DATA
4791 030022 013737 002342 002346         MOV DATBUF+2,TSTDAT+2                 :BOTH WORDS
4792 030030 012737 002344 002402         MOV #TSTDAT,SOURCE                    :NEED ADDRESS FOR CHKGEN
4793 030036 004737 040540                 CALL CHKGEN                            :GENERATE CHECK BITS
4794 030042 013701 002350                 MOV SBEMSK,R1
4795 030046 074137 002344                 XOR R1,TSTDAT
4796 030052 013701 002352                 MOV SBEMSK+2,R1
4797 030056 074137 002346                 XOR R1,TSTDAT+2
4798 030062 013704 002506 3$:           MOV TESTADD,R4                        :FIRST TEST ADDRESS
4799 030066 010401                         MOV R4,R1                             :PUT IT IN R1 ALSO
4800 030070 104471                         ECC1DIS                               :DISABLE ECC ON 1 SELECTED CSR
4801 030072 013721 002344                 MOV TSTDAT,(R1)+                      :WRITE 16 BITS
4802 030076 104475                         CB1CSR                               :WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
4803 030100 013711 002346                 MOV TSTDAT+2,(R1)                    :WRITE HIGH WORD+CHECKBITS
4804 030104 104503                         CLR1CSR                               :CLEAR 1 SELECTED MK11 CSR
4805                                     :IT'S DANGEROUS IF WE DON'T
4806 030106 012702 002350                 MOV #SBEMSK,R2                       :ADDRESS OF ERROR MASK
4807 030112 162701 000002                 SUB #2,R1                             :ADJUST ADDRESS
4808 030116 112711 177777 4$:           MOVB #-1,(R1)                         :WRITE A BYTE OF 1'S
4809 030122 132712 177777                 BITB #-1,(R2)                        :DID THIS BYTE HAVE THE BAD BIT IN IT?
4810 030126 001420                         BEQ 6$                                :NO - BRANCH
4811 030130 104510                         TSTREAD                               :TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4812 030132 103011                         BCC 5$                                :NO - SKIP
4813 030134 012737 177777 002724         SET HEADER                            :ENABLE PRINTING OF ERROR HEADER INFO
4814 030142 010137 002040                         MOV R1,ADDRESS                        MOV #-1,HEADER
4815 030146 104017                         ERROR +17
4816 030150 012737 177777 002724         SET HEADER                            :ENABLE PRINTING OF ERROR HEADER INFO
4817                                     MOV #-1,HEADER
4818 030156 111100 5$:                   MOVB (R1),R0
4819 030160 122700 177777                 CMPB #-1,R0                          :CHECK DATA
4820 030164 001414                         BEQ 7$                                :BR IF OK
4821 030166 104457                         PERR33
4822                                     :TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4823 030170 104510 6$:                   TSTREAD                               :READ THE BYTE
4824                                     :SBE ERROR BIT ONLY SET ?
4825                                     :SHOULD BE SET, BR IF OK
4826 030172 103771                         BCS 5$                                :ENABLE PRINTING OF ERROR HEADER INFO
4827 030174 012737 177777 002724         SET HEADER                            MOV #-1,HEADER
4828 030202 010137 002040                         MOV R1,ADDRESS
4829 030206 104460                         PERR34
4830 030210 012737 177777 002724         SET HEADER                            :ENABLE PRINTING OF ERROR HEADER INFO
4831                                     MOV #-1,HEADER

```

```

4832 030216 132712 177777      7$:  BITB    #-1,(R2)      ;CHECK FOR LAST BYTE
4833 030222 001012              BNE     8$          ;
4834 030224 005202              INC     R2          ;
4835 030226 005201              INC     R1          ;MOVE TO NEXT BYTE
4836 030230 013704 002506      MOV     TESTADD,R4  ;FIRST TEST ADDRESS
4837 030234 032701 000002      BIT     #2,R1      ;TEST FOR LOWER WORD
4838 030240 001726              BEQ     4$          ;BR IF IT'S LOW 16 BITS
4839 030242 062704 000002      ADD     #2,R4      ;ADJUST POINTER FOR ERROR REPT.
4840 030246 000723              BR      4$
4841 030250 005737 002352      8$:  TST     SBEMSK+2  ;LAST ERROR BIT ?
4842 030254 100405              BMI     9$          ;MINUS MEANS BIT 31
4843 030256              DLEFT  SBEMSK
              030256 006137 002350  ROL     SBEMSK
              030262 006137 002352  ROL     SBEMSK+2
              .DSABL CRF
              BR      2$
4844 030266 000652
4845 030270              9$:  IF #SW11 SET.IN @SWR THEN GOTO 10$
              030270 032777 004000 152470
              030276 001013
4846 030300              IF QVFLAG IS TRUE THEN GOTO 10$
              030300 005737 002440
              030304 001010
4847 030306 005737 002342      TST     DATBUF+2   ;LAST DATA BIT?
4848 030312 100405              BMI     10$        ;MINUS = BIT 31
4849 030314              DLEFT  DATBUF
              030314 006137 002340  ROL     DATBUF
              030320 006137 002342  ROL     DATBUF+2
              .DSABL CRF
              BR      1$
4850 030324 000626
4851              ;CLEAR OUT ANY DBE'S OR SBE'S
4852 030326 104471              10$: ECC1DIS  ;DISABLE ECC ON 1 SELECTED CSR
4853 030330 013701 002506      MOV     TESTADD,R1
4854 030334              CLEAR  (R1)+,(R1)
              030334 005021
              030336 005011
4855 030340 104503              CLR   (R1)+
4856 030342 000207              CLR   (R1)
              CLR1CSR
              RETURN  ;CLEAR 1 SELECTED MK11 CSR
  
```

4860 030344

MTP013: SUBTST <<MTP013 CREATE DOUBLE BIT ERROR TEST>>
:*****
:*SUBTEST MTP013 CREATE DOUBLE BIT ERROR TEST
:*****

```

4861
4862 030344 104503          CLR1CSR          ;DOUBLE BIT ERROR FORCE TO CHECK DOUBLE ERROR LOGIC
4863 030346 013701 002506  MOV TESTADD,R1   ;CLEAR 1 SELECTED MK11 CSR
4864 030352 005037 002340  1$: CLR DATBUF     ;MAKE INITIAL DATA
4865 030356 005037 002342  CLR DATBUF+2     ;ALL ZEROS
4866 030362 012737 000001 002350 2$: MOV #1,SBEMSK   ;INITIAL SINGLE ERROR MASK
4867 030370 005037 002352  CLR SBEMSK+2     ;SECOND WORD
4868 030374 012737 000001 002354 3$: MOV #1,DBEMSK   ;INITIAL DOUBLE ERROR MASK
4869 030402 005037 002356  CLR DBEMSK+2     ;32 BITS HERE ALSO
4870 030406 013737 002340 002344 4$: MOV DATBUF,TSTDAT
4871 030414 013737 002342 002346  MOV DATBUF+2,TSTDAT+2
4872 030422 105737 002362  TSTB PASFLG ;NO COMPLEMENTING FIRST PASS
4873 030426 001404          BEQ 5$
4874 030430 005137 002344          COM TSTDAT ;COMP FIRST WORD
4875 030434 005137 002346          COM TSTDAT+2 ;SECOND WORD
4876 030440 104503          CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
4877 030442 023737 002350 002354 5$: CMP SBEMSK,DBEMSK ;CAN'T HAVE THE SAME ERROR BIT SET
4878 030450 001004          BNE 6$ ;IN BOTH MASKS
4879 030452 023737 002352 002356  CMP SBEMSK+2,DBEMSK+2 ;COULD BE EQUAL IN SECOND WORD
4880 030460 001452          BEQ 9$ ;GO MAKE THEM NOT EQUAL
4881 030462 012737 002344 002402 6$: MOV #TSTDAT,SOURCE ;SOURCE ADDRESS FOR CHKGEN
4882 030470 004737 040540          CALL CHKGEN ;GO GENERATE CHECK BITS
4883 030474 013702 002350          MOV SBEMSK,R2
4884 030500 074237 002344          XOR R2,TSTDAT
4885 030504 013702 002352          MOV SBEMSK+2,R2
4886 030510 074237 002346          XOR R2,TSTDAT+2
4887 030514 013702 002354          MOV DBEMSK,R2
4888 030520 074237 002344          XOR R2,TSTDAT
4889 030524 013702 002356          MOV DBEMSK+2,R2
4890 030530 074237 002346          XOR R2,TSTDAT+2
4891 030534 104471          ECC1DIS          ;DISABLE ECC ON 1 SELECTED CSR
4892 030536 013721 002344          MOV TSTDAT,(R1)+ ;WRITE 16 BITS
4893 030542 104475          CB1CSR          ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
4894 030544 013711 002346          MOV TSTDAT+2,(R1) ;WRITE HIGH WORD
4895 030550 104503          CLR1CSR          ;CLEAR 1 SELECTED MK11 CSR
4896 030552 162701 000002          SUB #2,R1 ;ADJUST TEST ADDRESS
4897 030556 005711          TST (R1) ;READ THE LOCATION
4898 030560 104501          WAS1DBE ;WAS THERE ANY DOUBLE BIT ERRORS ON 1 SELECTED CSR
4899 030562 103411          BCS 9$ ;IT SHOULD BE SET
4900 030564          SET HEADER
4901 030572 012737 177777 002724          MOV #-1,HEADER
4902 030576 010137 002040          MOV R1,ADDRESS
4903 030600 104030          ERROR +30
4903 030600 012737 177777 002724          SET HEADER
4903 030600          MOV #-1,HEADER

```

```

4906 030606 005737 002356      9$:  TST    DBEMSK+2      ;CHECK MASK FOR LAST BIT
4907 030612 100405              BMI    10$           ;MINUS = BIT31
4908 030614 006137 002354      DLEFT DBEMSK
      030614 006137 002354      ROL   DBEMSK
      030620 006137 002356      ROL   DBEMSK+2
      .DSABL CRF
4909 030624 000670              BR    4$
4910 030626 032777 004000 152132 10$:  IF #SW11 SET.IN @SWR THEN GOTO 11$
      030626 001013
      030634 001013
4911 030636 005737 002440      IF QVFLAG IS TRUE THEN GOTO 11$
      030636 001010
      030642 001010
4912 030644 005737 002352      TST    SBEMSK+2      ;CHECK SINGLE ERROR MASK TOO
4913 030650 100405              BMI    11$           ;BR IF DONE
4914 030652 006137 002350      DLEFT SBEMSK
      030652 006137 002352      ROL   SBEMSK
      030656 006137 002352      ROL   SBEMSK+2
      .DSABL CRF
4915 030662 000644              BR    3$
4916 030664 105737 002362      11$:  TSTB   PASFLG ;FIRST PASS
4917 030670 001003              BNE   12$           ;NON ZERO MEANS WE'RE DONE
4918 030672 105237 002362      INCB  PASFLG ;FIRST PASS, NOT DONE
4919 030676 000625              ;CLEAR OUT ANY DBE'S OR SBE'S
4920 030700 104471              BR    1$           ;KEEP GOING
4921 030702 013701 002506      12$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
4922 030706 005021              MOV   TESTADD,R1
4923 030706 005011              CLEAR (R1)+,(R1)
      030710 005011
4924 030712 104503              CLR 1 CSR
4925 030714 000207              CLR 1 CSR
      RETURN

```

4929 030716

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

4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982

030716 005037 002340
 030722 005037 002342
 030726 012737 000001 002350
 030734 005037 002352
 030740 012737 000001 002354
 030746 005037 002356
 030752 013737 002340 002344
 030760 013737 002342 002346
 030766 105737 002362
 030772 001404
 030774 005137 002344
 031000 005137 002346
 031004 104503
 031006 023737 002354 002350
 031014 001004
 031016 023737 002356 002352
 031024 001466
 031026 012737 002344 002402
 031034 004737 040540
 031040 013701 002350
 031044 074137 002344
 031050 013701 002352
 031054 074137 002346
 031060 013701 002354
 031064 074137 002344
 031070 013701 002356
 031074 074137 002346
 031100 013701 002506
 031104 104471
 031106 013721 002344
 031112 104475
 031114 013711 002346
 031120 105037 002363
 031124 013703 002506
 031130 104503
 031132 106223
 031134 014100
 031136 023700 002344
 031142 001401
 031144 104455
 031146 062701 000002
 031152 011100
 031154 023700 002346
 031160 001401
 031162 104455

```

*****
: *SUBTEST      MTP014 WRITE INHIBIT DURING DATIP WITH DBE TEST
*****
; THIS TEST CHECKS THE WRITE INHIBIT ON DOUBLE
; BIT ERRORS DURING A DATIP OPERATION BY USE
; OF AN 'ASRB' INSTRUCTION.
;
; NOTE: THIS IS ONLY A USEFULL TEST ON PDP11/70 WITH THE MULTIPOINT ECO
; IE THOSE THAT DO A DATIP ON THE ASRB (LOCK) INSTRUCTION.
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 ?
   BEQ 5$ ;BR IF NO
   COM TSTDAT ;COMPL DATA ON SECOND PASS
   COM TSTDAT+2
5$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
   CMP DBEMSK,SBEMSK ;CHECK FOR SAME MASKS
   BNE 6$ ;BR IF OK
   CMP DBEMSK+2,SBEMSK+2
   BEQ 11$ ;BR IF THEY'RE EQUAL
6$: MOV #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
   CALL CHKGEN ;GENERATE CHECK BITS
   MOV SBEMSK,R1
   XOR R1,TSTDAT
   MOV SBEMSK+2,R1
   XOR R1,TSTDAT+2
   MOV DBEMSK,R1
   XOR R1,TSTDAT
   MOV DBEMSK+2,R1
   XOR R1,TSTDAT+2
7$: 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$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
   ASRB (R3)+ ;SPECIAL DATIP INSTRUCTION
   MOV -(R1),R0
   CMP TSTDAT,R0 ;CHECK FOR UNCHANGED DATA
   BEQ 9$ ;SHOULD BE UNCHANGED
   PERR31
9$: ADD #2,R1 ;POINT TO UPPER WORD
   MOV (R1),R0
   CMP TSTDAT+2,R0 ;READ IT
   BEQ 10$ ;BR IF UNCHANGED
   PERR31

```

```

4983 031164 122737 000003 002363 10$:  CMPB  #3,UPPFLG ;LOWER WORD
4984 031172 001403  BEQ  11$ ;BR IF NO
4985 031174 105237 002363  INCB  UPPFLG
4986 031200 000753  BR  8$
4987 031202 005737 002356 11$:  TST  DBEMSK+2 ;LAST BIT IN MASK ?
4988 031206 100405  BMI  12$ ;BR IF BIT 31
4989 031210  DLEFT DBEMSK
      031210 006137 002354  ROL  DBEMSK
      031214 006137 002356  ROL  DBEMSK+2
      .DSABL CRF
4990 031220 000654  BR  4$
4991 031222  IF #SW11 SET.IN @SWR THEN GOTO 13$ 12$:
      031222 032777 004000 151536  BIT #SW11,@SWR
      031230 001013  BNE 13$
4992 031232  IF QVFLAG IS TRUE THEN GOTO 13$
      031232 005737 002440  TST QVFLAG
      031236 001010  BNE 13$
4993 031240 005737 002352  TST  SBEMSK+2 ;LAST BIT IN SINGLE ERROR MASK ?
4994 031244 100405  BMI  13$ ;BR IF YES
4995 031246  DLEFT  SBEMSK
      031246 006137 002350  ROL  SBEMSK
      031252 006137 002352  ROL  SBEMSK+2
      .DSABL CRF
4996 031256 000630  BR  3$
4997 031260 105737 002362 13$:  TSTB  PASFLG ;WHICH PASS
4998 031264 001003  BNE  14$ ;BR IF WE'RE DONE
4999 031266 105237 002362  INCB  PASFLG ;INDICATE SECOND PASS COMING
5000  ;CLEAR OUT ANY DBE'S OR SBE'S
5001 031272 000611  BR  1$ ;GO DO IT!
5002 031274 104471  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5003 031276 013701 002506 14$:  MOV  TESTADD,R1
5004 031302  CLEAR (R1)+,(R1)
      031302 005021  CLR (R1)+
      031304 005011  CLR (R1)
5005 031306 104503  CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
5006 031310 000207  RETURN

```

5010 031312

MTP015: SUBTST <<MTP015 WRITE INHIBIT OF BYTE WITH DBE>>
:*****
:*SUBTEST MTP015 WRITE INHIBIT OF BYTE WITH DBE
:*****

5011

5012

5013 031312 005037 002340
5014 031316 005037 002342
5015 031322 012737 000001 002350
5016 031330 005037 002352
5017 031334 012737 000001 002354
5018 031342 005037 002356
5019 031346 013737 002340 002344
5020 031354 013737 002342 002346
5021 031362 105737 002362
5022 031366 001404
5023 031370 005137 002344
5024 031374 005137 002346
5025 031400 104503
5026 031402 023737 002350 002354
5027 031410 001004
5028 031412 023737 002352 002356
5029 031420 001463
5030 031422 012737 002344 002402
5031 031430 004737 040540
5032 031434 013701 002350
5033 031440 074137 002344
5034 031444 013701 002352
5035 031450 074137 002346
5036 031454 013701 002354
5037 031460 074137 002344
5038 031464 013701 002356
5039 031470 074137 002346
5040 031474 013701 002506
5041 031500 104471
5042 031502 013721 002344
5043
5044 031506 104475
5045 031510 013711 002346
5046 031514 104503
5047 031516 013702 002506
5048 031522 010203
5049 031524 062703 000003
5050 031530 112722 000360
5051 031534 013701 002506
5052 031540 012100
5053 031542 023700 002344
5054 031546 001401
5055 031550 104455
5056
5057 031552 011100
5058 031554 023700 002346
5059 031560 001401
5060 031562 104455
5061
5062 031564 020203
5063 031566 001360

```

1$: CLR DATBUF ;INITIAL DATA
CLR DATBUF+2 ;32 BITS WORTH
2$: MOV #1,SBEMSK ;SINGLE ERROR MASK
CLR SBEMSK+2 ;
3$: MOV #1,DBEMSK ;DOUBLE ERROR MASK
CLR DBEMSK+2 ;
4$: MOV DATBUF,TSTDAT ;PRESERVE ORIG DATA
MOV DATBUF+2,TSTDAT+2
TSTB PASFLG ;WHICH PASS ?
BEQ 5$ ;FIRST PASS, NO COMPLEMENTING
COM TSTDAT
COM TSTDAT+2 ;SECOND PASS, COMPLEMENT TSTDAT
5$: CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
CMP SBEMSK,DBEMSK ;CHECK FOR SAME MASKS
BNE 6$ ;BR IF NOT EQUAL
CMP SBEMSK+2,DBEMSK+2 ;SECOND WORD ALSO
BEQ 11$ ;BR TO MAKE THEM NOT EQUAL
6$: MOV #TSTDAT,SOURCE ;ADDRESS FOR CHKGEN
CALL CHKGEN ;GO GENERATE CHECK BITS
MOV SBEMSK,R1
XOR R1,TSTDAT
MOV SBEMSK+2,R1
XOR R1,TSTDAT+2
MOV DBEMSK,R1
XOR R1,TSTDAT
MOV DBEMSK+2,R1
XOR R1,TSTDAT+2
7$: MOV TESTADD,R1 ;TEST LOCATION
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
MOV TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
;LOAD CSR WITH IMAGE FROM R2
CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
MOV TSTDAT+2,(R1) ;WRITE SECOND 16 BITS + CHECKBITS
CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
MOV TESTADD,R2 ;GET ADDRESS OF TEST LOC
MOV R2,R3 ;R2 DESIGNATES FIRST BYTE
ADD #3,R3 ;R3 DESIGNATES LAST BYTE
8$: MOVB #360,(R2)+ ;TRY WRITING A BYTE
MOV TESTADD,R1
MOV (R1)+,R0
CMP TSTDAT,R0 ;CHECK FOR UNCHANGED DATA
BEQ 9$ ;BR IF OK
PERR31
9$: MOV (R1),R0
CMP TSTDAT+2,R0 ;READ SECOND WORD
BEQ 10$ ;BR IF UNCHANGED
PERR31
10$: CMP R2,R3 ;TESTED LAST BYTE ?
BNE 8$ ;BR IF NO

```

```

5064 031570 005737 002356      11$:  TST    DBEMSK+2      ;CHECKING FOR LAST ERROR BIT
5065 031574 100405              BMI    12$           ;BR IF DONE HERE
5066 031576              DLEFT  DBEMSK
      031576 006137 002354      ROL    DBEMSK
      031602 006137 002356      ROL    DBEMSK+2
      .DSABL CRF
5067 031606 000657              BR     4$
5068 031610              12$:  IF #SW11 SET.IN @SWR THEN GOTO 13$
      031610 032777 004000 151150
      031616 001013
5069 031620              IF QVFLAG IS TRUE THEN GOTO 13$
      031620 005737 002440
      031624 001010
5070 031626 005737 002352      TST    SBEMSK+2      ;LAST SBE MASK
5071 031632 100405              BMI    13$           ;BR IF DONE WITH THIS PASS
5072 031634              DLEFT  SBEMSK
      031634 006137 002350      ROL    SBEMSK
      031640 006137 002352      ROL    SBEMSK+2
      .DSABL CRF
5073 031644 000633              BR     3$
5074 031646 105737 002362      13$:  TSTB   PASFLG ;TEST PASS FLAG
5075 031652 001003              BNE    14$           ;NON ZERO MEANS WE'RE DONE
5076 031654 105237 002362      INCB  PASFLG ;NOT DONR
5077 031660 000614              BR     1$
5078 031662 104471              14$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5079 031664 013701 002506      MOV    TESTADD,R1   ;TEST LOCATION
5080 031670              CLEAR  (R1)+,(R1)   ;TO ERASE ANY DBE'S FROM TESTING
      031670 005021
      031672 005011
5081              ;RESTORE CSR
5082 031674 104503              CLR1CSR
5083 031676 000207              RETURN
                                     ;CLEAR 1 SELECTED MK11 CSR
                                     CLR (R1)+
                                     CLR (R1)
  
```

5087 031700

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

5088
5089
5090
5091

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

5092 031700 005037 002340
5093 031704 005037 002342
5094 031710 012737 000001 002350
5095 031716 005037 002352
5096 031722 012737 000001 002354
5097 031730 005037 002356
5098 031734 013737 002340 002344
5099 031742 013737 002342 002346
5100 031750 105737 002362
5101 031754 001404
5102 031756 005137 002344
5103 031762 005137 002346
5104 031766 023737 002350 002354
5105 031774 001004
5106 031776 023737 002352 002356
5107 032004 001470
5108 032006 012737 002344 002402
5109 032014 004737 040540
5110 032020 013701 002350
5111 032024 074137 002344
5112 032030 013701 002352
5113 032034 074137 002346
5114 032040 013701 002354
5115 032044 074137 002344
5116 032050 013701 002356
5117 032054 074137 002346
5118 032060 013701 002506
5119 032064 104471
5120 032066 013721 002344
5121 032072 104475
5122 032074 013711 002346
5123 032100 105037 002363
5124 032104 162701 000002
5125 032110 104503
5126 032112 012711 177400
5127 032116 013701 002506
5128 032122 011100
5129 032124 023700 002344
5130 032130 001401
5131 032132 104455
5132
5133 032134 062701 000002
5134 032140 011100
5135 032142 023700 002346
5136 032146 001401
5137 032150 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 UPFLG ;SET FOR 2 LOOPS
SUB #2,R1 ;POINT TO LOW WORD
5\$: CLR1CSR ;CLEAR 1 SELECTED MK11 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
PERR31
6\$: ADD #2,R1
MOV (R1),R0
CMP TSTDAT+2,R0 ;THIS SHOULD BE UNCHANGED ALSO
BEQ 7\$
PERR31

```

5140 032152 105737 002363      7$:  TSTB  UPPFLG      ;WHICH LOOP ?
5141 032156 001003              BNE   8$          ;SECOND, BR OUT
5142 032160 105237 002363      INCB  UPPFLG      ;FIRST, KEEP GOING
5143 032164 000751              BR    5$
5144 032166 005737 002356      8$:  TST  DBEMSK+2   ;LAST BIT ?
5145 032172 100405              BMI   9$          ;MINUS = BIT 31
5146 032174              DLEFT DBEMSK
      032174 006137 002354      ROL  DBEMSK
      032200 006137 002356      ROL  DBEMSK+2
      .DSABL CRF
5147 032204 000653              BR    1$
5148 032206              9$:  IF #SW11 SET.IN @SWR THEN GOTO 10$
      032206 032777 004000 150552
      032214 001014
5149 032216              IF QVFLAG IS TRUE THEN GOTO 10$
      032216 005737 002440
      032222 001011
5150 032224 005737 002352      TST  SBEMSK+2    ;LAST BIT IN THIS MASK ?
5151 032230 100406              BMI   10$        ;BR IF LAST BIT
5152 032232              DLEFT SBEMSK
      032232 006137 002350      ROL  SBEMSK
      032236 006137 002352      ROL  SBEMSK+2
      .DSABL CRF
5153 032242 000137 031722      JMP  T12B
5154 032246 105737 002362      10$: TSTB PASFLG ;FIRST PASS ?
5155 032252 001004              BNE  11$         ;BR IF SECOND
5156 032254 105237 002362      INCB PASFLG ;INDICATE SECOND PASS COMING
5157 032260 000137 031700      JMP  T12A
5158 032264 104471              11$: ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
5159 032266 013701 002506      MOV  TESTADD,R1 ;RESTORE TEST ADDRESS
5160 032272 005021              CLR  (R1)+       ;CLEAR ANY DBE'S FROM TEST
5161 032274 005011              CLR  (R1)
5162 032276 104503              CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
5163 032300 000207              RETURN

```

BIT #SW11,@SWR
BNE 10\$
TST QVFLAG
BNE 10\$

5167 032302

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

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

5168
5169
5170
5171
5172
5173
5174 032302 012701 060000
5175 032306 010104
5176 032310 012705 160000
5177 032314 012700 000377
5178 032320 010003
5179 032322 000303
5180 032324 110021
5181 032326 110321
5182 032330 020105
5183 032332 103774
5184
5185 032334 014102
5186 032336 020002
5187
5188 032340 001401
5189 032342 104446
5190
5191 032344 020104
5192 032346 101372
5193 032350 000303
5194 032352 000300
5195 032354 001763
5196
5197 032356 000207

```

:* (1) THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
:* OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
:* OF 000377 AND READING IT
:* (2) MEMORY IS WRITTEN USING A BYTE AT A TIME
:* (3) STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
;NOTE: THIS TEST WRITES BYTES & READS WORDS
MOV #FIRST,R1
MOV R1,R4
MOV #LAST+2,R5
MOV #377,R0 ;GET THE PATTERN INTO R0
MOV R0,R3
SWAB R3
1$: MOVB R0,(R1)+ ;WRITE A BYTE
MOVB R3,(R1)+ ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
CMP R1,R5 ;COMPARE TEST LOC TO TOP + 2
BLO 1$ ;BRANCH IF LOWER

2$: MOV -(R1),R2
CMP R0,R2 ;TEST THE MEMORY TO SEE IF IT CONTAINS
;THE WORD STORED IN BAKPAT
BEQ 3$
PERR2

3$: CMP R1,R4 ;KEEP ON TESTING THE MEMORY UNTIL
BHI 2$ ;R1 EQUALS THE LOWEST ADDRESS
SWAB R3 ;CHANGE THE DATA PATTERN
SWAB R0
BEQ 1$ ;IF THE DATA PATTERN DOES NOT HAVE LOW
;BYTE =0 THEN FALL THRU

RETURN

```

5201 032360
5202
5203
5204
5205
5206 032360 010311
5207 032362 010461 000002
5208 032366 060201
5209 032370 020105
5210 032372 103772
5211 032374 000207
5212
5213
5214 032376 160201
5215 032400 020461 000002
5216 032404 001002
5217 032406 005711
5218 032410 001401
5219 032412 104452
5220 032414 010361 000002
5221 032420 010311
5222 032422 020100
5223 032424 001364
5224 032426 000207
5225
5226
5227 032430 005711
5228 032432 001003
5229 032434 005761 000002
5230 032440 001401
5231 032442 104453
5232 032444 010311
5233 032446 010461 000002
5234 032452 060201
5235 032454 020105
5236 032456 103764
5237 032460 000207
5238
5239
5240 032462 005711
5241 032464 001003
5242 032466 020461 000002
5243 032472 001401
5244 032474 104452
5245 032476 010311
5246 032500 010361 000002
5247 032504 060201
5248 032506 020105
5249 032510 103764
5250 032512 000207

```

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.

:WRITE UP      :100 --> CHECKBITS
MTP20A: MOV      R3,(R1)      :V177640
      MOV      R4,2(R1)     :V177642
      ADD      R2,R1        :V177646
      CMP      R1,R5        :V177650      :TOP + 2 ?
      BLO     MTP20A        :V177652      :BR IF NOT
      RETURN     :V177654

:100 --> 077 DOWN
MTP20B: SUB      R2,R1        :V177640
      CMP      R4,2(R1)     :V177642      :2ND WORD OK?
      BNE     21$          :V177646      :NO - SKIP
      TST     (R1)         :V177650      :1ST WORD OK?
      BEQ     3$           :V177652      :YES - SKIP
21$:  PERR26          :V177654      :GOOD=000000,,100000,,100
3$:  MOV      R3,2(R1)     :V177656
      MOV      R3,(R1)     :V177662
      CMP      R1,R0        :V177664
      BNE     MTP20B        :V177666
      RETURN     :V177670

:077 --> 100 UP
MTP20C: TST      (R1)        :V177640      :1ST WORD OK?
      BNE     41$          :V177642      :NO - SKIP
      TST     2(R1)        :V177644      :2ND WORD OK?
      BEQ     5$           :V177650      :YES - SKIP
41$:  PERR27          :V177652      :GOOD=000000,,000000,,077
5$:  MOV      R3,(R1)     :V177654      :WRITE 1ST WORD
      MOV      R4,2(R1)     :V177656      :WRITE 2ND WORD
      ADD      R2,R1        :V177662
      CMP      R1,R5        :V177664      :TOP + 2 YET?
      BLO     MTP20C        :V177666      :NO - LOOP
      RETURN     :V177670

:100 --> 077 UP
MTP20D: TST      (R1)        :V177640      :1ST WORD OK?
      BNE     61$          :V177642      :NO - SKIP
      CMP      R4,2(R1)     :V177644      :2ND WORD OK?
      BEQ     7$           :V177650      :YES - SKIP
61$:  PERR26          :V177652
7$:  MOV      R3,(R1)     :V177654
      MOV      R3,2(R1)     :V177656
      ADD      R2,R1        :V177662
      CMP      R1,R5        :V177664      :TOP + 2 YET?
      BLO     MTP20D        :V177666      :NO - LOOP
      RETURN     :V177670

```

5253					:077 --> 100 DOWN		
5254	032514	160201		MTP20E:	SUB R2,R1	:V177640	
5255	032516	005761	000002		TST 2(R1)	:V177642	:2ND WORD OK?
5256	032522	001002			BNE 81\$:V177646	:NO - SKIP
5257	032524	005711			TST (R1)	:V177650	:1ST WORD OK?
5258	032526	001401			BEQ 9\$:V177652	:YES - SKIP
5259	032530	104453		81\$:	PERR27	:V177654	
5260	032532	010461	000002	9\$:	MOV R4,2(R1)	:V177656	:WRITE 2ND WORD
5261	032536	010311			MOV R3,(R1)	:V177662	:WRITE 1ST WORD
5262	032540	020100			CMP R1,R0	:V177664	:BOTTOM YET?
5263	032542	001364			BNE MTP20E	:V177666	:NO - LOOP
5264	032544	000207			RETURN	:V177670	
5265							
5266					:100 UP		
5267	032546	005711		MTP20F:	TST (R1)	:V177640	:1ST WORD OK?
5268	032550	001003			BNE 101\$:V177642	:NO - SKIP
5269	032552	020461	000002		CMP R4,2(R1)	:V177642	:2ND WORD OK?
5270	032556	001401			BEQ 11\$:V177646	:YES - SKIP
5271	032560	104452		101\$:	PERR26	:V177650	
5272	032562	060201		11\$:	ADD R2,R1	:V177652	
5273	032564	020105			CMP R1,R5	:V177654	:TOP + 2 YET?
5274	032566	103767			BLO MTP20F	:V177656	:NO - LOOP
5275	032570	000207			RETURN	:V177660	

5279 032572

MTPA20: SUBTST <<MTPA20 SLOW MARCHING 1'S & 0'S IN CHECK BITS TEST>>

 :*SUBTEST MTPA20 SLOW MARCHING 1'S & 0'S IN CHECK BITS TEST

5280
 5281
 5282 032572 012737 177777 002022
 5283 032600 013737 002144 002150
 5284 032606 013737 002146 002152
 5285 032614 013701 002336
 5286 032620 013702 002414
 5287 032624 010103
 5288 032626 005004
 5289 032630 012705 160000
 5290 032634 005037 002144
 5291 032640 104505
 5292
 5293
 5294 032642 010411
 5295 032644 060201
 5296 032646 020105
 5297 032650 103774
 5298
 5299
 5300
 5301 032652 160201
 5302 032654 005711
 5303 032656 104426
 5304 032660 032737 077400 002144
 5305 032666 001406
 5306 032670 013700 002144
 5307 032674 000300
 5308 032676 042700 000200
 5309 032702 104435
 5310
 5311 032704 012737 077400 002144
 5312 032712 104505
 5313 032714 005104
 5314 032716 010411
 5315 032720 020103
 5316 032722 001353
 5317
 5318
 5319 032724 005037 002144
 5320 032730 104505

```

    ;*THIS TEST IS CONCERNED ONLY WITH THE INTEGRITY
    ;*OF THE MOS RAMS THAT STORE THE CHECKBITS.
    SET SBFLAG
    MOV #-1,SBFLAG
    MOV CSR,SBECR
    MOV CSR+2,SBECR+2
    MOV CSRFIRST,R1 ;FIRST TEST LOC
    MOV CSRINC,R2
    MOV R1,R3
    CLR R4
    MOV #LAST+2,R5
    CLR CSR
    CHK1DIS ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
    *****
    2$: MOV R4,(R1) ;WRITE CHECKBITS BY WRITING LOC * WRITE UP
    ADD R2,R1 ;POINT TO NEXT WORD * 0'S --> CHK BITS
    CMP R1,R5 ;TOP + 2
    BLO 2$ ;BR IF NOT
    *****
    3$: SUB R2,R1 ;ADJUST R1
    TST (R1) ;READ
    READCSR
    BIT #77400,@#CSR ;SHOULD BE ALL ZEROS
    BEQ 4$ ;BR IF OK
    MOV CSR,R0
    SWAB R0
    BIC #200,R0
    PERR11
    *****
    4$: MOV #77400,@#CSR ;MAKE CHECKBITS ALL 1'S *0'S --> 1'S
    CHK1DIS ;DSABL ECC & WRITE CHKBITS FROM 1 CSR * DOWN
    COM R4 ;COMP THE DATA
    MOV R4,(R1) ;WRITE THE CHECKBITS
    CMP R1,R3 ;BOTTOM YET?
    BNE 3$ ;BR IF NO
    *****
    CLR CSR
    CHK1DIS ;DISABLE ECC & WRITE CHKBITS FROM 1 SELECTED CSR
    
```

```

5323
5324 032732 005711          5$:  TST      (R1)          :READ THE LOC AGAIN
5325 032734 104426          READCSR
5326 032736 013700 002144  MOV     @#CSR,R0      :GET CSR CONTENTS INTO R0
5327 032742 042700 100377  BIC     #100377,R0    :ONLY WANT BITS 11-5
5328 032746 022700 077400  CMP     #77400,R0    :READ FOR ALL 1'S
5329 032752 001404          BEQ     6$           :BR IF OK
5330 032754 000300          SWAB   R0
5331 032756 042700 000200  BIC     #200,R0
5332 032762 104445          PERR21
5333
5334 032764 005037 002144  6$:  CLR     CSR          :DSABL ECC & WRITE CHKBITS FROM 1 CSR
5335 032770 104505          CHK1DIS
5336 032772 005104          COM    R4
5337 032774 010411          MOV    R4,(R1)       :ACCESS THE LOC
5338 032776 060201          ADD    R2,R1         :POINT TO NEXT ADDRESS
5339 033000 020105          CMP    R1,R5         :TOP+2 YET?
5340 033002 103753          BLO   5$           :BR IF NOT
5341
5342
5343 033004 010301          MOV    R3,R1
5344
5345
5346 033006 005711          7$:  TST      (R1)          :READ
5347 033010 104426          READCSR
5348 033012 032737 077400 002144  BIT     #77400,@#CSR  :CHECKBITS SHOULD BE ZERO
5349 033020 001406          BEQ     8$           :BR IF OK
5350 033022 013700 002144  MOV    CSR,R0
5351 033026 000300          SWAB   R0
5352 033030 042700 000200  BIC     #200,R0
5353 033034 104435          PERR11
5354
5355 033036 012737 077400 002144  8$:  MOV    #77400,@#CSR  :COMP CHECKBITS
5356 033044 104505          CHK1DIS             :DSABL ECC & WRITE CHKBITS FROM 1 CSR
5357 033046 005104          COM    R4
5358 033050 010411          MOV    R4,(R1)       :WRITE THE LOC
5359 033052 060201          ADD    R2,R1         :POINT TO NEXT ADDRESS
5360 033054 020105          CMP    R1,R5         :TOP+2 YET?
5361 033056 103753          BLO   7$           :BR IF NOT
5362
5363
5364 033060 005037 002144  CLR     CSR
5365 033064 104505          CHK1DIS             :DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
    
```

*1'S --> 0'S
 *UP

*0'S --> 1'S
 *UP

```

5368
5369 033066 160201          9$:  SUB   R2,R1          :ADJUST R1
5370 033070 005711          TST   (R1)           :READ
5371 033072 104426          READCSR
5372 033074 013700 002144  MOV   @#CSR,R0       :GET CSR CONTENTS
5373 033100 042700 100377  BIC   #100377,R0     :ONLY WANT CHECKBITS
5374 033104 022700 077400  CMP   #77400,R0     :ALL 1'S?
5375 033110 001404          BEQ   10$           :BR IF OK
5376 033112 000300          SWAB  R0
5377 033114 042700 000200  BIC   #200,R0
5378 033120 104445          PERR21
5379
5380 033122 005037 002144  10$: CLR   CSR
5381 033126 104505          CHK1DIS           :DSABL ECC & WRITE CHKBTS FROM 1 CSR
5382 033130 005104          COM   R4          :COMP THE DATA
5383 033132 010411          MOV   R4,(R1)     :WRITE THE CHECKBITS
5384 033134 020103          CMP   R1,R3       :BOTTOM YET?
5385 033136 001353          BNE   9$          :BR IF NO
5386
5387
5388
5389 033140 005711          11$: TST   (R1)           :READ
5390 033142 104426          READCSR
5391 033144 032737 077400 002144  BIT   #77400,@#CSR  :SHOULD BE ALL 0'S
5392 033152 001406          BEQ   12$           :BR IF OK
5393 033154 013700 002144  MOV   CSR,R0
5394 033160 000300          SWAB  R0
5395 033162 042700 000200  BIC   #200,R0
5396 033166 104435          PERR11
5397
5398 033170 060201          12$: ADD   R2,R1          :POINT TO NEXT WORD
5399 033172 005037 002144  CLR   CSR
5400 033176 104505          CHK1DIS           :DSABL ECC & WRITE CHKBTS FROM 1 CSR
5401 033200 020105          CMP   R1,R5       :TOP + 2 YET?
5402 033202 103756          BLO   11$         :BR IF NOT
5403
5404
5405
5406 033204          :FLUSH OUT ALL THESE DBE'S
      033204 004537 043710  BMOV  MTP030
      033210 034704          JSR  R5,BLOCK1
      .DSABL CRF
5407 033212 012700 060000  MOV   #FIRST,R0
5408 033216 012701 040000  MOV   #SIZE,R1
5409 033222 104471          ECC1DIS           :DISABLE ECC ON 1 SELECTED CSR
5410 033224 004737 177640  CALL  FASTCITY     :CALL TO THE USER INSTRUCTION PAR'S
5411 033230 104503          CLR1CSR           :CLEAR 1 SELECTED MK11 CSR
5412 033232 000207          RETURN

```

*1'S --> 0'S
*DOWN

*0'S
*UP

5416 033234

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

:SUBTEST MTPA21 MARCHING 1'S & 0'S PATTERN TEST

5417
5418 033234 014100
5419 033236 020200
5420 033240 001401
5421 033242 104443

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

5422
5423 033244 000311
5424 033246 011100
5425 033250 020300
5426 033252 001401
5427 033254 104444

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

5428
5429 033256 020401
5430 033260 001365
5431 033262 000207

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

5432
5433 033264
5434 033264 011100
5435 033266 020300
5436 033270 001401
5437 033272 104444

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

5438
5439 033274 000311
5440 033276 011100
5441 033300 020200
5442 033302 001401
5443 033304 104443

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

5444
5445 033306 062701 000002
5446 033312 020501
5447 033314 001363
5448 033316 000207

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

5449
5450 033320
5451 033320 011100
5452 033322 020200
5453 033324 001401
5454 033326 104443

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

5455
5456 033330 000311
5457 033332 011100
5458 033334 020300
5459 033336 001401
5460 033340 104444

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

5461
5462 033342 062701 000002
5463 033346 020501
5464 033350 001363
5465 033352 000207

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

```

5468 033354          MTPD21: ;READ,BYTESWAP-MODIFY,READ,DOWN
5469 033354 014100 1$:    MOV    -(R1),R0 ;V177640
5470 033356 020300      CMP    R3,R0    ;V177642
5471 033360 001401      BEQ    2$      ;V177644
5472 033362 104444      PERR20   ;V177646
5473
5474 033364 000311 2$:    SWAB   (R1)    ;V177650
5475 033366 011100      MOV    (R1),R0 ;V177652
5476 033370 020200      CMP    R2,R0    ;V177654
5477 033372 001401      BEQ    3$      ;V177656
5478 033374 104443      PERR17   ;V177660
5479
5480 033376 020401 3$:    CMP    R4,R1    ;V177662      ;DONE?
5481 033400 001365      BNE    1$      ;V177664      ;NO - LOOP
5482 033402 000207      RETURN   ;V177666      ;YES - RETURN
5483

```

```

5487 033404 MTP022: SUBTST <<MTP022 REFRESH & SHIFTING DIAGONAL TEST>>
:*****
:*SUBTEST MTP022 REFRESH & SHIFTING DIAGONAL TEST
:*****
5488 : (1) WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
5489 : (2) IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
5490 : (3) WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
5491 : (WITH CACHE OFF).
5492 : ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
5493 033404 000010 KDIAG=8. ;FOR DATA & COMPLEMENT DATA
033404 012737 000001 002456 MOV #1,EVEN
033412 B43:*****
5494 033412 IF EVEN EQ #1
033412 023727 002456 000001 CMP EVEN,#1
033420 001005 BNE L367
5495 033422 LET R2 := ZEROS
033422 013702 002430 MOV ZEROS,R2
5496 033426 LET R3 := ONES
033426 013703 002726 MOV ONES,R3
5497 033432 ELSE
033432 000404 BR L370
033434 L367:*****
5498 033434 LET R2 := ONES
033434 013702 002726 MOV ONES,R2
5499 033440 LET R3 := ZEROS
033440 013703 002430 MOV ZEROS,R3
5500 033444 END ;OF IF EVEN
5501 033444 FOR STRIPES := #0 TO #KDIAG-1 ;FOR THE NUMBER OF STRIPES
033444 005037 002460 CLR STRIPES
033450 B44:*****
5502
5503 ;WRITE LOOP
5504 033450 104423 CACHON ;TURN CACHE ON
5505 033452 LET COUNT := STRIPES
033452 013737 002460 002462 MOV STRIPES,COUNT
5506 033460 LET R1 := #FIRST
033460 012701 060000 MOV #FIRST,R1
5507 033464 WHILE R1 LOS #LAST
033464 B45:*****
033464 020127 157776 CMP R1,#LAST
033470 101032 BHI L371
5508 033472 IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
033472 005737 002462 TST COUNT
033476 002003 BGE L372
033500 012737 000007 002462 MOV #KDIAG-1,COUNT
033506 L372:*****
5509 033506 IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
033506 032701 000374 BIT #374,R1
033512 001002 BNE L373
033514 005337 002462 DEC COUNT
033520 L373:*****
5510 033520 IF COUNT NE #0
033520 005737 002462 TST COUNT
033524 001404 BEQ L374
5511 033526 LET (R1) := R2
033526 010211 MOV R2,(R1)

```

```
5512 033530          LET 2(R1) := R2
      033530 010261 000002
5513 033534          ELSE
      033534 000403
      033536
5514 033536          LET (R1) := R3
      033536 010311
5515 033540          LET 2(R1) := R3
      033540 010361 000002
5516 033544          END ;OF IF COUNT
      033544
5517 033544          LET COUNT := COUNT - #1
      033544 005337 002462
5518 033550          LET R1 := R1 + #4
      033550 062701 000004
5519 033554          END ;OF WHILE
      033554 000743
      033556
5520          ;END OF WRITE LOOP
5521
5522 033556          IF DIAGFLAG IS FALSE THEN $CALL REFRESH
      033556 005737 002006
      033562 001002
      033564 004737 033760
      033570
```

```

      MOV R2,2(R1)
      BR L375
L374:
      MOV R3,(R1)
      MOV R3,2(R1)
L375:
      DEC COUNT
      ADD #4,R1
      BR B45
L371:
E45:
      TST DIAGFLAG
      BNE L376
      JSR PC,REFRESH
L376:
```

5525					:READ LOOP	
5526	033570				LET COUNT := STRIPES	
	033570	013737	002460	002462		MOV STRIPES,COUNT
5527	033576				LET R1 := #FIRST	
	033576	012701	060000			MOV #FIRST,R1
5528	033602	104424			CACHOFF	
5529	033604				WHILE R1 LOS #LAST	:TURN CACHE OFF
	033604					B46:.....
	033604	020127	157776			CMP R1,#LAST
	033610	101046				BHI L377
5530	033612				IF COUNT LT #0 THEN LET COUNT := #KDIAG-1	
	033612	005737	002462			TST COUNT
	033616	002003				BGE L400
	033620	012737	000007	002462		MOV #KDIAG-1,COUNT
	033626					L400:.....
5531	033626				IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1	
	033626	032701	000374			BIT #374,R1
	033632	001002				BNE L401
	033634	005337	002462			DEC COUNT
	033640					L401:.....
5532	033640				IF COUNT NE #0	
	033640	005737	002462			TST COUNT
	033644	001412				BEQ L402
5533	033646				LET R0 := (R1)	
	033646	011100				MOV (R1),R0
5534	033650				IF R2 NE R0	
	033650	020200				CMP R2,R0
	033652	001401				BEQ L403
5535	033654	104443			PERR17	
5536	033656				END ;OF IF R2	
	033656					L403:.....
5537	033656				LET R0 := 2(R1)	
	033656	016100	000002			MOV 2(R1),R0
5538	033662				IF R2 NE R0	
	033662	020200				CMP R2,R0
	033664	001401				BEQ L404
5539	033666	104443			PERR17	
5540	033670				END ;OF IF R2	
	033670					L404:.....
5541	033670				ELSE	
	033670	000411				BR L405
	033672					L402:.....
5542	033672				LET R0 := (R1)	
	033672	011100				MOV (R1),R0
5543	033674				IF R3 NE R0	
	033674	020300				CMP R3,R0
	033676	001401				BEQ L406
5544	033700	104444			PERR20	
5545	033702				END ;OF IF R3	
	033702					L406:.....
5546	033702				LET R0 := 2(R1)	
	033702	016100	000002			MOV 2(R1),R0
5547	033706				IF R3 NE R0	
	033706	020300				CMP R3,R0
	033710	001401				BEQ L407
5548	033712	104444			PERR20	
5549	033714				END ;OF IF R3	

```

5550 033714          END ;OF IF COUNT          L407:.....
5551 033714          LET COUNT := COUNT - #1    L405:.....
5552 033720 005337 002462          DEC COUNT
5553 033720 062701 000004          LET R1 := R1 + #4          ADD #4,R1
5554 033724          END ;OF WHILE
5555 033724 000727          ;END OF READ LOOP          BR B46
5556 033726          END ;OF FOR STRIPES
5557 033732 005237 002460 000007          INC STRIPES
5558 033740 023727 002460          CMP STRIPES,#KDIAG-1
5559 033742 003643          BLE B44          E44:.....
5560 033742          END ;OF FOR EVEN
5561 033742 005237 002456 000002          INC EVEN
5562 033746 023727 002456          CMP EVEN,#2
5563 033754 003616          BLE B43          E43:.....
5564 033756 000207          RETURN
5565 033760          REFRESH:SUBTST <<SUBR REFRESH DELAY>>
:*****
:*SUBTEST          SUBR          REFRESH DELAY
:*****
5566 033760          ;DISTURB EACH ROW FOR > 3.2 MS
5567 033760 012700 060000          FOR R0 := #FIRST TO #FIRST+374 BY #4
5568 033764          CALL REFSUB
5569 033770          END ;OF FOR R0
5570 033770 062700 000004          MOV #FIRST,R0
5571 033774 020027 060374          B47:.....
5572 034000 003771          ADD #4,R0
5573 034002          CMP R0,#FIRST+374
5574 034002          BLE B47          E47:.....
5575 034002          LET R0 := #FIRST+BIT14
5576 034006          WHILE R0 LOS #LAST+BIT14+374
5577 034006 020027 020372          MOV #FIRST+BIT14,R0
5578 034012 101005          B50:.....
5579 034014 004737 034030          CMP R0,#LAST+BIT14+374
5580 034020          CALL REFSUB          BHI L410
5581 034020 062700 000004          LET R0 := R0 + #4
5582 034024          END ;OF WHILE          ADD #4,R0
5583 034024 000770          ;TIME FOR A > 3.2 MS LOOP          BR B50
5584 034026          L410:.....
5585 034026          E50:.....
5586 034026 000207          RETURN
5587 034030 012704 000640          REFSUB: MOV #640,R4
5588 034034 062700 000002          ADD #2,R0
5589 034040 005140          1$: COM -(R0)
5590 034042 005120          COM (R0)+
5591 034044 005110          COM (R0)

```

5575 034046 005110
5576 034050 077405
5577 034052 162700 000002
5578 034056 000207

COM (R0)
SOB R4.1\$
SUB #2.R0
RETURN

5582 034060

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

5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5620
5621
5622
5623

034060 011100
034062 020004
034064 001401
034066 104447

034070 011200
034072 020003
034074 001401
034076 104450

034100 062702 000400
034104 020205
034106 101764

034110 062701 000002
034114 000137 172260

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

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

```

:UIPAR'S
1$: MOV (R1),R0 :V177640 :READ 'A'
    CMP R0,R4 :V177642 :CHECK 'A'
    BEQ 2$ :V177644 :BR IF OK
    PERR23 :V177646 :REPORT ERROR

2$: MOV (R2),R0 :V177650 :READ 'B'
    CMP R0,R3 :V177652 :CHECK 'B'
    BEQ 3$ :V177654 :BR IF OK
    PERR24 :V177656 :REPORT ERROR

3$: ADD #400,R2 :V177660 :BUMP 'B'
    CMP R2,R5 :V177664 :AT END YET?
    BLOS 1$ :V177666 :BR IF NO

    ADD #2,R1 :V177670 :BUMP 'A'
    JMP @#SDPAR0 :V177674 :GOTO V177260
    
```

5626 034120

MTPB24: SUBTST <<MTPB24 FAST GALLOP PART B>>

: *SUBTEST MTPB24 FAST GALLOP PART B

5627

:SDPAR'S

5628 034120 010411
5629 034122 020105
5630 034124 001001
5631 034126 000207
5632 034130 000137 172360
5633
5634 034134

MOV R4,(R1) :V172260 :WRITE 'A'
CMP R1,R5 :V172262 :DONE?
BNE 1\$:V172264 :BR IF NO
RETURN :V172266 :YES - RETURN
1\$: JMP @#KDPAR0 :V172270 :GOTO V172360

MTPC24: SUBTST <<MTPC24 FAST GALLOP PART C>>

: *SUBTEST MTPC24 FAST GALLOP PART C

5635

:KDPAR'S

5636 034134 010102
5637 034136 011100
5638 034140 020004
5639 034142 001401
5640 034144 104447
5641 034146 000137 177660

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 @#UDPAR0 :V172372 :GOTO V177660

5645 034152

MTP025: SUBTST <<MTP025 INTERRUPT ENABLE TEST>>
:*****
: *SUBTEST MTP025 INTERRUPT ENABLE TEST
:*****

5646	034152	005037	002344		CLR	TSTDAT	:GENERATE CHECKBITS ON 0,,0
5647	034156	005037	002346		CLR	TSTDAT+2	
5648	034162	012737	002344	002402	MOV	#TSTDAT,SOURCE	
5649	034170	004737	040540		CALL	CHKGEN	
5650	034174	012737	000003	002100	MOV	#3,NOPAR	:SETUP PARITY ACTION
5651	034202	013701	002506		MOV	TESTADD,R1	:FIRST TEST ADDRESS
5652	034206	012737	034242	002372	MOV	#1\$,PARTHERE	:SETUP TRAP DESTINATION
5653	034214	004737	034464		CALL	MTPA25	:WRITE DATA & CHECKBITS
5654	034220	104473			ECC1INIT		:INITIALIZE 1 SELECTED MK11 CSR
5655	034222	005711			TST	(R1)	:ACCESS LOCATIONS FOR DBE TRAPS
5656	034224	005761	000002		TST	2(R1)	
5657							:NONE - GOOD - ACCESS FOR SBE TRAPS
5658	034230	104507			ENA1SBE		:DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
5659	034232	005711			TST	(R1)	
5660	034234	005761	000002		TST	2(R1)	
5661	034240	000404			BR	2\$:NONE - GOOD - SKIP
5662	034242	104426			1\$: READCSR		
5663	034244				FATAL	27	
	034244	005237	002070		INC	FATAL\$:SET FATAL INDICATOR
	034250	104027			ERROR	+27	
					.DSABL	CRF	
5664	034252	005237	002344		2\$: INC	TSTDAT	:CHECK FOR CORRECT ACTION ON SBE'S
5665	034256	004737	034412		CALL	MTPD25	:IN ALL 4 BYTES
5666	034262	012737	000400	002344	MOV	#400,TSTDAT	
5667	034270	004737	034412		CALL	MTPD25	
5668	034274	005037	002344		CLR	TSTDAT	
5669	034300	005237	002346		INC	TSTDAT+2	
5670	034304	004737	034412		CALL	MTPD25	
5671	034310	012737	000400	002346	MOV	#400,TSTDAT+2	
5672	034316	004737	034412		CALL	MTPD25	
5673							
5674	034322	005037	002346		CLR	TSTDAT+2	:CHECK FOR CORRECT ACTION ON DBE'S
5675	034326	012737	000003	002344	MOV	#3,TSTDAT	:IN ALL 4 BYTES
5676	034334	004737	034434		CALL	MTPE25	
5677	034340	012737	001400	002344	MOV	#1400,TSTDAT	
5678	034346	004737	034434		CALL	MTPE25	
5679	034352	005037	002344		CLR	TSTDAT	
5680	034356	012737	000003	002346	MOV	#3,TSTDAT+2	
5681	034364	004737	034434		CALL	MTPE25	
5682	034370	012737	001400	002346	MOV	#1400,TSTDAT+2	
5683	034376	004737	034434		CALL	MTPE25	
5684	034402	104503			CLR1CSR		:CLEAR 1 SELECTED MK11 CSR
5685	034404	005037	002100		CLR	NOPAR	:INDICATE PARITY ACTION
5686	034410	000207			RETURN		
5687							
5688	034412	004737	034464		MTPD25: CALL	MTPA25	:WRITE DATA & CHECKBITS
5689	034416	104471			ECC1DIS		:DISABLE ECC ON 1 SELECTED CSR
5690	034420	004737	034504		CALL	MTPB25	:CHECK FOR NO TRAPS
5691	034424	104507			ENA1SBE		:DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
5692	034426	004737	034536		CALL	MTPC25	:CHECK FOR EXPECTED TRAP
5693	034432	000207			RETURN		

```

5696 034434 004737 034464      MTP25: CALL      MTPA25      ;WRITE DATA & CHECKBITS
5697 034440 104471              ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
5698 034442 004737 034504      CALL      MTPB25      ;CHECK FOR NO TRAPS
5699              ;ENABLE DBE TRAPS
5700 034446 104473              ECC1INIT      ;INITIALIZE 1 SELECTED MK11 CSR
5701 034450 004737 034536      CALL      MTPC25      ;CHECK FOR EXPECTED TRAP
5702 034454 104507              ENA1SBE      ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
5703 034456 004737 034536      CALL      MTPC25      ;CHECK FOR EXPECTED TRAP
5704 034462 000207              RETURN
5705
5706              ;WRITE TSTDAT & TSTDAT+2 & CHECKBITS
5707 034464 104471      MTPA25: ECC1DIS      ;DISABLE ECC ON 1 SELECTED CSR
5708 034466 013711 002344      MOV      TSTDAT,(R1) ;WRITE FIRST 16 BITS
5709 034472 104475      CB1CSR      ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5710 034474 013761 002346 000002      MOV      TSTDAT+2,(R1) ;WRITE 2ND 16 BITS & CHECKBITS
5711 034502 000207      RETURN
5712
5713              ;CHECK FOR NO TRAP OCCURING CONDITION
5714 034504 012737 034522 002372      MTPB25: MOV      #1$,PARTHERE ;SETUP TRAP DESTINATION
5715 034512 005711      TST      (R1)      ;ACCESS LOCATIONS
5716 034514 005761 000002      TST      2(R1)
5717 034520 000207      RETURN      ;NO TRAP - GOOD - RETURN
5718
5719 034522 104426      1$:      READCSR
5720 034524 104024      ERROR      +24
5721 034526      SET      HEADER
5722 034526 012737 177777 002724      RETURN      MOV #-1,HEADER
5723 034534 000207
5724
5725 034536 012737 034550 002372      MTPC25: ;TRAP SHOULD OCCURE TEST
5726 034544 005711      MOV      #1$,PARTHERE ;SETUP TRAP DESTINATION
5727 034546 000405      TST      (R1)      ;ACCESS 1ST LOCATION
5728 034550 012737 034574 002372      BR      2$      ;NO TRAP - BAD NEWS - SKIP
5729 034556 005761 000002      1$:      MOV      #3$,PARTHERE ;SETUP TRAP DESTINATION
5730 034562 104426      TST      2(R1)      ;ACCESS 2ND LOCATION
5731 034564 104025      2$:      READCSR      ;NO TRAP - BAD NEWS
5732 034566      ERROR      +25
5733 034574 012737 177777 002724      SET      HEADER
5734 034574 000207      3$:      RETURN      MOV #-1,HEADER

```

5738 034576
 5739 034576 000137 172360
 5740 034602 010221
 5741 034604 010321
 5742 034606 077005
 5743 034610 000207
 5744
 5745 034612

```
MTPA26: SUBTST <<MTPA26      RANDOM DATA (WRITE)>>
:*****
:*SUBTEST      MTPA26  RANDOM DATA (WRITE)
:*****
1$:      JMP      KDPAR0      ;V177640      GOTO V172360
        MOV      R2,(R1)+    ;V177644
        MOV      R3,(R1)+    ;V177646
        SOB      R0,1$      ;V177650
        RETURN    ;V177652
```

5746
 5747
 5748 034612 000137 172360
 5749 034616 020221
 5750 034620 001401
 5751 034622 104451
 5752 034624 005127
 5753 034626 000000
 5754 034630 020321
 5755 034632 001401
 5756 034634 104451
 5757 034636 005167 177764
 5758 034642 077015
 5759 034644 000207
 5760
 5761
 5762
 5763 034646

```
MTPB26: SUBTST <<MTPB26      RANDOM DATA (READ)>>
:*****
:*SUBTEST      MTPB26  RANDOM DATA (READ)
:*****
        .DSABL  AMA
        .ENABL  LSB
1$:      JMP      @#KDPAR0    ;V177640      GOTO V172360
        CMP      R2,(R1)+    ;V177644
        BEQ      2$          ;V177646
        PERR25  ;V177650
2$:      COM      (PC)+      ;V177652
        RANODD: 0          ;V177654      FOR ERROR REPORTING
        CMP      R3,(R1)+    ;V177656
        BEQ      3$          ;V177660
        PERR25  ;V177662
3$:      COM      RANODD     ;V177664
        SOB      R0,1$      ;V177670
        RETURN    ;V177672
        .DSABL  LSB
        .ENABL  AMA
```

5764
 5765
 5766
 5767
 5768
 5769 034646 073427 000007
 5770 034652 060305
 5771 034654 005504
 5772 034656 060204
 5773 034660 062705 001057
 5774 034664 000730
 5775
 5776 034666

```
MTPC26: SUBTST <<RANDOM NUMBER SUBPROGRAM>>
:*****
:*SUBTEST      RANDOM NUMBER SUBPROGRAM
:*****
        :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
        BR     -.116      ;V172376      GOTO V172260
```

5777 034666 005504
 5778 034670 062704 047401
 5779 034674 010503
 5780 034676 010402
 5781 034700 000137 177644

```
MTPD26: SUBTST <<RANDOM NUMBER SUBSUBPROGRAM>>
:*****
:*SUBTEST      RANDOM NUMBER SUBSUBPROGRAM
:*****
        ADC     R4         ;V172260
        ADD     #47401,R4  ;V172262
        MOV     R5,R3      ;V172266
        MOV     R4,R2      ;V172270
        JMP     UIPAR2    ;V172272      GOTO V177644
```

5784 034704

```

MTP030: 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
    
```

5785 034704 011002
 5786 034706 010220
 5787 034710 077103
 5788 034712 000207
 5789
 5790 034714

```

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

5791
 5792 034714 000000
 5793 034716 077001
 5794 034720 005167 177772
 5795 034724 020167 177766
 5796 034730 001403
 5797 034732 104454
 5798 034734 010167 177756
 5799 034740 005167 177752
 5800 034744 010200
 5801
 5802 034746 010503
 5803 034750 005725
 5804 034752 010504
 5805 034754 020527 160000
 5806 034760 001001
 5807 034762 000207
 5808
 5809 034764 014344
 5810 034766 001376
 5811 034770 000752
 5812 000056
 5813

```

      .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,#LAST+2  ;DONE?
      BNE      3$          ;NO - SKIP
      RETURN                     ;YES
3$:  MOV      -(R3),-(R4)
      BNE      3$
      BR      1$
SOBLENGTH=.-MTP031
      .ENABL  AMA
    
```

5816 034772

```

MTPA32: SUBTST <<MTPA32      SKEWED UP ADDRESS TEST (WRITE)>>
:*****
:*SUBTEST      MTPA32 SKEWED UP ADDRESS TEST (WRITE)
:*****
1$:  ADD      #6,R4          ;V177640      ;ADD OFFSET TO OFFSET COUNTER
    BIC      #^C6,R4        ;V177644      ;MASK TO (0,2,4,6)
    MOV      R3,R1          ;V177650      ;PUT IN LSB'S OF ADDRESS
    ADD      R4,R1          ;V177652      ;PUT IN MSB'S OF ADDRESS
    MOV      R0,(R1)        ;V177654      ;WRITE DATA IN R0 INTO MEMORY
    ADD      #10,R3         ;V177656      ;BUMP ADDRESS MSB TO NEXT BLOCK OF 4 WORDS
    SOB      R5,1$         ;V177662      ;LOOP TILL DONE
    RETURN                      ;V177664
  
```

5817 034772 062704 000006
 5818 034776 042704 177771
 5819 035002 010301
 5820 035004 060401
 5821 035006 010011
 5822 035010 062703 000010
 5823 035014 077512
 5824 035016 000207
 5825
 5826 035020

```

MTPB32: SUBTST <<MTPB32      SKEWED UP ADDRESS TEST (READ)>>
:*****
:*SUBTEST      MTPB32 SKEWED UP ADDRESS TEST (READ)
:*****
1$:  ADD      #6,R4          ;V177640      ;ADD OFFSET TO OFFSET COUNTER
    BIC      #^C6,R4        ;V177644      ;MASK TO (0,2,4,6)
    MOV      R3,R1          ;V177650      ;PUT IN LSB'S OF ADDRESS
    ADD      R4,R1          ;V177652      ;PUT IN MSB'S OF ADDRESS
    MOV      (R1),R2        ;V177654      ;READ DATA INTO R0
    CMP      R2,R0          ;V177656      ;IS IT CORRECT?
    BEQ      2$             ;V177660      ;YES - SKIP
    PERR22                      ;V177662      ;NO - PRINT ERROR
2$:  ADD      #10,R3         ;V177664      ;BUMP ADDRESS MSB TO NEXT BLOCK OF 4 WORDS
    SOB      R5,1$         ;V177666      ;LOOP TILL DONE
    RETURN                      ;V177670
  
```

5827 035020 062704 000006
 5828 035024 042704 177771
 5829 035030 010301
 5830 035032 060401
 5831 035034 011102
 5832 035036 020200
 5833 035040 001401
 5834 035042 104446
 5835 035044 062703 000010
 5836 035050 077515
 5837 035052 000207
 5838
 5839 035054

```

MTP033: SUBTST <<MTP033      WRITE RECOVERY TEST>>
:*****
:*SUBTEST      MTP033 WRITE RECOVERY TEST
:*****
: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.
  
```

5840
 5841
 5842
 5843
 5844
 5845 035054 012401
 5846 035056 020102
 5847 035060 001401
 5848 035062 104430
 5849 035064 077305
 5850 035066 012703 020000
 5851 035072 012400
 5852 035074 020005
 5853 035076 001401
 5854 035100 104427
 5855 035102 077305
 5856 035104 000207

```

1$:  MOV      (R4)+,R1       ;V177640      ;GET DATA FROM LOWER 1/2 BANK
    CMP      R1,R2          ;V177642      ;IS IT #5141?
    BEQ      2$             ;V177644      ;YES - SKIP
    PERR02                      ;V177646      ;NO - TAKE ERROR TRAP
2$:  SOB      R3,1$         ;V177650      ;LOOP FOR 1/2 BANK
    MOV      #SIZE/2,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
    PERR01                      ;V177664      ;NO- TAKE ERROR TRAP
4$:  SOB      R3,3$         ;V177666      ;LOOP FOR 1/2 BANK
    RETURN
  
```

5859 035106

MTP034: SUBTST <<MTP034 BRANCH GOBBLE TEST>>
 :*****
 :*SUBTEST MTP034 BRANCH GOBBLE TEST
 :*****

5860
 5861 035106 000000
 5862 035110 000000
 5863 035112 000261
 5864 035114 105511
 5865 035116 100402
 5866 035120 105212
 5867 035122 000773
 5868
 5869
 5870 035124 102401
 5871 035126 104461
 5872
 5873 035130 000242
 5874 035132 105212
 5875 035134 103402
 5876 035136 102001
 5877 035140 100401
 5878 035142 104461
 5879
 5880
 5881
 5882 035144 010701
 5883 035146 162701 000036
 5884 035152 010102
 5885 035154 005202
 5886
 5887
 5888 035156 010503
 5889 035160 005725
 5890 035162 010504
 5891
 5892
 5893 035164 020527 160000
 5894 035170 001001
 5895 035172 000207
 5896
 5897
 5898 035174 014344
 5899 035176 001376
 5900 035200 005011
 5901 035202 000743
 5902 000076
 5903

```

      .DSABL  AMA
      0
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$:      CLV
        INCB      (R2) ;CLEAR V-BIT
        BCS       3$   ;INCREMENT HIGH BYTE OF 'BGTEST' ONCE MORE
        BVC       3$   ;BR IF C-BIT SET (SHOULD NOT BE)
        BMI       4$   ;BR IF V-BIT CLEAR (SHOULD NOT BE)
        PERR35    ;BR IF N-BIT SET (SHOULD BE)
                          ;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 REGISTERS
        MOV        R5,R3
        TST        (R5)+ ;BUMP (SAFELY) BY 2
        MOV        R5,R4

        ;DONE?
        CMP        R5,#LAST+2 ;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=-MTP034
        .ENABL  AMA
  
```

```

5905 .SBTTL MISC SUBROUTINES
5906
5907 035204 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
:*****
5908 035204 010004 MOV R0,R4
5909 035206 010103 MOV R1,R3
5910 035210 010205 MOV R2,R5
5911 035212 000207 RETURN
5912
5913 035214 FLIPWARN:SUBTST <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
:*****
:*SUBTEST FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
:*****
5914 035214 PUSH R0
035214 010046 MOV R0,-(SP)
5915 035216 005237 002734 INC FLIPLOC
5916 035222 042737 177774 002734 BIC #^C3,FLIPLOC
5917 035230 022737 000001 002734 CMP #1,FLIPLOC
5918 035236 001414 BEQ 1$
5919 035240 022737 000002 002734 CMP #2,FLIPLOC
5920 035246 001413 BEQ 2$
5921 035250 022737 000003 002734 CMP #3,FLIPLOC
5922 035256 001414 BEQ 3$
5923 035260 005000 CLR R0
5924 035262 013704 002726 MOV ONES,R4
5925 035266 000414 BR 4$
5926 035270 1$: CLEAR R0,R4
035270 005000 CLR R0
035272 005004 CLR R4
5927 035274 000411 BR 4$
5928 035276 012700 000401 2$: MOV #401,R0
5929 035302 013704 002726 MOV ONES,R4
5930 035306 000404 BR 4$
5931 035310 012700 000401 3$: MOV #401,R0
5932 035314 012704 000401 MOV #401,R4
5933 035320 010037 026522 4$: MOV R0,WARN2
5934 035324 010037 026536 MOV R0,WARN3
5935 035330 010037 026562 MOV R0,WARN4
5936 035334 010037 026576 MOV R0,WARN5
5937 035340 POP R0
035340 012600 MOV (SP)+,R0
5938 035342 000207 RETURN
5939
5940 035344 BACKGND:SUBTST <<SUBR WRITE BACKGROUND>>
:*****
:*SUBTEST SUBR WRITE BACKGROUND
:*****
5941 ;WRITES DATA FROM R2
5942 035344 104415 SAVREG
5943 035346 012700 060000 MOV #FIRST,R0
5944 035352 012701 040000 MOV #SIZE,R1
5945 035356 BMOV MTP000
035356 004537 043710 JSR R5,BLOCK1
035362 026376 MTP000
.DSABL CRF

```

5946	035364	012737	000207	177644	WARN6:	MOV	#207,UIPAR2
5947	035372	004737	026042			CALL	SUPD01
5948	035376	104416				RESREG	
5949	035400	000207				RETURN	

;WARNING PUTTING 'RETURN' INSTRUCTION AFTER WRITE

5953 035402

PCONFIG:SUBTST <<SUBR PRINT CONFIGURATION MAP>>

```

:*****
:*SUBTEST SUBR PRINT CONFIGURATION MAP
:*****
  
```

5954 035402

PUSH TKVEC,TKVEC+2,R0

```

MOV TKVEC,-(SP)
MOV TKVEC+2,-(SP)
MOV R0,-(SP)
  
```

035402 013746 000060
 035406 013746 000062
 035412 010046

5955 035414

MOV SP,PCONFS

;SAVE LAST GOOD SP

5956 035420 012737 035630 000060

MOV #PCONF2,TKVEC

5957 035426 012737 000340 000062

MOV #340,TKVEC+2

5958 035434 017700 145334

MOV @\$TKB,R0

;KILL ANY OLD INTERRUPT

5959 035440 042737 000200 177776

BIC #BIT7,PSW

;LOWER CPU PRIORITY TO 140

5960 035446 052777 000100 145316

BIS #BIT6,@\$TKS

;ENABLE KEYBOARD INTERRUPTS

5961

5962 035454

TYPE MSG001

035454 104401 111254

TYPEIT ,MSG001

.DSABL CRF

5963 035460

TYPE MSG002

035460 104401 111336

TYPEIT ,MSG002

.DSABL CRF

5964 035464

TYPE MSG003

035464 104401 111413

TYPEIT ,MSG003

.DSABL CRF

5965

;IF FAT PAPER ON TERMINAL GOTO 15

5966 035470

IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1

035470 032777 000020 145270

BIT #SW4,@SWR

035476 001402

BEG L411

035500 000137 035572

JMP PCONF1

035504

L411:::~::~

5967 035504

MOV #60,R0

035510 010004

MOV R0,R4

5969 035512

CLEAR R1,R3

035512 005001

CLR R1

035514 005003

CLR R3

5970 035516

TYPE MSG004

035516 104401 111520

TYPEIT ,MSG004

.DSABL CRF

5971 035522

CALL TCONFIG

;GO TYPE CONFIGURATION (1ST HALF)

5972 035526

TYPE \$CRLF

035526 104401 003010

TYPEIT ,CRLF

.DSABL CRF

5973 035532

TYPE MSG017

;PRINT SPACE(S)

035532 104401 112150

TYPEIT ,MSG017

.DSABL CRF

5974 035536

TYPE MSG011

035536 104401 111745

TYPEIT ,MSG011

.DSABL CRF

5975 035542

TYPE \$CRLF

035542 104401 003010

TYPEIT ,CRLF

.DSABL CRF

5976 035546

TYPE MSG017

;PRINT SPACE(S)

035546 104401 112150

TYPEIT ,MSG017

.DSABL CRF

5977 035552

TYPE MSG012

035552 104401 112033

TYPEIT ,MSG012

.DSABL CRF

```

5978 035556 012701 000550      MOV      #60.*2*3,R1
5979 035562 010103      MOV      R1,R3
5980 035564 004737 035664      CALL    TCONFIG
5981 035570 000417      BR      PCONF2
5982
5983 035572 012700 000170      PCONF1: MOV    #120.,R0
5984 035576 010004      MOV    R0,R4
5985 035600      CLEAR  R1,R3
           035600 005001      CLR    R1
           035602 005003      CLR    R3
5986 035604      TYPE   MSG014      ;SPACE
           035604 104401 112132      TYPEIT ,MSG014
           .DSABL CRF
5987 035610      TYPE   MSG011
           035610 104401 111745      TYPEIT ,MSG011
           .DSABL CRF
5988 035614      TYPE   MSG004
           035614 104401 111520      TYPEIT ,MSG004
           .DSABL CRF
5989 035620      TYPE   MSG012
           035620 104401 112033      TYPEIT ,MSG012
           .DSABL CRF
5990 035624 004737 035664      CALL    TCONFIG
5991
5992 035630 013706 035662      PCONF2: MOV    PCONFS,SP      ;RESTORE STACK
5993 035634 042777 000100 145130      BIC    #BIT6,@$TKS
5994 035642 117700 145126      MOVB  @$TKB,R0      ;READ CHAR TO KILL FLAG
5995 035646      POP    R0,TKVEC+2,TKVEC
           035646 012600      MOV (SP)+,R0
           035650 012637 000062      MOV (SP)+,TKVEC+2
           035654 012637 000060      MOV (SP)+,TKVEC
5996 035660 000207      RETURN
5997
5998 035662 000000      PCONF2: 0      ;STACK SAVED HERE!
    
```

6002 035664

SUBTST <<SUBR TYPE CONFIGURATION>>

```

*****
*SUBTEC          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
*****
    
```

6003
6004
6005
6006
6007
6008
6009
6010
6011
6012
6013
6014

```

*****
** ERROR **
*****
    
```

6015 035664 104401 111626

```

TCONFIG:TYPE      MSG005
TYPEIT      ,MSG005
.DSABL      CRF
1$: BIT      #BIT11,CONFIG(R1) ;ERROR ON THIS BANK?
BEQ         2$ ;NO - SKIP
TYPE      MSG013
TYPEIT      ,MSG013
.DSABL      CRF
:PRINT 'X'
BR         3$
2$: TYPE      MSG014 ;PRINT SPACE
TYPEIT      ,MSG014
.DSABL      CRF
3$: ADD      #6,R1 ;BUMP POINTER
SOB        R0,1$ ;LOOP TILL DONE
MOV        R4,R0
MOV        R3,R1
    
```

6016 035670 032761 004000 003016
6017 035676 001403
6018 035700 104401 112130
6019 035704 000402
6020 035706 104401 112132
6021 035712 062701 000006
6022 035716 077014
6023 035720 010400
6024 035722 010301

```

*****
** CPU'S **
*****
    
```

6025
6026
6027
6028
6029 035724 104401 111707

```

TYPE      MSG008
TYPEIT      ,MSG008
.DSABL      CRF
4$: MOV      CONFIG(R1),R5 ;GET CPU BITS
ASH        #-4,R5 ;CLEAR NON INTERESTING BITS
BIC        #^C17,R5
CMP        #10.,R5
BPL        8$
ADD        #67,R5 ;MAKE NUMBERS OVER 9 HEX
BR         9$
8$: ADD        #60,R5 ;MAKE ASCII
9$: MOVB     R5,MSG015 ;PLUG INTO MEMORY
TYPE      MSG015
TYPEIT      ,MSG015
.DSABL      CRF
ADD        #6,R1 ;BUMP POINTER
SOB        R0,4$ ;LOOP TILL DONE
MOV        R4,R0
MOV        R3,R1
    
```

6030 035730 016105 003016
6031 035734 072527 177774
6032 035740 042705 177760
6033 035744 022705 000012
6034 035750 100003
6035 035752 062705 000067
6036 035756 000402
6037 035760 062705 000060
6038 035764 110537 112134
6039 035770 104401 112134
6040 035774 062701 000006
6041 036000 077025
6042 036002 010400
6043 036004 010301

```

6046
6047
6048
6049 036006 104401 111675
036006 104401 111675
6050
6051 036012 016105 003016
6052 036016 072527 177763
6053 036022 042705 177774
6054 036026 005705
6055 036030 001004
6056 036032 112737 000077 112134
6057 036040 000435
6058 036042 016105 003016
6059 036046 042705 177770
6060 036052 022705 000001
6061 036056 001002
6062 036060 006305
6063 036062 000420
6064 036064 022705 000002
6065 036070 001002
6066 036072 006305
6067 036074 000413
6068 036076 022705 000004
6069 036102 001003
6070 036104 012705 000010
6071 036110 000405
6072 036112 005705
6073 036114 001403
6074 036116 012705 000077
6075 036122 000402
6076 036124 062705 000060
6077 036130 110537 112134
6078 036134 104401 112134
036134 104401 112134
6079 036140 005737 002464
036140 005737 002464
036144 001401
036146 000207
036150
6080 036150 062701 000006
6081 036154 077062
6082 036156 010400
6083 036160 010301
6084
6085
6086
6087
6088
6089 036162 104401 111721
036162 104401 111721
6090 036166 016105 003016
6091 036172 072527 177763
6092 036176 042705 177774

```

```

:*****
:** INTERLEAVE **
:*****
TYPE MSG007
TYPEIT ,MSG007
.DSABL CRF
:THIS IS AN ENTRY POINT FROM ERROR REPORTS
TCFIG1: MOV CONFIG(R1),R5
ASH #-13.,R5 ;GET MEMORY TYPE
BIC #^C3,R5 ;CLEAR NON INTERESTING BITS
TST R5
BNE 1$
MOV# #'?,MSG015
BR 7$
1$: MOV CONFIG(R1),R5
BIC #^C7,R5 ;GET INTERLEAVE BITS
CMP #1,R5
BNE 2$
ASL R5 ;2 TO 1
BR 5$
2$: CMP #2,R5
BNE 3$
ASL R5 ;4 TO 1
BR 5$
3$: CMP #4,R5
BNE 4$
MOV #8.,R5 ;8 TO 1
BR 5$
4$: TST R5
BEQ 5$
MOV #'?,R5
BR 6$
5$: ADD #60,R5 ;MAKE ASCII
6$: MOV# R5,MSG015 ;PLUG INTO MEMORY
7$: TYPE MSG015
TYPEIT ,MSG015
.DSABL CRF
IF NOTAB NE #0 THEN $RETURN
TST NOTAB
BEQ L412
RTS PC
L412:::::
ADD #6,R1 ;BUMP POINTER
SOB R0,TCFIG1 ;LOOP TILL DONE
MOV R4,R0
MOV R3,R1
:*****
:** MEMORY TYPE **
:*****
.ENABL LSB
TYPE MSG009
TYPEIT ,MSG009
.DSABL CRF
8$: MOV CONFIG(R1),R5
ASH #-13.,R5 ;GET MEMORY TYPE
BIC #^C3,R5 ;CLEAR NON INTERESTING BITS

```

```

6093 036202          IF R5 EQ #1
      036202 020527 000001
      036206 001004
6094 036210          MOV B #'K,MSG015
      112737 000113 112134
6095 036216          ELSE
      036216 000413
      036220
6096 036220          IF #SLAVE3!SLAVE2!SLAVE1!MASTER SET.IN CONFIG(R1)
      036220 032761 000360 003016
      036226 001404
      036230 112737 000112 112134
6097 036230          MOV B #'J,MSG015
6098 036236          ELSE
      036236 000403
      036240
6099 036240          MOV B #'O,MSG015
6100 036246          END :OF IF #SLAVE3!SLAVE2!SLAVE1!MASTER
      036246
6101 036246          END :OF IF R5
      036246
6102 036246          TYPE MSG015
      036246 104401 112134
      TYPEIT ,MSG015
      .DSABL CRF
6103 036252 062701 000006
6104 036256 077035
6105 036260 010400
6106 036262 010301
6107
6108
6109
6110
6111
6112 036264          TYPE MSG016
      036264 104401 112136
      TYPEIT ,MSG016
      .DSABL CRF
6113 036270 016105 003016 9$: MOV CONFIG(R1),R5
6114 036274 032705 000010 BIT #BIT3,R5
6115 036300 001404 BEQ 10$
6116 036302 112737 000115 112134 MOV B #'M,MSG015
6117 036310 000417 BR 12$
6118 036312 032705 020014 10$: BIT #BIT13!14,R5
6119 036316 001004 BNE 11$
6120 036320 112737 000077 112134 MOV B #'?,MSG015
6121 036326 000410 BR 12$
6122 036330 072527 177770 11$: ASH #-8,R5
6123 036334 042705 177774 BIC #^C3,R5
6124 036340 062705 000060 ADD #60,R5
6125 036344 110537 112134 MOV B R5,MSG015
6126 036350          TYPE MSG015
      036350 104401 112134
      TYPEIT ,MSG015
      .DSABL CRF
6127 036354 062701 000006
6128 036360 077035
6129 036362 010400
6130 036364 010301
6131
6132
6133
    
```

CMP R5,#1
BNE L413

BR L414

L413:::~::~

BIT #SLAVE3!SLAVE2!SLAVE1!MA
BEQ L415

BR L416

L415:::~::~

L416:::~::~

L414:::~::~

:BUMP POINTER
:LOOP TILL DONE

:EXTERNAL INTERLEAVE?
:NO - SKIP
:INDICATE MULTIPLE BOXES

:MJ11?
:NO - SKIP
:I DON'T KNOW HOW MJ11'S INTERLEAVE

:GET BOX NUMBER
:CLEAR NON INTERESTING BITS
:MAKE ASCII
:PLUG INTO MEMORY

:BUMP POINTER
:LOOP TILL DONE

:*****
:** PROTECTED **

```

6134
6135 036366 104401 111733
6136 036372 005761 003016 13$:
6137 036376 100004
6138 036400 112737 000120 112134
6139 036406 000407
6140 036410 032761 010000 003016 14$:
6141 036416 001406
6142 036420 112737 000113 112134
6143 036426 104401 112134 15$:
036426
6144 036432 000402
6145 036434 104401 112132 16$:
036434
6146 036440 062701 000006 17$:
6147 036444 077026
6148 036446 010400
6149 036450 010301
6150 036452 000207
:*****
TYPE MSG010
TYPEIT ,MSG010
.DSABL CRF
TST CONFIG(R1) ;BANK PROTECTED?
BPL 14$ ;NO - SKIP
MOVB #'P,MSG015
BR 15$
BIT #BIT12,CONFIG(R1) ;PROTECTED REGION OF MK11 BOX?
BEQ 16$ ;NO - SKIP
MOVB #'K,MSG015
TYPE MSG015
TYPEIT ,MSG015
.DSABL CRF
BR 17$
TYPE MSG014 ;PRINT SPACE
TYPEIT ,MSG014
.DSABL CRF
ADD #6,R1 ;BUMP POINTER
SOB R0,13$ ;LOOP TILL DONE
MOV R4,R0
MOV R3,R1
RETURN
    
```

```

6154                                     .SBTTL TRAP PARITY ERROR HANDLER
6155                                     :*****
6156                                     :VECTOR TO HERE FROM TRAPS TO 114
6157                                     :IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
6158                                     :*****
6159
6160                                     :
6161                                     :
6162                                     :
6163                                     :
6164                                     :
6165                                     :
6166                                     :
6167                                     :
6168 036454 022737 000001 002100 PARITY: CMP #1,NOPAR ;COUNTING PARITY ERRORS?
6169 036462 001023 BNE 3$ ;NO - SKIP
6170 036464 032737 000001 177744 BIT #BIT0,MEMERR ;MEMORY TIMEOUT?
6171 036472 001402 BEQ 1$ ;NO - SKIP
6172 036474 000137 036646 JMP NONEXIST ;YES - SIMULATE A TRAP TO 4
6173 036500 005237 002074 1$: INC PARCNT ;PARITY ERROR COUNTER + 1
6174 036504 022737 000001 002074 CMP #1,PARCNT ;FIRST PARITY ERROR?
6175 036512 001003 BNE 2$ ;NO - SKIP
6176 036514 013737 177740 002042 MOV LOADRS,PADDRESS ;SAVE 1ST BAD PARITY ADDRESS
6177 036522 013737 177744 177744 2$: MOV MEMERR,MEMERR ;CLEAR ERROR BITS
6178 036530 000002 RTI
6179 036532 022737 000002 002100 3$: CMP #2,NOPAR ;ACTION CODE = 2 ?
6180 036540 001016 BNE 6$ ;NO - SKIP
6181 036542 SET ABORTFLAG ;YES
6182 036542 012737 177777 002142 MOV #-1,ABORTFLAG
6183 036550 004737 036770 CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6184 036554 063716 002406 ADD PCBUMP,(SP) ;UPDATE RETURN PC
6185 036560 013737 177744 177744 MOV MEMERR,MEMERR ;CLEAR ERROR BITS
6186 036566 042766 000004 000002 BIC #BIT2,2(SP) ;SHOW FAILURE BY .NE.
6187 036574 000002 RTI
6188 036576 022737 000003 002100 6$: CMP #3,NOPAR ;ACTION CODE = 3 ?
6189 036604 001006 BNE 7$ ;NO - SKIP
6190 036606 013716 002372 MOV PARTHERE,(SP)
6191 036612 013737 177744 177744 MOV MEMERR,MEMERR ;CLEAR ERROR BITS
6192 036620 000002 RTI
6193 036622 022737 000004 002100 7$: CMP #4,NOPAR
6194 036630 001001 BNE 8$
6195 036632 000722 BR 1$
6196 036634 004737 036770 8$: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
6197 036640 FATAL 32
036640 005237 002070 INC FATAL$ ;SET FATAL INDICATOR
036644 104032 ERROR +32
.DSABL CRF

```

```

6200 .SBTTL TRAP NON-EXISTANT MEMORY (HOLES) HANDLER
6201 :*****
6202 :VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
6203 : CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
6204 : 1) IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
6205 : 2) TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
6206 :*****
6207
6208 036646 022737 000001 002102 NONEXIST: CMP #1, NONEM ;COUNTING NON-EXISTANT MEMORY ERRORS?
6209 036654 001014 BNE 2$ ;NO - SKIP
6210 036656 005237 002072 INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
6211 036662 022737 000001 002072 CMP #1, NEMCNT ;FIRST ERROR?
6212 036670 001005 BNE 1$ ;NO - SKIP
6213 036672 010037 002040 MOV R0, ADDRESS ;ASSUME R0 CONTAINS THE ADDRESS ACCESSED
6214 036676 013737 177744 177744 MOV MEMERR, MEMERR ;CLEAR OUT HIADRS & LOADRS REG'S
6215 036704 000002 1$: RTI
6216 036706 005237 002072 2$: INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
6217 036712 012701 000001 MOV #1, R1 ;DUMMY UP R1 FOR A FORCED SOB EXIT
6218 036716 013737 177744 177744 MOV MEMERR, MEMERR ;CLEAR OUT HIADRS & LOADRS REG'S
6219 036724 000002 RTI
6220
6221 :*****
6222 .SBTTL TRAP TIMEOUT (TRAP TO 4) HANDLER
6223 036726 004737 036770 TIMEOUT: CALL BADSTACK ;FIND BAD SP, PC, PSW OFF STACK
6224 036732 FATAL 6
6224 036732 005237 002070 INC FATAL$ ;SET FATAL INDICATOR
6224 036736 104006 ERROR +6
6225 .DSABL CRF
6226 :*****
6227 .SBTTL TRAP MEMORY MANAGEMENT (TRAP TO 250) HANDLER
6227 036740 004737 036770 MMTRAP: CALL BADSTACK ;FIND BAD SP, PC, PSW OFF STACK
6228 036744 FATAL 7
6228 036744 005237 002070 INC FATAL$ ;SET FATAL INDICATOR
6228 036750 104007 ERROR +7
6229 .DSABL CRF
6230 .SBTTL TRAP RESERVED INSTRUCTION HANDLER
6230 036752 004737 036770 PDP1105: CALL BADSTACK ;FIND BAD SP, PC, PSW OFF STACK
6231 036756 FATAL 5
6231 036756 005237 002070 INC FATAL$ ;SET FATAL INDICATOR
6231 036762 104005 ERROR +5
6232 .DSABL CRF
6233 036764 IITRAP: SUBTST <<UNEXPECTED IIST TRAP HANDLER>>
6234 :*****
6234 036764 104057 *SUBTEST UNEXPECTED IIST TRAP HANDLER
6235 036766 000002 ERROR +57
6236 RTI
6237 036770 BADSTACK: SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
6238 :*****
6238 036770 010637 002032 *SUBTEST FIND BAD SP, PC, & PSW FROM STACK
6239 036774 062737 000002 002032 MOV SP, BADSP
6240 037002 016637 000002 002026 ADD #2, BADSP
6241 037010 016637 000004 002036 MOV 2(SP), BADPC
MOV 4(SP), BADPSW
  
```

6242 037016 000207

RETURN

```

6245          .SBTTL TRAP   KERNEL TRAP HANDLER
6246          ;*****
6247          ;KERNEL IS A TRAP THAT COMES HERE
6248          ;*****
6249
6250 037020 042766 140000 000002 $KERNEL:   BIC   #140000,2(SP)
6251 037026 000002          RTI
6252          ;*****
6253          .SBTTL TRAP   ENERGIZE TRAP HANDLER
6254 037030 052737 001001 177572 $ENERGIZE: BIS   #BIT9!BIT0,MMRO
6255 037036 000002          RTI
6256          ;*****
6257          .SBTTL TRAP   DEENERGIZE TRAP HANDLER
6258 037040 042737 001001 177572 $DEENERGIZE: BIC #BIT9!BIT0,MMRO
6259 037046 000002          RTI
6260          ;*****
6261          .SBTTL TRAP   CACHON TRAP HANDLER
6262 037050 013737 002700 177746 $CACHN:  MOV   CACHK,CONTRL   ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
6263 037056 052737 000001 177746   BIS   #BIT0,CONTRL   ;DISABLE TRAPS (BUT NOT ABORTS)
6264 037064 000002          RTI
6265          ;*****
6266          .SBTTL TRAP   CACHOFF TRAP HANDLER
6267          ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
6268 037066 052737 001415 177746 $CACHF:  BIS   #BIT0!BIT2!BIT3!BIT8!BIT9,CONTRL
6269 037074 000002          RTI

```



```

6302          .SBTTL TRAP READ CSR TRAP HANDLER
6303          ;READ THE CORRECT CSR INTO LOCATIONS CSR & CSR+2
6304 037220   $READC: PUSH    R0
6305 037220   010046          MOV    #MK11ADD,R0          ;CREATE CSR ADDRESS
6306 037222   012700 172100   ADD    CSRNO,R0
6307 037232   005737 002254   TST    CSRREL          ;CSR'S RELOCATED?
6308 037236   001005          BNE    1$              ;YES - SKIP
6309 037240   012037 002144   MOV    (R0)+,CSR       ;READ IT
6310 037244   011037 002146   MOV    (R0),CSR+2     ;READ IT
6311 037250   000415          BR     2$
6312 037252   042700 020000   1$:   BIC    #20000,R0
6313 037256   013737 177746 002370   MOV    CONTRL,OLDCACHE ;SAVE CACHE STATUS
6314 037264   104424          CACHOFF ;TURN CACHE OFF
6315 037266   012037 002144   MOV    (R0)+,CSR       ;READ IT
6316 037272   011037 002146   MOV    (R0),CSR+2     ;READ IT
6317 037276   013737 002370 177746   MOV    OLDCACHE,CONTRL ;RESTORE CACHE
6318 037304   037304 012600   2$:   POP    R0
6319 037306   000002          RTI
6320
6321          .SBTTL TRAP TEST (R1) & READ CSR CAREFULLY
6322 037310   $TSTRD: PUSH    R0
6323 037312   010046          MOV    R0,-(SP)
6324 037316   004537 043710   BMOV   TSTRD1
6325 037320   037434          JSR   R5,BLOCK1
6326 037324   012700 172100   TSTRD1
6327 037326   063700 002256   .DSABL CRF
6328 037330   005737 002254   MOV    #MK11ADD,R0     ;CREATE CSR ADDRESS
6329 037334   001003          ADD    CSRNO,R0
6330 037336   004737 176640   TST    CSRREL          ;CSR'S RELOCATED?
6331 037342   000413          BNE    1$              ;YES - SKIP
6332 037344   042700 020000   CALL   FASTCITY        ;CALL TO THE USER INSTRUCTION PAR'S
6333 037350   013737 177746 002370   BR     2$
6334 037356   004737 177640   1$:   BIC    #20000,R0
6335 037364   013737 002370 177746   MOV    CONTRL,OLDCACHE ;SAVE CACHE STATUS
6336 037372   037372 012600   CACHOFF ;TURN CACHE OFF
6337 037374   032737 000020 002144   CALL   FASTCITY        ;CALL TO THE USER INSTRUCTION PAR'S
6338 037402   001410          MOV    OLDCACHE,CONTRL ;RESTORE CACHE
6339 037404   032737 100000 002144   ;IF SINGLE BIT ERROR ONLY - SET CARRY BIT
6340 037412   001004          2$:   POP    R0
6341 037414   052766 000001 000002   MOV    (SP)+,R0
6342 037422   000403          IF #BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
6343 037424   042766 000001 000002   BIT   #BIT4,CSR
6344 037432   005010          BEQ   L423
6345 037434   005010          BIT   #BIT15,CSR
6346 037436   005010          BNE   L423
6347 037438   005010          BIS   #BIT0,2(SP)
6348 037440   005010          ELSE
6349 037442   005010          BR   L424
6350 037444   005010          L423:.....
6351 037446   005010          BIC   #BIT0,2(SP)
6352 037448   005010          END ;OF IF #BIT4
6353 037450   005010          RTI
6354 037452   005010          L424:.....
6355 037454   005010          TSTRD1: CLR    (R0)          ;V177640
    
```

6345	037436				
	037436	052737	040000	177776	
6346	037444	105711			
6347	037446	042737	140000	177776	
6348	037454	012037	002144		
6349	037460	011037	002146		
6350	037464	000207			

```

SUPERVISOR
BIS #BIT14,PSW
.DSABL CRF
TSTB (R1)
BIC #BIT15!BIT14,PSW
MOV (R0)+,CSR
MOV (R0),CSR+2
RETURN

```

```

;V177642 ;ENTER SUPERVISOR MODE
;DO IT TO IT!
;V177646
;V177650
;V177656
;V177662
;V177666

```

TRAP ECC DISABLE ALL CSR'S TRAP HANDLER

```

6354 .SBTTL TRAP ECC DISABLE ALL CSR'S TRAP HANDLER
6355 037466 012737 000002 002144 $ECCDIS:MOV #BIT1,CSR
6356 037474 005737 002122 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
                                TST KPFLAG
                                BEQ L425
                                BIS #BIT3,CSR
                                L425:~~~~~
6357 037510 005037 002146 CLR CSR+2
6358 037514 004737 040442 CALL CSROUT
6359 037520 000002 RTI
6360 .SBTTL TRAP ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
6361 037522 012737 000002 002144 $ECC1DIS:MOV #BIT1,CSR
6362 037530 005737 002122 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
                                TST KPFLAG
                                BEQ L426
                                BIS #BIT3,CSR
                                L426:~~~~~
6363 037544 005037 002146 CLR CSR+2
6364 037550 104425 LOADCSR
6365 037552 000002 RTI
6366 .SBTTL TRAP MK11 INITIALIZE ALL CSR'S TRAP HANDLER
6367 037554 012737 000001 002144 $ECCINIT:MOV #BIT0,CSR
6368 037562 005037 002146 CLR CSR+2
6369 037566 004737 040442 CALL CSROUT
6370 037572 000002 RTI
6371 .SBTTL TRAP MK11 INITIALIZE 1 SELECTED CSR TRAP HANDLER
6372 037574 012737 000001 002144 $ECC1INIT:MOV #BIT0,CSR
6373 037602 005037 002146 CLR CSR+2
6374 037606 104425 LOADCSR
6375 037610 000002 RTI
6376 .SBTTL TRAP ENABLE SBE PARITY TRAPS ON ALL CSR'S
6377 037612 012737 000003 002144 $ENASBE:MOV #BIT0!BIT1,CSR
6378 037620 005737 002122 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
                                TST KPFLAG
                                BEQ L427
                                BIS #BIT3,CSR
                                L427:~~~~~
6379 037634 005037 002146 CLR CSR+2
6380 037640 004737 040442 CALL CSROUT
6381 037644 000002 RTI
6382 .SBTTL TRAP ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
6383 037646 012737 000003 002144 $ENA1SBE:MOV #BIT0!BIT1,CSR
6384 037654 005737 002122 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
                                TST KPFLAG
                                BEQ L430
                                BIS #BIT3,CSR
                                L430:~~~~~
6385 037670 005037 002146 CLR CSR+2
6386 037674 104425 LOADCSR
6387 037676 000002 RTI
6388 .SBTTL TRAP WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
6389 037700 013737 002404 002144 $CBCSR:MOV CHECK,CSR ;BITS 14-8
6390 037706 052737 000006 002144 BIS #BIT1!BIT2,CSR ;CHECK MODE
6391 037714 005737 002122 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
                                TST KPFLAG
                                BEQ L431
                                BIS #BIT3,CSR
037722 052737 000010 002144

```

TRAP WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER

L431:::~::~

037730
 6392 037730 005037 002146
 6393 037734 004737 040442
 6394 037740 000002
 6395
 6396 037742 013737 002404 002144
 6397 037750 052737 000006 002144
 6398 037756
 037756 005737 002122
 037762 001403
 037764 052737 000010 002144
 037772
 6399 037772 005037 002146
 6400 037776 104425
 6401 040000 000002

CLR CSR+2
 CALL CSROUT
 RTI
 .SBTTL TRAP WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
 \$CB1CSR:MOV CHECK,CSR ;BITS 14-8
 BIS #BIT1!BIT2,CSR ;CHECK MODE
 IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3

TST KPFLAG
 BEQ L432
 BIS #BIT3,CSR

L432:::~::~

CLR CSR+2
 LOADCSR
 RTI

```

6405          .SBTTL TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER
6406 040002    $WASSBE: PUSH R1,R4
        040002    010146
        040004    010446
6407 040006    013701 002334    MOV MKCSRS,R1 ;GET CSR'S BYTE
6408 040012    005004          CLR R4
6409 040014          BEGIN LWSBE
        040014
6410 040014          FOR CSRNO := #0 TO #34 BY #4
        040014    005037 002256
        040020
6411 040020    106301          ASLB R1
6412 040022          ON.ERROR
        040022    103010
6413 040024    104426          READCSR
6414 040026          IF #BIT4 SET.IN CSR
        040026    032737 000020 002144
        040034    001403
6415 040036          SET R4
        040036    012704 177777
6416 040042          LEAVE LWSBE
        040042    000411
6417 040044          END ;OF IF #BIT4
        040044
6418 040044          END ;OF ON.ERROR
        040044
6419 040044          IFB R1 EQ #0 THEN LEAVE LWSBE
        040044    105701
        040046    001407
6420 040050          END ;OF FOR CSRNO
        040050    062737 000004 002256
        040056    023727 002256 000034
        040064    003755
        040066
6421 040066          END LWSBE
        040066
6422 040066    006004          ROR R4 ;SET C BIT FOR ERROR
6423 040070          POP R4,R1
        040070    012604
        040072    012601
6424 040074          ON.ERROR
        040074    103004
6425 040076    052766 000001 000002    BIS #BIT0,2(SP)
6426 040104          ELSE
        040104    000403
        040106
6427 040106    042766 000001 000002    BIC #BIT0,2(SP)
6428 040114          END ;OF ON.ERROR
        040114
6429 040114    000002          RTI
6430          .SBTTL TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
6431          ;ON RETURN IF CARRY IS SET THERE WAS A SBE
6432 040116    104426          $WAS1SBE: READCSR
6433 040120    042766 000001 000002    BIC #BIT0,2(SP) ;CLR C BIT ON STACK
6434 040126    032737 000020 002144    BIT #BIT4,CSR
6435 040134    001403          BEQ 1$
6436 040136    052766 000001 000002    BIS #BIT0,2(SP) ;SET C BIT ON STACK
  
```

```

MOV R1,-(SP)
MOV R4,-(SP)
B51:
CLR CSRNO
B52:
BCC L433
BIT #BIT4,CSR
BEQ L434
MOV #-1,R4
BR E51
L434:
L433:
TSTB R1
BEQ E51
ADD #4,CSRNO
CMP CSRNO,#34
BLE B52
E52:
E51:
MOV (SP)+,R4
MOV (SP)+,R1
BCC L435
BR L436
L435:
L436:
  
```

CEMKABO 1170 MAIN MEMORY DIAG 1 MACRO M1113 03-AUG-79 16:50 PAGE 297-1
TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER

SEQ 0807

6437 040144 000002

18: RTI

```

6440          .SBTTL TRAP WAS THERE A DBE ON ANY CSR TRAP HANDLER
6441 040146          $WASDBE: PUSH R1,R4
        040146 010146          MOV R1,-(SP)
        040150 010446          MOV R4,-(SP)
6442 040152 013701 002334      MOV MKCSRS,R1 ;GET CSR'S BYTE
6443 040156 005004          CLR R4
6444 040160          BEGIN LWDBE
        040160          FOR CSRNO := #0 TO #34 BY #4
6445 040160          005037 002256          CLR CSRNO
        040164          B54:
6446 040164 106301          ASLB R1
6447 040166          ON.ERROR
        040166 103010          READCSR
6448 040170 104426          IF #BIT15 SET.IN CSR
6449 040172          032737 100000 002144          SET R4
        040172 001403          LEAVE LWDBE
6450 040202          012704 177777          END ;OF IF #BIT15
6451 040206          000411          END ;OF ON.ERROR
6452 040210          IFB R1 EQ #0 THEN LEAVE LWDBE
6453 040210          L440:
6454 040210          105701          TSTB R1
        040212 001407          BEQ E53
6455 040214          062737 000004 002256          ADD #4,CSRNO
        040222 023727 002256 000034          CMP CSRNO,#34
        040230 003755          BLE B54
6456 040232          END LWDBE
        040232          E54:
6457 040232 006004          ROR R4 ;SET C BIT FOR ERROR
6458 040234          POP R4,R1
        040234 012604          MOV (SP)+,R4
        040236 012601          MOV (SP)+,R1
6459 040240          ON.ERROR
        040240 103004          BCC L441
6460 040242 052766 000001 000002          BIS #BIT0,2(SP)
6461 040250          ELSE
        040250 000403          BR L442
        040252          L441:
6462 040252 042766 000001 000002          BIC #BIT0,2(SP)
6463 040260          END ;OF ON.ERROR
        040260          L442:
6464 040260 000002          RTI
6465          .SBTTL TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
6466          :ON RETURN IF CARRY IS SET THERE WAS A DBE
6467 040262 104426          $WAS1DBE: READCSR
6468 040264 005737 002144          TST CSR ;DBE?
6469 040270 100010          BPL 3$ ;NO - SKIP
6470 040272 005737 002146          TST CSR+2 ;DBE IN OTHER WORD?
6471 040276 100401          BMI 2$ ;YES - SKIP
    
```

6472	040300	104021				ERROR	+21		;NO - ONLY ONE FLAG POPPED UP!
6473	040302	052766	000001	000002	2\$:	BIS	#BIT0,2(SP)		;SET C BIT ON STACK
6474	040310	000002				RTI			
6475	040312	042766	000001	000002	3\$:	BIC	#BIT0,2(SP)		;CLR C BIT ON STACK
6476	040320	000002				RTI			

```

6479          .SBTTL TRAP CLEAR ALL MK11 CSR'S TRAP HANDLER
6480 040322 $CLRCSR:CLR CSR,CSR+2
        040322 005037 002144          CLR CSR
        040326 005037 002146          CLR CSR+2
6481 040332 004737 040442          CALL CSROUT
6482 040336 000002          RTI
6483          .SBTTL TRAP CLEAR 1 SELECTED MK11 CSR TRAP HANDLER
6484 040340 $CLR1CSR:CLR CSR,CSR+2
        040340 005037 002144          CLR CSR
        040344 005037 002146          CLR CSR+2
6485 040350 104425          LOADCSR
6486 040352 000002          RTI
6487          .SBTTL TRAP ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
6488          :CHECKBITS ALREADY IN LOC "CSR"
6489 040354 052737 000006 002144 $CHKDIS:BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
6490 040362          IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
        040362 005737 002122          TST KPFLAG
        040366 001403          BEQ L443
        040370 052737 000010 002144  BIS #BIT3,CSR
        040376          L443:~~~~~
6491 040376 005037 002146          CLR CSR+2
6492 040402 004737 040442          CALL CSROUT
6493 040406 000002          RTI
6494          .SBTTL TRAP ECC DISABLE, CHECH MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
6495          :CHECKBITS ALREADY IN LOC "CSR"
6496 040410 052737 000006 002144 $CHK1DIS:BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
6497 040416          IF KPFLAG IS TRUE THEN LET CSR := CSR SET.BY #BIT3
        040416 005737 002122          TST KPFLAG
        040422 001403          BEQ L444
        040424 052737 000010 002144  BIS #BIT3,CSR
        040432          L444:~~~~~
6498 040432 005037 002146          CLR CSR+2
6499 040436 104425          LOADCSR
6500 040440 000002          RTI
  
```

```

6503 040442 CSROUT: SUBTST <<SUBR WRITE IN ALL CSR'S>>
:*****
:SUBTEST SUBR WRITE IN ALL CSR'S
:*****
6504 040442 010146 PUSH R1
6505 040444 013701 002334 MOV MKCSRS,R1 ;GET CSR'S BYTE
6506 040450 BEGIN LCSROUT
6507 040450 005037 002256 FOR CSRNO := #0 TO #34 BY #4
6508 040454 106301 ASLB R1
6509 040456 ON.ERROR
6510 040460 103001 LOADCSR
6511 040462 104425 END ;OF ON.ERROR
6512 040462 IFB R1 EQ #0 THEN LEAVE LCSROUT
6513 040466 105701
040464 001407
040466 062737 000004 002256
040474 023727 002256 000034
040502 003764
6514 040504 END LCSROUT
6515 040504 POP R1
6516 040506 012601 RETURN
6517 040506 000207
6518 .SBTTL TRAP INVALIDATE BACKGROUND PATTERN
6519 $INVALID:PUSH RO,R1
6520 040510 010046 MOV RO,-(SP)
6521 040512 010146 MOV R1,-(SP)
6522 040514 013701 002106 MOV BANK,R1
6523 040520 070127 000006 MUL #6,R1
040524 042761 020000 003020 BIC #BIT13,(CONFIG+2(R1))
6524 040532 012601 POP R1,RO
040534 012600 MOV (SP)+,R1
040536 000002 RTI MOV (SP)+,RO
    
```

B55:.....
 CLR CSRNO
 B56:.....
 BCC L445
 L445:.....
 TSTB R1
 BEQ E55
 ADD #4,CSRNO
 CMP CSRNO,#34
 BLE B56
 E56:.....
 E55:.....
 MOV (SP)+,R1
 MOV RO,-(SP)
 MOV R1,-(SP)
 MOV (SP)+,R1
 MOV (SP)+,RO

6528 040540

```
CHKGEN: SUBTST<<SUBR GENERATE CHECK BITS>>
:*****
:*SUBTEST SUBR GENERATE CHECK BITS
:*****
```

6529
6530
6531
6532
6533
6534
6535
6536

```
:CHECK BIT GENERATOR ROUTINE
:CALLING SEQUENCE IS:
:      MOV #WORD1,SOURCE ;SOURCE = ADDRESS OF DATA
:      CALL CHKGEN
:CHECK BITS RETURNED IN BITS 14-8 OF LOCATION CHECK
PUSH R0,R1,R2,R3,R4,R5
```

040540 010046
040542 010146
040544 010246
040546 010346
040550 010446
040552 010546
6537 040554 012702 000077
6538 040560 012703 040640
6539 040564 013705 002402
6540 040570 012501
6541 040572 011500
6542
6543 040574 006704
6544 040576 142304
6545 040600 074402
6546 040602 073027 000001
6547 040606 001372
6548
6549 040610 042702 177600
6550 040614 000302
6551 040616 010237 002404
6552 040622
040622 012605
040624 012604
040626 012603
040630 012602
040632 012601
040634 012600
6553 040636 000207

```
MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)
MOV R3,-(SP)
MOV R4,-(SP)
MOV R5,-(SP)
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 14-8
MOV R2,CHECK
POP R5,R4,R3,R2,R1,R0
MOV (SP)+,R5
MOV (SP)+,R4
MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
RETURN
```

6556	040640		CHKTAB: ;BYTE #3	
6557	040640	200	.BYTE ^C177	;BIT 31
6558	040641	301	.BYTE ^C076	;BIT 30
6559	040642	302	.BYTE ^C075	;BIT 29
6560	040643	203	.BYTE ^C174	;BIT 28
6561	040644	304	.BYTE ^C073	;BIT 27
6562	040645	205	.BYTE ^C172	;BIT 26
6563	040646	206	.BYTE ^C171	;BIT 25
6564	040647	307	.BYTE ^C070	;BIT 24
6565			;BYTE #2	
6566	040650	310	.BYTE ^C067	;BIT 23
6567	040651	211	.BYTE ^C166	;BIT 22
6568	040652	212	.BYTE ^C165	;BIT 21
6569	040653	313	.BYTE ^C064	;BIT 20
6570	040654	214	.BYTE ^C163	;BIT 19
6571	040655	315	.BYTE ^C062	;BIT 18
6572	040656	316	.BYTE ^C061	;BIT 17
6573	040657	217	.BYTE ^C160	;BIT 16
6574			;BYTE #1	
6575	040660	320	.BYTE ^C057	;BIT 15
6576	040661	221	.BYTE ^C156	;BIT 14
6577	040662	222	.BYTE ^C155	;BIT 13
6578	040663	323	.BYTE ^C054	;BIT 12
6579	040664	224	.BYTE ^C153	;BIT 11
6580	040665	325	.BYTE ^C052	;BIT 10
6581	040666	326	.BYTE ^C051	;BIT 9
6582	040667	227	.BYTE ^C150	;BIT 8
6583			;BYTE #0	
6584	040670	340	.BYTE ^C037	;BIT 7
6585	040671	241	.BYTE ^C136	;BIT 6
6586	040672	242	.BYTE ^C135	;BIT 5
6587	040673	343	.BYTE ^C034	;BIT 4
6588	040674	244	.BYTE ^C133	;BIT 3
6589	040675	345	.BYTE ^C032	;BIT 2
6590	040676	346	.BYTE ^C031	;BIT 1
6591	040677	247	.BYTE ^C130	;BIT 0

6595 040700

SUBTST<<SUBR MAPPER>>

 :*SUBTEST SUBR MAPPER

6596
 6597
 6598
 6599
 6600
 6601
 6602
 6603
 6604

:THIS SUBROUTINE MAPS THE MEMORY BANK (16K WORDS = 1 BANK)
 :IN R3 TO THE TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000 - 157777)).
 :CALL MOV BANKNO,R3 ;SET UP BANK ARGUMENT
 :CALL CALL MAPPER ;ACTUAL CALL
 :RETURN ;ONLY RETURN

040700
 040700 010046
 040702 010146
 040704 010246
 040706 010446
 040710 010546

:SET SUPERVISOR UP FOR 1 TO 1 MAP
 MAPPER: PUSH R0,R1,R2,R4,R5
 MOV R0,-(SP)
 MOV R1,-(SP)
 MOV R2,-(SP)
 MOV R4,-(SP)
 MOV R5,-(SP)

6605 040712 012700 172340
 6606 040716 012701 172240
 6607 040722 012702 077406
 6608 040726 012704 172200
 6609 040732 012705 000010
 6610 040736 012021
 6611 040740 010224
 6612 040742 077503
 6613 040744 012741 177600
 6614

MOV #KIPAR0,R0 ;FIRST AREA TO MAP TO
 MOV #SIPAR0,R1 ;FIRST ADDRESS REGISTER
 MOV #77406,R2 ;CONSTANT FOR 4K PAGE, UP, R/W
 MOV #SIPDR0,R4 ;FIRST DISCRIPTOR REGISTER
 MOV #8,R5 ;COUNTER
 1\$: MOV (R0)+,(R1)+ ;PUT IN SUPERVISOR ADDRESS
 MOV R2,(R4)+ ;PUT IN SUPERVISOR DISCRIPTOR
 SOB R5,1\$;LOOP TILL DONE
 MOV #177600,-(R1) ;CORRECT LAST FIELD FOR PERIPHERALS PAGE

6615
 6616 040750 022703 000170
 6617 040754 001412
 6618 040756 072327 000011
 6619
 6620 040762 012701 172246
 6621 040766 012702 000004
 6622 040772 010321
 6623 040774 062703 000200
 6624 041000 077204
 6625 041002

:SET UP SUPERVISOR 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
 MOV #4,R2 ;COUNTER
 2\$: MOV R3,(R1)+ ;PLUG IN PAR INFO
 ADD #200,R3 ;BUMP ADDRESS 4K
 SOB R2,2\$;LOOP TILL DONE
 3\$: POP R5,R4,R2,R1,R0

041002 012605
 041004 012604
 041006 012602
 041010 012601
 041012 012600
 6626 041014 000207

MOV (SP)+,R5
 MOV (SP)+,R4
 MOV (SP)+,R2
 MOV (SP)+,R1
 MOV (SP)+,R0

6627
 6628 041016
 041016 010046
 041020 010146
 041022 010246
 041024 010346
 041026 010446

RETURN
 .SBTTL TRAP MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
 \$KMAP: PUSH R0,R1,R2,R3,R4
 MOV R0,-(SP)
 MOV R1,-(SP)
 MOV R2,-(SP)
 MOV R3,-(SP)
 MOV R4,-(SP)

6629 041030 005000
 6630 041032 012701 172340
 6631 041036 012702 077406
 6632 041042 012703 172300
 6633 041046 012704 000010

CLR R0 ;1ST AREA TO MAP TO
 MOV #KIPAR0,R1 ;FIRST ADDRESS
 MOV #77406,R2 ;CONSTANT FOR 4K PAGE,UP,R/W
 MOV #KIPDR0,R3 ;1ST PAGE DISCRIPTOR REGISTER
 MOV #8,R4 ;COUNTER

```
6634 041052 010021          1$:  MOV     R0,(R1)+      ;PUT IN KERNEL ADDRESS
6635 041054 010223          MOV     R2,(R3)+      ;PUT IN KERNEL DISRIPTOR
6636 041056 062700 000200    ADD     #200,R0        ;ADD ADDRESS CONSTANT FOR 4K CHANGE
6637 041062 077405          SOB     R4,1$         ;LOOP TILL DONE
6638 041064 012741 177600    MOV     #177600,-(R1)  ;THE PERIPHERALS PAGE TO KIPAR7
6639 041070 012741 177400    MOV     #177400,-(R1)  ;AND NEXT LOWER PAGE TO KIPAR6
6640 041074          IF CPUBIT NE #MASTER ;IF I'M NOT THE MASTER
        041074 023727 002114 000020    CMP     CPUBIT,#MASTER
        041102 001404          BEQ     L446
6641 041104 013700 002710    MOV     LOADHOME,R0   ;MAP TO LOADERS LIKE THE MASTER DID!
6642 041110 004737 042370    CALL   NEWLOAD
6643 041114          END ;OF IF CPUBIT
        041114          L446:;:::::
6644 041114          POP     R4,R3,R2,R1,R0
        041114 012604          MOV     (SP)+,R4
        041116 012603          MOV     (SP)+,R3
        041120 012602          MOV     (SP)+,R2
        041122 012601          MOV     (SP)+,R1
        041124 012600          MOV     (SP)+,R0
6645 041126 000002          RTI
```

```

6649 041130 RELOCATE:SUBST <<RELOCATE PROGRAM>>
:*****
:*SUBTEST RELOCATE PROGRAM
:*****
6650 041130 032777 010000 141630 IF #SW12 SET.IN @SWR THEN $RETURN ERROR
041130 001402 BIT #SW12,@SWR
041140 000261 BEQ L447
041142 000207 SEC
041144 RTS PC
6651 041144 BEGIN LOADERBANK L447:*****
041144 B57:*****
6652 041144 FOR BANK := #1 TO #167
041144 012737 000001 002106 MOV #1,BANK
041152 B60:*****
6653 041152 004737 042422 CALL EXBANK
6654 041156 IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE
041156 005737 C02124 TST ACFLAG
041162 001431 BEQ L450
041164 005737 002130 TST PFLAG
041170 001026 BNE L450
041172 005737 002136 TST BMFLAG
041176 001023 BNE L450
6655 041200 013700 002106 MOV BANK,R0
6656 041204 010037 002532 MOV R0,LOADBANK
6657 041210 013701 002710 MOV LOADHOME,R1
6658 041214 004737 042072 CALL BANKMOV
6659 041220 004737 042370 CALL NEWLOAD ;MAP NEW LOADER BANK IN KERNEL
6660 041224 013701 002110 MOV BANKINDEX,R1
6661 041230 052761 100000 003020 BIS #BIT15,CONFIG+2(R1) ;MARK LOADER
6662 041236 042761 020000 003020 BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
6663 041244 LEAVE LOADERBANK
041244 000416 BR E57
6664 041246 END ;OF IF ACFLAG
041246 L450:*****
6665 041246 END ;OF FOR BANK
041246 005237 002106 INC BANK
041252 023727 002106 000167 CMP BANK,#167
041260 003734 BLE B60
041262 E60:*****
6666 041262 IF #SW13 OFF.IN @SWR
041262 032777 020000 141476 BIT #SW13,@SWR
041270 001002 BNE L451
6667 041272 TYPE MSG075 ;RELOCATION NOT POSSIBLE
041272 104401 115341 TYPEIT ,MSG075
.DSABL CRF .DSABL CRF
END ;OF IF #SW13
6668 041276 $RETURN ERROR L451:*****
6669 041276 000261 SEC
041300 000207 RTS PC
6670 041302 END LOADERBANK E57:*****
6671 041302 BEGIN FINDBANK B61:*****
6672 041302 FOR R1 := #3*2 TO #167*3*2 BY #6
041302 012701 000006 MOV #3*2,R1
  
```

```

6673 041306 032761 104000 003016      IF #BIT11!BIT15 OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT PROGRAM SPACE
      041306 001035                                BIT #BIT11!BIT15,CONFIG(R1)
6674 041316 032761 100000 003020      IF #BIT15 OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK
      041316 001031                                BIT #BIT15,CONFIG+2(R1)
6675 041326 033761 002114 003016      IF CPUBIT SET.IN CONFIG(R1) ;IF ACCESSABLE
      041326 001425                                BEQ L454
6676 041336 032761 060000 003016      IF #BIT13!BIT14 OFF.IN CONFIG(R1) THEN LEAVE FINDBANK ;IF MJ11
      041336 001436                                BIT #BIT13!BIT14,CONFIG(R1)
6677 041346 032761 020000 003016      IF #BIT13 SET.IN CONFIG(R1) AND #BIT14 OFF.IN CONFIG(R1) ;IF MK11
      041354 001415                                BEQ E61
      041356 032761 040000 003016      BIT #BIT13,CONFIG(R1)
      041364 001011                                BEQ L455
6678 041366 032761 010000 003010      IF #BIT12 SET.IN CONFIG-6(R1) AND #BIT12 OFF.IN CONFIG(R1)
      041374 001405                                BIT #BIT12,CONFIG-6(R1)
      041376 032761 010000 003016      BEQ L456
      041404 001001                                BIT #BIT12,CONFIG(R1)
6679                                     ;IF 1ST UNPROTECTED MK11 BANK
6680 041406 000415                                LEAVE FINDBANK
6681 041410                                     BR E61
6682 041410                                     END ;OF IF #BIT12
6683 041410                                     END ;OF IF #BIT13
6684 041410                                     END ;OF IF CPUBIT
6685 041410                                     END ;OF IF #BIT15
6686 041410                                     END ;OF IF #BIT11
      041410 062701 000006                                ADD #6,R1
      041414 020127 001312                                CMP R1,#167*3*2
      041420 003732                                BLE B62
6687 041422 032777 020000 141336      IF #SW13 OFF.IN @SWR
      041422 001002                                E62:::
      041430 001002                                BIT #SW13,@SWR
6688 041432 104401 115341      TYPE MSG075 ;RELOCATION NOT POSSIBLE
      041432 104401 115341      TYPEIT ,MSG075
      .DSABL CRF
      END ;OF IF #SW13
6689 041436                                     L457:::
6690 041436 $RETURN ERROR
      041436 000261                                     SEC
      041440 000207                                     RTS PC
6691 041442      END FINDBANK
6692 041442 042761 020000 003020      BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
6693 041450 005000      CLR R0
6694 041452 071027 000006      DIV #6,R0
    
```

6697	041456				RELOC1: LET NEWBANK := R0	
	041456	010037	002400			MOV R0,NEWBANK
6698	041462	004737	042240		CALL USERMAP	;MAP NEWBANK TO USER PAR
6699	041466				USER	;ENTER USER MODE
	041466	052737	140000	177776	BIS #BIT15!BIT14,PSW	;DO IT TO IT!
					.DSABL CRF	
6700	041474				BMOV 0,100000,SIZE	;MOVE PROGRAM
	041474	004537	043744		JSR R5,BLOCK3	
	041500	040000			SIZE	
	041502	100000			100000	
	041504	000000			0	
					.DSABL CRF	
6701	041506	104417			KERNEL	;ENTER KERNEL MODE
6702	041510	042737	000040	172516	BIC #BIT5,MMR3	;TURN OFF UNIBUS MAP
6703	041516	013700	002400		MOV NEWBANK,R0	
6704	041522	006200			ASR R0	
6705	041524				ON.ERROR	
	041524	103003				BCC L460
6706	041526	012737	100000	170200	MOV #BIT15,MAPLO	
6707	041534				END ;OF ON.ERROR	
	041534					L460:::~::~
6708	041534	010037	170202		MOV R0,MAPH0	
6709	041540	004737	042026		CALL LOWMAP	;SETUP LOWER 16K IN UNIBUS MAP
6710	041544	052737	000040	172516	BIS #BIT5,MMR3	;ENERGIZE UNIBUS MAP
6711	041552	042737	001001	177572	BIC #BIT9!BIT0,MMR0	;DEENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
6712	041560	004737	042322		CALL NEWKERNEL	
6713	041564	052737	001001	177572	BIS #BIT9!BIT0,MMR0	;ENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
6714	041572				SET RLFLAG	
	041572	012737	177777	002134		MOV #-1,RLFLAG
6715	041600				\$RETURN NOERROR	
	041600	000241				CLC
	041602	000207				RTS PC

6718 041604

UNRELOCATE: SUBTST <<UNRELOCATE PROGRAM>>

 :*SUBTEST UNRELOCATE PROGRAM

```

6719
6720 041604 010046 002532 002106      :RESTORE LOADERS
        041604 013701 002532      PUSH      R0
                                           MOV R0,-(SP)
6721 041606 013701 002532      MOV      LOADBANK,R1
6722 041612 013700 002710      MOV      LOADHOME,R0
6723 041616 004737 042072      CALL     BANKMOV
6724 041622 004737 042370      CALL     NEWLOAD      ;MAP NEW LOADER BANK IN KERNEL SPACE
6725 041626 013746 002106      CALL     BANK
                                           MOV BANK,-(SP)
6726 041632 013737 002532 002106      MOV      LOADBANK,BANK
6727 041640 004737 042422      CALL     EXBANK
6728 041644 013701 002110      MOV      BANKINDEX,R1
6729 041650 042761 100000 003020      BIC      #BIT15,CONFIG+2(R1)  ;CLEAR LOADER FLAG
6730 041656 013737 002710 002106      MOV      LOADHOME,BANK
6731 041664 004737 042422      CALL     EXBANK
6732 041670 013701 002110      MOV      BANKINDEX,R1
6733 041674 042761 020000 003020      BIC      #BIT13,CONFIG+2(R1)  ;INVALIDATE BACKGROUND PATTERN
6734 041702 012637 002106      POP      BANK
                                           MOV (SP)+,BANK
6735
6736      :RESTORE BANK 0
6737 041706 042737 020000 003020      BIC      #BIT13,CONFIG+2
                                           ;INVALIDATE BACKGROUND PATTERN
6738 041714 005037 002400      LET NEWBANK := #0
                                           CLR NEWBANK
6739 041720 004737 042240      CALL     USERMAP      ;MAP NEWBANK TO USER PAR
6740 041724 052737 140000 177776      USER
                                           ;ENTER USER MODE
                                           ;DO IT TO IT!
        041724 052737 140000 177776      BIS      #BIT15!BIT14,PSW
        .DSABL CRF
        BMOV 0,100000,SIZE
                                           ;MOVE PROGRAM
        JSR R5,BLOCK3
        SIZE
        100000
        0
        .DSABL CRF
6741 041732 004537 043744      KERNEL
                                           ;ENTER KERNEL MODE
6742 041732 004537 043744      BIC      #BIT9!BIT0,MMR0
                                           ;DEENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
6743 041746 042737 001001 177572      CALL     NEWKERNEL
6744 041754 004737 042322      BIS      #BIT9!BIT0,MMR0
                                           ;ENERGIZE MEMORY MANAGEMENT - BUT DON'T TRAP
6745 041760 052737 001001 177572      CLR      RLFLAG
6746 041766 005037 002134      BIC      #BIT5,MMR3
                                           ;TURN OFF UNIBUS MAP
6747 041772 042737 000040 172516      CLEAR   MAPLO,MAPHO
6748 042000 005037 170200      CLR     MAPLO
        042000 005037 170202      CLR     MAPHO
6749 042010 004737 042026      CALL     LOWMAP
                                           ;SETUP LOWER 16K OF UNIBUS MAP
6750 042014 052737 000040 172516      BIS      #BIT5,MMR3
                                           ;ENERGIZE UNIBUS MAP
6751 042022 012600      POP      R0
                                           MOV (SP)+,R0
6752 042024 000207      RETURN
6753
6754 042026
    
```

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

 :*SUBTEST SETUP LOWER 16K OF UNIBUS MAP

```
6755 042026          PUSH    R0,R1,R2
      042026 010046
      042030 010146
      042032 010246
6756 042034 012700 170200
6757 042040 012701 170204
6758 042044 012702 000003
6759 042050 012011          1$: MOV    #MAPL0,R0
6760 042052 062721 020000      MOV    #MAPL1,R1
6761 042056 012021          MOV    #3,R2
6762 042060 077205          MOV    (R0)+,(R1)
6763 042062          ADD    #BIT13,(R1)+
      042062 012602          MOV    (R0)+,(R1)+
      042064 012601          SOB   R2,1$
      042066 012600          POP   R2,R1,R0
6764 042070 000207          RETURN

      MOV (SP)+,R2
      MOV (SP)+,R1
      MOV (SP)+,R0
```

6767 042072

BANKMOV:SUBTST <<MOVE BANKS>>

 :*SUBTEST MOVE BANKS

6768					:MOVE 3/4 OF A BANK	
6769					:CALLING SEQUENCE	
6770					:R0 = DESTINATION BANK	
6771					:R1 = SOURCE BANK	
6772	042072	104415			SAVREG	
6773	042074	004737	042240		CALL USERMAP	
6774	042100	104416			RESREG	
6775	042102	104415			SAVREG	
6776	042104	072027	000011		ASH #9,R0	
6777	042110	072127	000011		ASH #9,R1	
6778	042114	012702	177650		MOV #UIPAR4,R2	
6779	042120	012703	000200		MOV #200,R3	
6780						
6781	042124	010122			MOV R1,(R2)+	:MAP 1ST HALF BANK
6782	042126	060301			ADD R3,R1	:BUMP BY 4K
6783	042130	010122			MOV R1,(R2)+	
6784	042132	060301			ADD R3,R1	
6785						
6786	042134	010022			MOV R0,(R2)+	
6787	042136	060300			ADD R3,R0	
6788	042140	010022			MOV R0,(R2)+	
6789	042142	060300			ADD R3,R0	
6790						
6791	042144				USER	
	042144	052737	140000	177776	BIS #BIT15!BIT14,PSW	:DO IT TO IT!
					.DSABL CRF	
6792	042152				BMOV 100000,140000,SIZE/2	:MOV 1ST HALF BANK
	042152	004537	043744		JSR R5,BLOCK3	
	042156	020000			SIZE/2	
	042160	140000			140000	
	042162	100000			100000	
					.DSABL CRF	
6793	042164	104417			KERNEL	:ENTER KERNEL MODE
6794						
6795	042166	012702	177650		MOV #UIPAR4,R2	
6796						
6797	042172	010122			MOV R1,(R2)+	:MAP 2ND HALF BANK
6798	042174	060301			ADD R3,R1	:BUMP BY 4K
6799	042176	010122			MOV R1,(R2)+	
6800	042200	060301			ADD R3,R1	
6801						
6802	042202	010022			MOV R0,(R2)+	
6803	042204	060300			ADD R3,R0	
6804	042206	010022			MOV R0,(R2)+	
6805	042210	060300			ADD R3,R0	
6806						
6807	042212				USER	
	042212	052737	140000	177776	BIS #BIT15!BIT14,PSW	:DO IT TO IT!
					.DSABL CRF	
6808	042220				BMOV 100000,140000,SIZE/4	:MOV 3ND FOURTH OF BANK
	042220	004537	043744		JSR R5,BLOCK3	
	042224	010000			SIZE/4	
	042226	140000			140000	

MOVE BANKS
042230 100000
6809 042232 104417
6810
6811 042234 104416
6812 042236 000207

100000
.DSABL CRF ;ENTER KERNEL MODE
KERNEL
RESREG
RETURN

6815 042240

```

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

6816 042240 012701 177640
 6817 042244 012702 172340
 6818 042250 012703 177600
 6819 042254 012704 172300
 6820 042260 012705 000004
 6821 042264 012221
 6822 042266 011423
 6823 042270 077503
 6824
 6825 042272 013700 002400
 6826 042276 072027 000011
 6827
 6828 042302 012705 000004
 6829 042306 010021
 6830 042310 062700 000200
 6831 042314 011423
 6832 042316 077505
 6833 042320 000207
 6834
 6835 042322

```

NEWKERNEL:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW BANK>>
:*****
:*SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW BANK
:*****
PUSH R0,R1,R5
MOV R0,-(SP)
MOV R1,-(SP)
MOV R5,-(SP)
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
MOV (SP)+,R5
MOV (SP)+,R1
MOV (SP)+,R0
RETURN
    
```

6836 042322
 042322 010046
 042324 010146
 042326 010546
 6837 042330 012700 172340
 6838 042334 013701 002400
 6839 042340 072127 000011
 6840
 6841 042344 012705 000004
 6842 042350 010120
 6843 042352 062701 000200
 6844 042356 077504
 6845 042360
 042360 012605
 042362 012601
 042364 012600
 6846 042366 000207
 6847
 6848 042370

```

NEWLOAD:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
:*****
:*SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK
:*****
;R0 CONTAINS THE DESTINATION BANK
PUSH R0,R1
MOV R0,-(SP)
MOV R1,-(SP)
MOV #KIPAR4,R1
ASH #9.,R0 ;BANK 1 STARTS AT 1000000 LESS 6 LSB'S (1000)
MOV R0,(R1)+ ;SETUP KIPAR4
ADD #200,R0
    
```

6849
 6850 042370
 042370 010046
 042372 010146
 6851 042374 012701 172350
 6852 042400 072027 000011
 6853 042404 010021
 6854 042406 062700 000200

```
6855 042412 010021      MOV    R0,(R1)+      ;SETUP KIPAR5
6856 042414      POP    R1,R0
      042414 012601      MOV    (SP)+,R1
      042416 012600      MOV    (SP)+,R0
6857 042420 000207      RETURN
```

```

6861 042422      EXBANK: SUBTST <<SUBR EXAMINE BANK>>
:*****
:*SUBTEST      SUBR      EXAMINE BANK
:*****
:DOES THE FOLLOWING:
:(1) SETS UP 'BANKINDEX' AND R0 BASED ON VALUE OF 'BANK'
:(2) SETS THE 'MKFLAG' IF THE BANK IS AN MK11
:(3) SETS THE 'KPFLAG' IF THE BANK IS THE PROTECTED REGION OF MK11 MEMORY
:(4) SETS THE 'ACFLAG' IF THE BANK CAN BE ACCESSED BY THIS CPU
:(5) SETS THE 'PFLAG' IF THE BANK IS IN PROGRAM SPACE
:(6) 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)
:   BUT! THE 'RRFLAG' IS ALWAYS SET TO DISABLE TESTING IF FIELD SERVICE MODE
:   'SELECTED BANKS' ARE BEING TESTED AND THIS BANK IS NOT SELECTED.
:(7) 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)
:   PUSH      R0,R1,R2
:
:   MOV R0,-(SP)
:   MOV R1,-(SP)
:   MOV R2,-(SP)
:
:   CLEAR    MKFLAG,KPFLAG
:
:   CLR  MKFLAG
:   CLR  KPFLAG
:
:   SET     ACFLAG
:
:   MOV  #-1,ACFLAG
:
:   CLEAR  PFLAG,RRFLAG,BMFLAG
:
:   CLR  PFLAG
:   CLR  RRFLAG
:   CLR  BMFLAG
:
:   MOV  BANK,R1
:   MUL  #6,R1      ;R0 = BANK * 6
:   MOV  R1,BANKINDEX
:   BIT  #BIT12,CONFIG(R1) ;PROTECTED REGION OF MK11 MEMORY?
:   BEQ  1$        ;NO - SKIP
:   SET  KPFLAG,RRFLAG
:
:   MOV  #-1,KPFLAG
:   MOV  #-1,RRFLAG
:
:1$:  MOV  #MASTER,R0
:   FOR R2 := #0 TO SLAVES
:
:   CLR R2
:
:   IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)
:
:   BIT R0,CPUBIT
:   BEQ L461
:   BIT R0,CONFIG(R1)
:   BNE L461
:
:   CLR ACFLAG
:   END ;OF IF R0
:
:   ASL  R0
:   END ;OF FOR R2
:
:   INC R2
:   CMP R2,SLAVES
:   BLE B63
:
:   B63:
:
:   L461:
:
:   E63:

```

6862
6863
6864
6865
6866
6867
6868
6869
6870
6871
6872
6873
6874
6875
6876
6877
6878
6879
6880
6881
6882
6883
6884
6885
6886
6887
6888
6889
6890
6891
6892

042422
042422 010046
042424 010146
042426 010246
042430
042430 005037 002126
042434 005037 002122
042440
042440 012737 177777 002124
042446
042446 005037 002130
042452 005037 002132
042456 005037 002136
042462 013701 002106
042466 070127 000006
042472 010137 002110
042476 032761 010000 003016
042504 001406
042506
042506 012737 177777 002122
042514 012737 177777 002132
042522 012700 000020
042526
042526 005002
042530
042530
042530 030037 002114
042534 001405
042536 030061 003016
042542 001002
042544 005037 002124
042550
042550
042550 006300
042552
042552 005202
042554 020237 002730
042560 003763
042562

1\$:

B63:.....
L461:.....
E63:.....

```

6893 042562 016100 003016      2$:  MOV    CONFIG(R1),R0
6894 042566 042700 117777      BIC    #^C60000,R0          ;GET MEMORY TYPE BITS
6895 042572 022700 020000      CMP    #20000,R0          ;IS IT AN MK11?
6896 042576 001003      BNE    3$                  ;NO - SKIP
6897 042600      SET    MKFLAG
        042600 012737 177777 002126      MOV    #-1,MKFLAG
6898 042606 032761 100000 003016  3$:  BIT    #BIT15,CONFIG(R1)   ;BANK = PROGRAM SPACE?
6899 042614 001406      BEQ    5$                  ;NO - SKIP
6900 042616      SET    PFLAG,RRFLAG
        042616 012737 177777 002130      MOV    #-1,PFLAG
        042624 012737 177777 002132      MOV    #-1,RRFLAG
6901 042632 005737 002134      5$:  TST    RLFLAG              ;IS PROGRAM RELOCATED?
6902 042636 001402      BEQ    6$                  ;NO - SKIP
6903 042640 005137 002132      COM    RRFLAG              ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
6904 042644 032761 004000 003016  6$:  BIT    #BIT11,CONFIG(R1)   ;ERRORS PRESENT IN THIS BANK?
6905 042652 001403      BEQ    8$                  ;NO - SKIP
6906 042654      SET    BMFLAG
        042654 012737 177777 002136      MOV    #-1,BMFLAG
6907 042662 005737 002712      8$:  TST    WORST              ;IS THIS A WORST FIRST PASS?
6908 042666 001002      BNE    9$                  ;YES - SKIP
6909 042670 005137 002136      COM    BMFLAG              ;NO - COMPLEMENT BAD MEMORY FLAG
6910 042674      9$:  IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
        042674 005737 002002      TST    SELONLY
        042700 001407      BEQ    L462
        042702 032761 040000 003020      BIT    #BIT14,CONFIG+2(R1)
        042710 001003      BNE    L462
6911 042712      SET    RRFLAG
        042712 012737 177777 002132      MOV    #-1,RRFLAG
6912 042720      END ;OF IF SELONLY
        042720
6913 042720      POP    R2,R1,R0
        042720 012602      MOV    (SP)+,R2
        042722 012601      MOV    (SP)+,R1
        042724 012600      MOV    (SP)+,R0
6914 042726 000207      RETURN
  
```

6918 042730

```
BANKOK: SUBTST <<SUBR BANK OK?>>
:*****
:*SUBTEST SUBR BANK OK?
:*****
:TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
:IS OF THE TYPE WE ARE TESTING 'TMFLAG'.
:RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).
MOV TMFLAG,R0
COM R0
MOV MKFLAG,R1
XOR R0,R1
RETURN ;OK = (=OK)
```

6919
6920
6921
6922 042730 013700 002140
6923 042734 005100
6924 042736 013701 002126
6925 042742 074001
6926 042744 000207
6927
6928 042746
6929 042746

```
INCRPT:  
INCPAT: SUBTST <<SUBR INCREMENT PATTERN TESTING MK11'S>>
:*****
:*SUBTEST SUBR INCREMENT PATTERN TESTING MK11'S
:*****
:INCREMNT THE PATTERN & SET UP THE CONDITION CODES
:RESULT - Z BIT SET INDICATES OVERFLOW
INC PATTERN
CMP #40,PATTERN ;SET UP CONDITION CODES
RETURN ;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)
```

6930
6931
6932 042746 005237 002120
6933 042752 022737 000040 002120
6934 042760 000207
6935
6936 042762
6937 042762

```
SETPAT:  
HIPAT: SUBTST <<SUBR SET HIGHEST PATTERN TESTING TYPE>>
:*****
:*SUBTEST SUBR SET HIGHEST PATTERN TESTING TYPE
:*****
MOV #37,PATTERN ;SET HIGHEST PATTERN
RETURN
```

6938 042762 012737 000037 002120
6939 042770 000207
6940
6941 042772

```
INCBNK: SUBTST <<SUBR INCREMENT BANK & TEST>>
:*****
:*SUBTEST SUBR INCREMENT BANK & TEST
:*****
:RESULTS RETURNED IN CONDITION CODES
INC BANK
CMP #120.,BANK ;TOO FAR?
RETURN
```

6942
6943 042772 005237 002106
6944 042776 022737 000170 002106
6945 043004 000207

6948 043006

```
INCMAR: SUBTST <<SUBR INCREMENT MARGINS & TEST>>
:*****
:*SUBTEST SUBR INCREMENT MARGINS & TEST
:*****
```

6949
 6950
 6951
 6952 043006 032777 004000 137752
 043006 001004
 043014 005737 002440
 043016 001001
 6953 043024
 043024 000410
 6954 043026
 043026

```
:SKIPS THE SPECIAL CASE (MARGIN CANNOT = 1)
:RETURNS THE RESULT OF MARGIN OVERFLOW IN THE CONDITION CODES
.ENABL LSB
IF #SW11 OFF.IN @SWR AND QVFLAG IS FALSE
BIT #SW11,@SWR
BNE L463
TST QVFLAG
BNE L463
```

6955 043026 013737 002112 002376
 6956 043034 005037 002112
 6957 043040 004737 043300
 6958 043044 000435
 6959 043046 013737 002112 002376 1\$:
 6960 043054 005237 002112
 6961 043060 022737 000001 002112
 6962 043066 001002
 6963 043070 005237 002112
 6964 043074

```
GOTO 1$
END ;OF IF #SW11
L463:::
MOV MARGIN,OLDMARGIN
CLR MARGIN ;YES - SET MARGIN TO ZERO
CALL COREHISTORY
BR 3$
MOV MARGIN,OLDMARGIN
INC MARGIN
CMP #1,MARGIN ;SPECIAL CASE?
BNE 2$ ;NO - SKIP
INC MARGIN ;YES - SET MARGIN TO '2'
```

6964 043074 005737 002126
 043100 001005
 043102 023727 002112 000006
 043110 001401
 6965 043112
 043112 000412
 6966 043114
 043114
 6967 043114
 043114 005737 002126
 043120 001405
 043122 023727 002112 000004
 043130 001401
 6968 043132
 043132 000402
 6969 043134
 043134

```
IF MKFLAG IS FALSE AND MARGIN NE #6
TST MKFLAG
BNE L464
CMP MARGIN,#6
BEQ L464
GOTO 3$
BR 3$
END ;OF IF MKFLAG
L464:::
IF MKFLAG IS TRUE AND MARGIN NE #4
TST MKFLAG
BEQ L465
CMP MARGIN,#4
BEQ L465
GOTO 3$
BR 3$
END ;OF IF MKFLAG
L465:::
CLR MARGIN ;YES - SET MARGIN TO ZERO
CALL COREHISTORY ;SET CONDITION CODE - DONE IF Z BIT SET
TST MARGIN
RETURN
.DSABL LSB
```

6970 043134 005037 002112
 6971 043140 004737 043300 3\$:
 6972 043144 005737 002112
 6973 043150 000207
 6974
 6975
 6976 043152

```
DECMAR: SUBTST <<DECREMENT MARGINS & TEST>>
:*****
:*SUBTEST DECREMENT MARGINS & TEST
:*****
```

6977
 6978
 6979 043152 013737 002112 002376
 6980 043160 005337 002112

```
:SKIPS THE SPECIAL CASE (MARGIN CANNOT = 1)
:RETURNS THE RESULT OF MARGIN UNDERFLOW IN THE CONDITION CODES
MOV MARGIN,OLDMARGIN
DEC MARGIN
```

6981 043164 022737 000001 002112
6982 043172 001002
6983 043174 005337 002112
6984 043200 004737 043300
6985 043204 005737 002112
6986 043210 000207

1\$:

CMP #1,MARGIN
BNE 1\$
DEC MARGIN
CALL COREHISTORY
TST MARGIN
RETURN

:SPECIAL CASE?
:NO - SKIP
:YES - SET MARGIN TO '0'
:SETUP CONDITION CODES - NEGATIVE IS UNDERFLOW

```

6989 043212
6990
6991
6992 043212 032777 004000 137546
    043212 001003
    043220 005737 002440
    043222 001401
    043226
    043230
6993 043230 000420
    043230
6994 043232
    043232
6995 043232 013737 002112 002376
6996 043240 005737 002126
    043240 001004
    043244 012737 000005 002112
6997 043246
6998 043254 000403
    043254
    043256
6999 043256 012737 000003 002112
7000 043264
    043264
7001 043264 004737 043300
7002 043270 000402
7003 043272 005037 002112
7004 043276 000207
7005
7006 043300
7007 043300 005737 002126
    043300 001041
    043304 005737 002436
    043306 001036
    043312
7008 043314
    043314 023737 002376 002112
    043322 001432
7009 043324
    043324 023727 002376 000004
    043332 001414
    043334 023727 002376 000005
    043342 001410
    043344 023727 002112 000004
    043352 001404
    043354 023727 002112 000005
    043362 001012
    043364
7010 043364 013700 002112
7011 043370 006300
7012 043372 010037 177750
    
```

```

SETMAR: SUBTST <<SUBR SET HIGHEST MARGIN>>
:*****
:*SUBTEST SUBR SET HIGHEST MARGIN
:*****
;RECOGNIZES THE "INHIBIT MARGINS SWITCH"
.ENABL LSB
IF #SW11 SET. IN @SWR OR QVFLAG IS TRUE
; BIT #SW11,@SWR
; BNE L466
; TST QVFLAG
; BEQ L467
L466:;:;:;:;
; BR 1$
L467:;:;:;:;
MOV MARGIN,OLDMARGIN
IF MKFLAG IS FALSE
; TST MKFLAG
; BNE L470
MOV #5,MARGIN ;HIGHEST MJ11 MARGIN
ELSE
; BR L471
L470:;:;:;:;
MOV #3,MARGIN ;HIGHEST MK11 MARGIN
END ;OF IF MKFLAG
L471:;:;:;:;
CALL COREHISTORY
BR 2$ ;SKIP
1$: CLR MARGIN ;SET MARGIN TO '0'
2$: RETURN
.DSABL LSB
COREHISTORY:SUBTST <<SUBR REWRITE CORE HISTORY>>
:*****
:*SUBTEST SUBR REWRITE CORE HISTORY
:*****
IF MKFLAG IS FALSE AND NULLFLAG IS FALSE
; TST MKFLAG
; BNE L472
; TST NULLFLAG
; BNE L472
IF OLDMARGIN NE MARGIN
; CMP OLDMARGIN,MARGIN
; BEQ L473
IF OLDMARGIN EQ #4 OR OLDMARGIN EQ #5 OR MARGIN EQ #4 OR MARGIN EQ #5
; CMP OLDMARGIN,#4
; BEQ L474
; CMP OLDMARGIN,#5
; BEQ L474
; CMP MARGIN,#4
; BEQ L474
; CMP MARGIN,#5
; BNE L475
L474:;:;:;:;
MOV MARGIN,R0
ASL R0
MOV R0,MAINT ;SET MARGINS
    
```

```
7013
7014 043376          LET R2 := ONES          MOV ONES,R2
      043376 013702 002726
7015 043402 004737 035344          CALL      BACKGND
7016 043406 104511          INVALIDATE
7017 043410          END ;OF IF
      043410
7018 043410          END ;OF IF
      043410
7019 043410          END ;OF IF
      043410
7020 043410 005037 002436          CLR      NULLFLAG
7021 043414 000207          RETURN

L475:.....
L473:.....
L472:.....
```

```

7024 043416
7025
7026
7027
7028
7029
7030
7031
7032 043416 104472
7033 043420 012737 043460 000004
7034 043426 005037 177750
7035 043432 005737 002134
       043436 001402
       043440 004737 041604
       043444
7036 043444 004737 024754
7037 043450 104421
7038 043452 042737 000040 172516
7039 043460 005001
7040 043462 000005
7041 043464 012700 177406
7042 043470 010160 000004
7043 043474 012710 177400
7044 043500 012740 000005
7045 043504 105710
7046 043506 100376
7047 043510 062701 020000
7048 043514 005710
7049 043516 100761
7050 043520 005007
    
```

```

BOOT:  SUBTST <<BOOTSTRAP ROUTINE>>
:*****
:*SUBTEST  BOOTSTRAP ROUTINE
:*****
:INITIALIZE ALL CSR'S
:CLEAR ANY MARGINS
:UNRELOCATE IF NECESSARY
:FLUSH OUT ANY DBE'S
:TURN OFF MEMORY MANAGEMENT
:TURN OFF THE UNIBUS MAP
:BOOT RK0 OR RK1
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
SET4 #BOOT1 ;TRAPS TO 4 GOTO BOOT1
MOV #BOOT1,4
.DSABL CRF
CLR MAINT
IF RLFLAG IS TRUE THEN $CALL UNRELOCATE

                                TST RLFLAG
                                BEQ L476
                                JSR PC,UNRELOCATE
                                L476:::
CALL MT0030 ;FLUSH OUT DBE'S
DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
BIC #BIT5,MMR3 ;TURN OFF THE UNIBUS MAP
BOOT1: CLR R1
1$: RESET
MOV #177406,R0
MOV R1,4(R0)
MOV #177400,(R0)
MOV #5,-(R0)
2$: TSTB (R0)
BPL 2$
ADD #BIT13,R1
TST (R0)
BMI 1$
CLR PC
    
```

7053 043522
7054 043522 004737 043554
7055 043526 005737 002444
043532 001003
043534 005737 002442
043540 001402
043542
7056 043542 000777
7057 043544
043544 000403
043546
7058 043546 000000
7059 043550 000137 004352
7060 043554
043554
7061
7062 043554

7063
7064
7065
7066
7067
7068
7069
7070
7071 043554 104472
7072 043556 005037 177750
7073 043562
043562 005737 002134
043566 001402
043570 004737 041604
043574
7074 043574
043574 005737 002536
043600 001002
7075 043602 004737 024754
7076 043606
043606
7077 043606 012700 000001
7078 043612 013701 002710
7079 043616 004737 042072
7080 043622 104421
7081 043624 042737 000040 172516
7082 043632 005037 002670
7083 043636 000207
7084
7085 043640

```
EXIT: SUBTST <<HALT PROGRAM>>
:*****
:*SUBTEST HALT PROGRAM
:*****
CALL SHUTUP
EXIT2: IF APTFLAG IS TRUE OR ACTFLAG IS TRUE

TST APTFLAG
BNE L477
TST ACTFLAG
BEQ L500
L477:*****

BR
ELSE

BR L501
L500:*****

$EXHALT: HALT
JMP START
END ;OF IF APTFLAG

L501:*****

SHUTUP: SUBTST <<SHUTDOWN DIAGNOSTIC>>
:*****
:*SUBTEST SHUTDOWN DIAGNOSTIC
:*****
:INITIALIZE ALL CSR'S
:CLEAR MARGINS
:UNRELOCATE
:FLUSH OUT DBE'S
:RESTORE LOADERS
:TURN OFF MEMORY MANAGEMENT
:UNMAP THE UNIBUS MAP
:CLEAR MULTIPROCESSING LOCK
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
CLR MAINT
IF RLFLAG IS TRUE THEN $CALL UNRELOCATE

TST RLFLAG
BEQ L502
JSR PC,UNRELOCATE
L502:*****

IF QUICK IS FALSE

TST QUICK
BNE L503
L503:*****

CALL MT0030 ;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
BIC #BITS,MMR3 ;TURN OFF UNIBUS MAP
CLR LOCK
RETURN

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

```
APT SHUTDOWN SEQUENCE

7086 043640          MAP      #0          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
      043640 010346          MOV      #0,R3          MOV R3,-(SP)
      043642 012703 000000    CALL     MAPPER
      043646 004737 040700    .DSABL   CRF

      043652 012603          SUPERVISOR          ;ENTER SUPERVISOR MODE
7087 043654          BIS      #BIT14,PSW          ;DO IT TO IT!
      043654 052737 040000 177776 .DSABL   CRF
      043662 012737 043640 060024    MOV      #APTDOWN,FIRST+24
7088 043662 012737 000340 060026    MOV      #340,FIRST+26
7089 043670 012737 000000 123640    MOV      #0,FIRST+APTDOWN
7090 043676 012737 000000 123640    MOV      #0,FIRST+APTDOWN
7091 043704 104417          KERNEL          ;ENTER KERNEL MODE
7092 043706 000000          APTHLT: HALT
```

7095 043710

SUBTST <<BLOCK MOVE SUBROUTINE>>
 :*****
 :*SUBTEST BLOCK MOVE SUBROUTINE
 :*****

7096
 7097
 7098
 7099
 7100
 7101
 7102
 7103
 7104
 7105
 7106
 7107
 7108
 7109
 7110
 7111
 7112
 7113
 7114
 7115
 7116
 7117
 7118
 7119
 7120

043710 010046
 043712 010146
 043714 010246
 043716 012702 177640
 043722 012701 000020
 043726 000413
 043730 010046
 043732 010146
 043734 010246
 043736 012701 000020
 043742 000404
 043744 010046
 043746 010146
 043750 010246
 043752 012501
 043754 012502
 043756 012500
 043760 012022
 043762 077102
 043764 012602
 043766 012601
 043770 012600
 043772 000205

```

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

7122
 7123
 7124 043774

 7125 043774 104415
 7126 043776 104401 112170

 7127
 7128 044002 005737 002134
 044006 001003
 044010 005737 002526
 044014 001404
 044016
 7129 044016 104401 114303
 044016

 7130 044022 104416
 7131 044024 000207
 7132 044026
 044026
 7133 044026 013746 177746
 044026 013746 002256
 044032 013746 002010
 044036
 7134 044042 104424
 7135 044044 012737 177777 002010
 044044
 7136 044052 104401 113371
 044052

 7137 044056 104414
 7138 044060 012600
 044060 020027 000024
 7139 044062 101403
 7140 044066
 7141 044070 104401 112224
 044070

 7142 044074 000766
 7143 044076 010046
 044100 006316
 044102 004737 044160

 7144 044106 044170
 7145 044110 044256
 7146 044112 044346
 7147 044114 044520
 7148 044116 044764
 7149 044120 045274
 7150 044122 046152
 7151 044124 046160
 7152 044126 046510
 7153 044130 047104

```

.SBTTL FIELD SERVICE MODE
FIELDSERVICE:SUBTST <<SUBR FIELD SERVICE COMMAND MODE>>
:*****
:*SUBTEST SUBR FIELD SERVICE COMMAND MODE
:*****
SAVREG
TYPE MSG020 ;FIELD SERVICE COMMAND MODE
TYPEIT ,MSG020
.DSABL CRF

IF RLFLAG IS TRUE OR NOFSMODE IS TRUE
TST RLFLAG
BNE L504
TST NOFSMODE
BEQ L505
L504:::
;NOT AVAILABLE NOW - TRY LATER!

TYPE MSG048
TYPEIT ,MSG048
.DSABL CRF
RESREG
RETURN
END ;OF IF RLFLAG
L505:::
PUSH CONTRL,CSRNO,KAMIKAZE ;SAVE CACHE STATUS
MOV CONTRL,-(SP)
MOV CSRNO,-(SP)
MOV KAMIKAZE,-(SP)

CACHOFF
SET KAMIKAZE ;TURN CACHE OFF

FS1: TYPE MSG026 ;COMMAND:
TYPEIT ,MSG026
.DSABL CRF
RDDEC ;READ A DECIMAL NUMBER
POP R0 ;COMMAND --> R0
MOV (SP)+,R0

CMP R0,#20.
BLOS 1$
TYPE MSG021
TYPEIT ,MSG021
.DSABL CRF
BR FS1

1$: CASE R0
MOV R0,-(SP)
ASL @SP
JSR PC,L506

FSCMD0 ;EXIT FIELD SERVICE COMMANDS
FSCMD1 ;READ CSR
FSCMD2 ;LOAD CSR
FSCMD3 ;EXAMINE MEMORY
FSCMD4 ;MODIFY MEMORY
FSCMD5 ;SELECT BANK,MARGIN, & PATTERN
FSCMD6 ;TYPE CONFIGURATION MAP
FSCMD7 ;BATTERY BACKUP - P/F TEST
FSCMD8 ;SOB-A-LONG TEST
FSCMD9 ;SUPER TIGHT SCOPE LOOP
    
```

7154	044132	047550	FCMD10	:ERROR SUMMARY
7155	044134	047734	FCMD11	:REFRESH TEST
7156	044136	050330	FCMD12	:SET FILL COUNT
7157	044140	050356	FCMD13	:ENTER KAMIKAZE MODE
7158	044142	050400	FCMD14	:EXIT KAMIKAZE MODE
7159	044144	050420	FCMD15	:TURN CACHE OFF
7160	044146	050436	FCMD16	:TURN CACHE ON
7161	044150	050454	FCMD17	:RUN ONLY MULTIPOINT TESTS
7162	044152	050470	FCMD18	:RESUME RUNNING BOTH SINGLE & MULTIPOINT TESTS
7163	044154	050502	FCMD19	:TEST ONLY SELECTED BANKS
7164	044156	050566	FCMD20	:RESUME TESTING ALL BANKS
7165	044160		END ;OF CASE	

	044160			L506:::~::~
	044160	062616		ADD (SP)+,@SP
	044162	013646		MOV @ (SP)+,@-(SP)
	044164	004736		JSR PC,@(SP)+
7166	044166	000731	BR FS1	

```
7169 044170 FSCMD0: SUBTST <<COMMAND 0 EXIT>>
:*****
:*SUBTEST COMMAND 0 EXIT
:*****
7170 044170 TYPE MSG103 ;LEAVING FIELD SERVICE MODE
044170 104401 116364 TYPEIT ,MSG103
7171 044174 062706 000002 .DSABL CRF
7172 044200 005737 002012 ADD #2,SP
044204 001405 IF SKIPKAMI IS TRUE TST SKIPKAMI
7173 044206 062706 000002 ADD #2,SP ;THROW AWAY OLD KAMIKAZE FLAG
7174 044212 005037 002012 CLR SKIPKAMI BEQ L507
7175 044216 000402 ELSE BR L510
044220 L507:*****
7176 044220 POP KAMIKAZE ;RESTORE OLD KAMIKAZE FLAG
044220 012637 002010 MOV (SP)+,KAMIKAZE
7177 044224 END ;OF IF SKIPKAMI L510:*****
7178 044224 POP CSRNO MOV (SP)+,CSRNO
7179 044230 IF CACHK EQ #BIT0!BIT2!BIT3!BIT8!BIT9 ;IF CACHE IS OFF
044230 023727 002700 001415 CMP CACHK,#BIT0!BIT2!BIT3!BI
044236 001003 BNE L511
7180 044240 062706 000002 ADD #2,SP ;THROW AWAY CACHE STATUS
7181 044244 000402 ELSE BR L512
044246 L511:*****
7182 044246 POP CONTRL ;RESTORE CACHE STATUS
044246 012637 177746 MOV (SP)+,CONTRL
7183 044252 END ;OF IF CACHK L512:*****
7184 044252 104416 RESREG
7185 044254 000207 RETURN
7186
7187 044256 FSCMD1: SUBTST <<FS COMMAND 1 READ CSR>>
:*****
:*SUBTEST FS COMMAND 1 READ CSR
:*****
7188 044256 004737 050630 CALL WHICHCSR
7189 044262 010637 002374 MOV SP,FSSTACK
7190 044266 012737 044322 000004 SET4 #1$ ;TRAPS TO 4 GOTO 1$
044266 MOV #1$,4
.DSABL CRF
7191 044274 104426 READCSR
7192 044276 012737 177777 002530 SET NOERROR MOV #-1,NOERROR
7193 044304 104026 ERROR +26 ;USE ERROR ROUTINE FOR PRINTOUT
7194 044306 012737 036726 000004 RES4 ;RESET TRAPS TO 4 TO DEFAULT
044306 005037 177766 MOV #TIMEOUT,4
044314 CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
;THAT A EXPECTED TRAP COULD HAVE SET
.DSABL CRF
7195 044320 000207 RETURN
7196 044322 104401 113340 1$: TYPE MSG025 ;THIS CSR DOES NOT EXIST
TYPEIT ,MSG025
```

7197 044326 013706 002374
7198 044332 012737 036726 000004
044332 005037 177766
044340

7199 044344 000207

.DSABL CRF
MOV FSSTACK,SP
RES4
MOV #TIMEOUT,4
CLR CPUERR

.DSABL CRF
RETURN

:RESET TRAPS TO 4 TO DEFAULT
:CLEAR OUT THE CPU ERROR REGISTER BITS
:THAT A EXPECTED TRAP COULD HAVE SET

```

7202 044346          FSCMD2: SUBTST <<FS  COMMAND 2  LOAD CSR>>
:*****
:*SUBTEST          FS  COMMAND 2  LOAD CSR
:*****
7203 044346 004737 050630          CALL  WHICHCSR
7204 044352 010637 002374          MOV   SP,FSSTACK
7205 044356 012737 044474 000004    SET4  #1$          ;TRAPS TO 4 GOTO 1$
          044356          MOV   #1$,4
          .DSABL CRF
7206 044364 104426          READCSR
7207 044366 104401 113403          TYPE  MSG027
          044366          TYPEIT ,MSG027
          .DSABL CRF
7208 044372          SET   NOERROR
          044372 012737 177777 002530    MOV  #-1,NOERROR
7209 044400 104026          ERROR +26          ;USE ERROR ROUTINE FOR PRINTOUT
7210 044402 012737 036726 000004    RES4          ;RESET TRAPS TO 4 TO DEFAULT
          044402          MOV   #TIMEOUT,4
          044410 005037 177766          CLR   CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
          .DSABL CRF          ;THAT A EXPECTED TRAP COULD HAVE SET
7211 044414          TYPE  MSG023
          044414 104401 113274          TYPEIT ,MSG023
          .DSABL CRF          ;FIRST CSR WORD
7212 044420 104413          RDOCT
7213 044422 012637 002144          POP   CSR          ;READ AN OCTAL NUMBER
          044422          TYPE  MSG024          ;PUT IN IN LOC 'CSR'
          044426 104401 113316          TYPEIT ,MSG024          MOV (SP)+,CSR
          .DSABL CRF          ;2ND CSR WORD
7215 044432 104413          RDOCT
7216 044434 012637 002146          POP   CSR+2          ;READ AN OCTAL NUMBER
          044434          SET   TWOCRS          ;PUT IN IN LOC 'CSR+2'
          044440 012737 177777 002600    MOV (SP)+,CSR+2
          044440          CLR   TWOCRS          MOV #-1,TWOCRS
7218 044446 104425          LOADCSR
7219 044450 005037 002600          CLR   TWOCRS
7220 044454 104426          READCSR
7221 044456 104401 113420          TYPE  MSG028
          044456          TYPEIT ,MSG028
          .DSABL CRF
7222 044462          SET   NOERROR
          044462 012737 177777 002530    MOV  #-1,NOERROR
7223 044470 104026          ERROR +26          ;USE FOR PRINTOUT - NOT AN ERROR
7224 044472 000207          RETURN
7225 044474 104401 113340          TYPE  MSG025          ;THIS CSR DOES NOT EXIST
          044474          TYPEIT ,MSG025
          .DSABL CRF
7226 044500 013706 002374          MOV   FSSTACK,SP
7227 044504 012737 036726 000004    RES4          ;RESET TRAPS TO 4 TO DEFAULT
          044504          MOV   #TIMEOUT,4
          044512 005037 177766          CLR   CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
          .DSABL CRF          ;THAT A EXPECTED TRAP COULD HAVE SET
7228 044516 000207          RETURN
    
```

```

7231 044520 FSCMD3: SUBTST <<FS COMMAND 3 EXAMINE MEMORY>>
:*****
:*SUBTEST FS COMMAND 3 EXAMINE MEMORY
:*****
7232 044520 PUSH BANK,NOPAR,PARTHERE,4
044520 013746 002106 MOV BANK,-(SP)
044524 013746 002100 MOV NOPAR,-(SP)
044530 013746 002372 MOV PARTHERE,-(SP)
044534 013746 000004 MOV 4,-(SP)
7233 044540 012737 000002 002100 MOV #2,NOPAR ;INDICATE PARITY ACTION
7234 044546 104401 113434 TYPE MSG029 ;EXAMINE MEMORY
TYPEIT ,MSG029
.DSABL CRF
7235 044552 104401 113473 1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-16777776)??
TYPEIT ,MSG031
.DSABL CRF
7236 044556 104413 RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
7237 044560 013737 074434 002106 MOV $HIOCT,BANK ;PUT MSB'S IN BANK
7238 044566 012600 POP R0 ;PUT LSB'S IN R0
044566 012600 MOV (SP)+,R0
7239 044570 000241 CLC
7240 044572 006100 ROL R0
7241 044574 006137 002106 ROL BANK
7242 044600 000241 CLC
7243 044602 006000 ROR R0
7244 044604 IF BANK GT #167 THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
044604 023727 002106 000167 CMP BANK,#167
044612 003357 BGT 1$
7245 044614 062700 060000 ADD #FIRST,R0
7246 044620 IF #BIT0 SET.IN R0 THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
044620 032700 000001 BIT #BIT0,R0
044624 001352 BNE 1$
7247 044626 IF R0 HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER 16K
044626 020027 157776 CMP R0,#LAST
044632 101347 BHI 1$
7248 044634 012737 044706 002372 MOV #3$,PARTHERE ;INCASE OF ABORTS
7249 044642 012737 044714 000004 SET4 #4$ ;TRAPS TO 4 GOTO 4$
MOV #4$,4
.DSABL CRF
7250 044650 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
044650 010346 MOV R3,-(SP)
044652 013703 002106 MOV BANK,R3
044656 004737 040700 CALL MAPPER
.DSABL CRF
7251 044662 012603 MOV (SP)+,R3
7251 044664 SUPERVISOR ;ENTER SUPERVISOR MODE
044664 052737 040000 177776 BIS #BIT14,PSW ;DO IT TO IT!
.DSABL CRF
7252 044672 011001 MOV (R0),R1
7253 044674 104417 KERNEL ;ENTER KERNEL MODE
7254 044676 TYPOCS R1
044676 010146 MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
044700 104403 TYPOS ;GO TYPE--OCTAL ASCII
044702 006 ;TYPE 6 DIGITS
044703 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
.DSABL CRF
7255 044704 000410 BR EXCMD3
  
```

```

7256
7257 044706 104401 113533 3$: TYPE MSG032 ;PARITY ABORT
      044706 .DSABL ,MSG032
      BR CRF EXCMD3
7258 044712 000405
7259
7260 044714 062706 000004 4$: ADD #4,SP ;FIX STACK
7261 044720 104401 113552 TYPE MSG033 ;TIMEOUT TRAP
      044720 .DSABL ,MSG033
      BR CRF EXCMD3
7262 044724 000400
7263
7264 044726 104417 EXCMD3: KERNEL ;ENTER KERNEL MODE
7265 044730 POP 4, PARTHERE, NOPAR, BANK
      044730 012637 000004 MOV (SP)+, 4
      044734 012637 002372 MOV (SP)+, PARTHERE
      044740 012637 002100 MOV (SP)+, NOPAR
      044744 012637 002106 MOV (SP)+, BANK
7266 044750 RES4 ;RESET TRAPS TO 4 TO DEFAULT
      044750 012737 036726 000004 MOV #TIMEOUT, 4
      044756 005037 177766 CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
      ;THAT A EXPECTED TRAP COULD HAVE SET
      .DSABL CRF
7267 044762 000207 RETURN
  
```

```

7270 044764          FSCMD4: SUBTST <<FS  COMMAND 4  MODIFY MEMORY>>
:*****
:*SUBTEST          FS  COMMAND 4  MODIFY MEMORY
:*****
7271 044764          PUSH  BANK,NOPAR,PARTHERE,4
044764 013746 002106          MOV  BANK,-(SP)
044770 013746 002100          MOV  NOPAR,-(SP)
044774 013746 002372          MOV  PARTHERE,-(SP)
045000 013746 000004          MOV  4,-(SP)
7272 045004 012737 000003 002100  MOV  #3,NOPAR          ;INDICATE PARITY ACTION
7273 045012 104401 113712          TYPE  MSG036          ;MODIFY MEMORY
045012 104401 113712          TYPEIT ,MSG036
.DSABL CRF
7274 045016 104401 113473          1$: TYPE  MSG031          ;PHYSICAL ADDRESS (0-1677776)??
045016 104401 113473          TYPEIT ,MSG031
.DSABL CRF
7275 045022 104413          RDOCT          ;READ OCTAL NUMBER ONTO STACK & $HIOCT
7276 045024 013737 074434 002106  MOV  $HIOCT,BANK    ;PUT MSB'S IN BANK
7277 045032 012600          POP  R0            ;PUT LSB'S IN R0
045032 012600          CLC              MOV  (SP)+,R0
7278 045034 000241          ROL  R0
7279 045036 006100          ROL  BANK
7280 045040 006137 002106          CLC
7281 045044 000241          ROR  R0
7282 045046 006000          IF BANK GT #167 THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
7283 045050          CMP  BANK,#167
045050 023727 002106 000167          BGT  1$
045056 003357
7284 045060 062700 060000          ADD  #FIRST,R0
7285 045064 032700 000001          IF #BIT0 SET.IN R0 THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
045064 032700 000001          BIT  #BIT0,R0
045070 001352          BNE  1$
7286 045072          IF R0 HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER 16K
045072 020027 157776          CMP  R0,#LAST
045076 101347          BHI  1$
7287 045100 012737 045146 002372  MOV  #3$,PARTHERE    ;INCASE OF ABORTS
7288 045106 012737 045154 000004  SET4  #4$          ;TRAPS TO 4 GOTO 4$
045106 012737 045154 000004  MOV  #4$,4
.DSABL CRF
7289 045114          MAP  BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
045114 010346          MOV  R3,-(SP)
045116 013703 002106          MOV  BANK,R3
045122 004737 040700          CALL MAPPER
.DSABL CRF
045126 012603          MOV  (SP)+,R3
7290 045130 104511          INVALIDATE
7291 045132 052737 040000 177776  SUPERVISOR          ;ENTER SUPERVISOR MODE
045132 052737 040000 177776  BIS  #BIT14,PSW      ;DO IT TO IT!
.DSABL CRF
7292 045140 011001          MOV  (R0),R1
7293          ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
7294 045142 104417          KERNEL          ;ENTER KERNEL MODE
7295 045144 000410          BR   5$
7296
7297 045146 104401 113533          3$: TYPE  MSG032          ;PARITY ABORT
045146 104401 113533          TYPEIT ,MSG032
.DSABL CRF

```

```

7298 045152 000431          BR      EXCMD4          ;EXIT
7299
7300 045154 062706 000004    4$:  ADD      #4,SP          ;FIX STACK
7301 045160 104401 113552    TYPE    MSG033        ;TIMEOUT TRAP
      045160 104401 113552    TYPEIT  ,MSG033
      .DSABL CRF
      BR      EXCMD4          ;EXIT
7302 045164 000424
7303
7304 045166 104401 113731    5$:  TYPE    MSG037        ;OLD DATA WAS
      045166 104401 113731    TYPEIT  ,MSG037
      .DSABL CRF
      TYPOCS R1          ;PRINT IT
7305 045172 010146    MOV      R1,-(SP)      ;;SAVE R1 FOR TYPEOUT
      045172 010146    TYPOS          ;;GO TYPE--OCTAL ASCII
      045174 104403    .BYTE    6          ;;TYPE 6 DIGITS
      045176 006          .BYTE    0          ;;SUPPRESS LEADING ZEROS
      045177 000          .DSABL CRF
7306 045200 104401 113766    TYPE    MSG039        ;INPUT NEW DATA
      045200 104401 113766    TYPEIT  ,MSG039
      .DSABL CRF
7307 045204 104413          RDOCT          ;READ ON OCTAL NUMBER ONTO THE STACK
7308 045206 012601          POP      R1         ;GET NEW NUMBER
7309 045210 052737 040000 177776 SUPERVISOR ;ENTER SUPERVISOR MODE
      045210 052737 040000 177776 BIS      #BIT14,PSW ;DO IT TO IT!
      .DSABL CRF
7310 045216 010110    MOV      R1,(R0)      ;PUT IT IN MEMORY
7311 045220 011001    MOV      (R0),R1     ;READ IT AGAIN
7312 045222 104417    KERNEL          ;ENTER KERNEL MODE
7313 045224 104401 113750    TYPE    MSG038        ;DATA IS NOW
      045224 104401 113750    TYPEIT  ,MSG038
      .DSABL CRF
7314 045230 010146    TYPOCS R1          ;PRINT IT
      045230 010146    MOV      R1,-(SP)      ;;SAVE R1 FOR TYPEOUT
      045232 104403    TYPOS          ;;GO TYPE--OCTAL ASCII
      045234 006          .BYTE    6          ;;TYPE 6 DIGITS
      045235 000          .BYTE    0          ;;SUPPRESS LEADING ZEROS
      .DSABL CRF
7315
7316 045236 104417    EXCMD4: KERNEL ;ENTER KERNEL MODE
7317 045240 012637 000004    POP      4,PARTHERE,NOPAR,BANK
      045240 012637 000004    MOV      (SP)+,4
      045244 012637 002372    MOV      (SP)+,PARTHERE
      045250 012637 002100    MOV      (SP)+,NOPAR
      045254 012637 002106    MOV      (SP)+,BANK
7318 045260 012737 036726 000004 RES4          ;RESET TRAPS TO 4 TO DEFAULT
      045260 012737 036726 000004 MOV      #TIMEOUT,4
      045266 005037 177766    CLR      CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
      ;THAT A EXPECTED TRAP COULD HAVE SET
      .DSABL CRF
7319 045272 000207    RETURN
  
```

```

7322 045274          FSCMD5: SUBTST <<FS  COMMAND 5  SELECT BANK,MARGIN, & PATTERN>>
:*****
:SUBTEST          FS  COMMAND 5  SELECT BANK,MARGIN, & PATTERN
:*****
7323 045274          PUSH  BANK,PATTERN,MARGIN,CSRFIRST,PCBUMP,TKVEC,TKVEC+2
045274 013746 002106          MOV BANK,-(SP)
045300 013746 002120          MOV PATTERN,-(SP)
045304 013746 002112          MOV MARGIN,-(SP)
045310 013746 002336          MOV CSRFIRST,-(SP)
045314 013746 002406          MOV PCBUMP,-(SP)
045320 013746 000060          MOV TKVEC,-(SP)
045324 013746 000062          MOV TKVEC+2,-(SP)
7324 045330 013737 002112 002376  MOV MARGIN,OLDMARGIN
7325 045336 010637 002374          MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
7326 045342          TYPE MSG040 ;SELECT BANK, MARGIN, & PATTERN TEST
045342 104401 114010          TYPEIT ,MSG040
.DSABL CRF
7327 045346          1$: TYPE MSG030 ;BANK(0-167)?
045346 104401 113454          TYPEIT ,MSG030
.DSABL CRF
7328 045352 104413          RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7329 045354          POP BANK ;PUT IT IN BANK
045354 012637 002106          MOV (SP)+,BANK
7330 045360          IF BANK GT #167 THEN GOTO 1$ ;CHECK FOR BANK TO HIGH
045360 023727 002106 000167          CMP BANK,#167
045366 003367          BGT 1$
7331
7332 045370 013701 002106          MOV BANK,R1
7333 045374 070127 000006          MUL #6,R1
7334 045400          IF CPUBIT OFF.IN CONFIG(R1)
045400 033761 002114 003016          BIT (PUBIT,CONFIG(R1))
045406 001003          BNE L513
7335 045410          TYPE MSG041 ;BANK NOT ACCESSABLE
045410 104401 114055          TYPEIT ,MSG041
.DSABL CRF
GOTO 1$
7336 045414          BR 1$
045414 000754
7337 045416          END ;OF IF
045416          L513:::::
7338
7339 045416          2$: TYPE MSG042 ;PATTERN(0-37)?
045416 104401 114102          TYPEIT ,MSG042
.DSABL CRF
7340 045422 104413          RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7341 045424          POP PATTERN ;PUT IT IN PATTERN
045424 012637 002120          MOV (SP)+,PATTERN
7342 045430          IF PATTERN GT #37 THEN GOTO 2$ ;CHECK FOR PATTERN TO HIGH
045430 023727 002120 000037          CMP PATTERN,#37
045436 003367          BGT 2$
7343 045440          IF PATTERN EQ #0
045440 005737 002120          TST PATTERN
045444 001004          BNE L514
7344 045446          TYPE MSG043 ;PATTERN 0 DATA IS?
045446 104401 114123          TYPEIT ,MSG043
.DSABL CRF
7345 045452 104413          RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
7346 045454          POP R2 ;PUT IT IN R2
    
```

```

7347 045454 012602                                MOV (SP)+,R2
045456                                END ;OF IF
045456                                L514:::
7348 045456                                CMD5A: TYPE MSG044 ;MARGIN(0,2,3,4,5)?
7349 045456 104401 114150                       TYPEIT ,MSG044
                                           .DSABL CRF
                                           RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
                                           POP MARGIN ;PUT IN IN MARGIN
7350 045462 104413
7351 045464 012637 002112                       MOV (SP)+,MARGIN
045464 002112 000005                           CMP MARGIN,#5
045470 023727 002112 000005                   BGT L515
045476 003004
045500 023727 002112 000001                   CMP MARGIN,#1
045506 001001                                 BNE L516
045510                                L515:::
7353 045510                                GOTO CMD5A
045510 000762                                BR CMD5A
7354 045512                                END ;OF IF
045512                                IF MARGIN NE #0 AND #SW11 SET.IN @SWR
7355 045512 005737 002112                       TST MARGIN
045516 001407                                 BEQ L517
045520 032777 004000 135240                   BIT #SW11,@SWR
045526 001403                                 BEQ L517
7356 045530                                TYPE MSG045 ;MARGINS INHIBITED BY SW11
045530 104401 114175                       TYPEIT ,MSG045
                                           .DSABL CRF
                                           GOTO CMD5A
7357 045534                                BR CMD5A
045534 000750
7358 045536                                END ;OF IF MARGIN
045536                                L517:::
7359 045536                                MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7360 045536 010346                                MOV R3,-(SP)
045540 013703 002106                            MOV BANK,R3
045544 004737 040700                            CALL MAPPER
                                           .DSABL CRF
7361 045550 012603                                MOV (SP)+,R3
045552 104511
7362 045554 004737 042422                       INVALIDATE
045560 005737 002132                           CALL EXBANK ;SET NEW MARGINS
045564 001403                                 IF RRFLAG IS TRUE
7363 045566                                TST RRFLAG
045566 104401 114343                           BEQ L520
                                           ;BANK REQUIRES RELOCATION
7364 045572                                TYPE MSG049
045572 000527                                TYPEIT ,MSG049
                                           .DSABL CRF
                                           GOTO CMD5C
7365 045574                                BR CMD5C
045574                                END ;OF IF RRFLAG
7366 045574 004737 043300                       CALL COREHISTORY
045574 004737 043300                           TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
7367 045600 104401 114231                       TYPEIT ,MSG046
045600 104401 114231                           .DSABL CRF
7368 045604 012737 046052 000060               MOV #CMD5C,TKVEC
045612 012737 000340 000062                   MOV #340,TKVEC+2
  
```

```

7371 045620 017700 135150      MOV    @STKB,R0      ;KILL ANY OLD INTERRUPT
7372 045624 042737 000200 177776  BIC    #BIT7,PSW    ;LOWER CPU PRIORITY TO 140
7373 045632 052777 000100 135132  BIS    #BIT6,@STKS  ;ENABLE KEYBOARD INTERRUPTS
7374
7375 045640 013700 002112      MOV    MARGIN,R0    ;SET MARGINS
7376 045644 006300      ASL    R0            ;SET MARGINS
7377
7378 045646 010037 177750      MOV    R0,MAINT
7379
7380 045652      SET    HEADER,MUT
                                MOV    #-1,HEADER
                                MOV    #-1,MUT
                                045652 012737 177777 002724
                                045660 012737 177777 002116
7381 045666 005037 002362      CMD5B: CLR    PASFLG
7382 045672 012737 060000 002336  MOV    #FIRST,CSRFIRST
7383 045700      IF #SWO SET.IN @SWR
                                BIT    #SWO,@SWR
                                BEQ    L521
                                045700 032777 000001 135060
                                045706 001402
7384 045710 104470      ECCDIS
7385 045712      ELSE
                                ;DISABLE ERROR CORRECTION
                                045712 000401
                                BR    L522
7386 045714 104502      CLRCR
7387 045716      END ;OF IF
                                ;CLEAR MK11 CSRS
                                L521:::
7388 045716 012737 000002 002100  MOV    #2,NOPAR    ;PARITY ACTION
7389 045724 012737 000002 002406  MOV    #2,PCBUMP   ;TRAPS ADD 2 TO PC
7390 045732 013700 002120      MOV    PATTERN,R0
7391 045736 006300      ASL    R0
7392 045740 004770 045752  CALL   @FSPAT(R0)
7393 045744 005037 002100  CLR    NOPAR
7394 045750 000746  BR     CMD5B
                                ;LOOP TILL KEYBOARD INTERRUPT
7395
7396 045752 021704      FSPAT: MT0000  ;<1 SEC  DATA PATTERN TEST
7397 045754 021740      MT0001  ;<1 SEC  ADDRESS TEST
7398 045756 022000      MT0002  ;<1 SEC  COMPLEMENT ADDRESS TEST
7399 045760 022050      MT0003  ; 1 SEC  3 XOR 9 WORST CASE NOISE TEST
7400 045762 022216      MT0004  ; 1 SEC  ROTATING ZEROS TEST
7401 045764 022300      MT0005  ; 1 SEC  ROTATING ONES TEST
7402 045766 022360      MT0006  ;<1 SEC  INITIAL DATA TEST
7403 045770 022414      MT0007  ;<1 SEC  ADDRESS BIT TEST
7404 045772 022456      MT0010  ;<1 SEC  BYTE ADDRESSING TEST
7405 045774 022512      MT0011  ;<2 SEC  CREATE SINGLE BIT ERROR TEST
7406 045776 022570      MT0012  ;<1 SEC  WRITE BYTE CLEARS SBE TEST
7407 046000 022646      MT0013  ; 1 SEC  CREATE DOUBLE BIT ERROR TEST
7408 046002 022722      MT0014  ; 1 SEC  WRITE INHIBIT DURING DATIP WITH DBE
7409 046004 023000      MT0015  ; 1 SEC  WRITE INHIBIT OF BYTE WITH DBE
7410 046006 023056      MT0016  ;<1 SEC  WRITE INHIBIT OF WORD WITH DBE
7411 046010 023134      MT0017  ;<1 SEC  HOLDING 1'S & 0'S TEST
7412 046012 023156      MT0020  ;<1 SEC  MARCHING 1'S & 0'S IN CHECK BITS
7413 046014 023612      MT0021  ; 1 SEC  MARCHING 0'S & 1'S TEST
7414 046016 023734      MT0022  ;10 SEC  REFRESH & SHIFTING DIAGONAL TEST
7415 046020 023766      MT0023  ;10 SEC  SHIFTING DIAGONAL TEST
7416 046022 024032      MT0024  ;20 SEC  FAST GALLOPING PATTERN TEST
7417 046024 024172      MT0025  ;<1 SEC  INTERRUPT ENABLE TEST
7418 046026 024240      MT0026  ;<1 SEC  RANDOM DATA TEST
7419 046030 024370      MT0027  ; 1 SEC  UNIQUE BANK TEST
7420 046032 024754      MT0030  ; 1 SEC  FLUSH OUT DBE'S TEST
  
```

```

7421 046034 025214 MT0031 : 3 SEC SOB-A-LONG TEST
7422 046036 025772 MT0999:MT0032
7423 046040 025340 MT0033 : <1 SEC WRITE RECOVERY TEST
7424 046042 025530 MT0034 : 35 SEC BRANCH GOBBLE TEST
7425 046044 025772 MT0999:MT0035
7426 046046 025772 MT0999:MT0036
7427 046050 025772 MT0999:MT0037
7428
7429 046052 013706 002374 CMD5C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
7430 046056 042777 000100 134706 BIC #BIT6,@$TKS
7431 046064 012637 000062 POP TKVEC+2,TKVEC
046064 012637 000062 MOV (SP)+,TKVEC+2
046070 012637 000060 MOV (SP)+,TKVEC
7432 046074 117700 134674 MOV#B @$TKB,R0 ;GET CHARACTER TO GET RID OF FLAG
7433 046100 012637 002406 POP PCBUMP,CSRFIRST,MARGIN
046100 012637 002406 MOV (SP)+,PCBUMP
046104 012637 002336 MOV (SP)+,CSRFIRST
046110 012637 002112 MOV (SP)+,MARGIN
7434 046114 004737 043300 CALL COREHISTORY ;RESTORE OLD MARGINS
7435 046120 012637 002120 POP PATTERN,BANK
046120 012637 002120 MOV (SP)+,PATTERN
046124 012637 002106 MOV (SP)+,BANK
7436 046130 010346 MAP BANK ;REMAP OLD BANK
046130 010346 MOV R3,-(SP)
046132 013703 002106 MOV BANK,R3
046136 004737 040700 CALL MAPPER
.DSABL CRF
046142 012603 MOV (SP)+,R3
7437 046144 004737 042422 CALL EXBANK
7438 046150 000207 RETURN
7439
7440 046152 FSCMD6: SUBTST <<FS COMMAND 6 TYPE CONFIGURATION MAP>>
:*****
:*SUBTEST FS COMMAND 6 TYPE CONFIGURATION MAP
:*****
CALL PCONFIG
7441 046152 004737 035402 RETURN
7442 046156 000207
7443
  
```

```

7447 046160          FSCMD7: SUBTST <<FS  COMMAND 7  BATTERY BACKUP TEST>>
:*****
:*SUBTEST          FS  COMMAND 7  BATTERY BACKUP TEST
:*****
7448 046160          PUSH  BANK,PATTERN,MARGIN,CSRNO,TKVEC,TKVEC+2
046160 013746 002106          MOV BANK,-(SP)
046164 013746 002120          MOV PATTERN,-(SP)
046170 013746 002112          MOV MARGIN,-(SP)
046174 013746 002256          MOV CSRNO,-(SP)
046200 013746 000060          MOV TKVEC,-(SP)
046204 013746 000062          MOV TKVEC+2,-(SP)
7449 046210          SET    FS7FLAG
046210 012737 177777 002546          MOV #-1,FS7FLAG
7450 046216          IF #SWO SET.IN @SWR
046216 032777 000001 134542          BIT #SWO,@SWR
046224 001402          BEQ L523
7451 046226 104470          ECCDIS          ;DISABLE ERROR CORRECTION
7452 046230          ELSE
046230 000401          BR L524
046232          L523:*****
7453 046232 104502          CLRCR          ;CLEAR MK11 CSRS
7454 046234          END ;OF IF
046234          L524:*****
7455 046234 104401 114375          TYPE MSG050          ;BATTERY BACKUP TEST
TYPEIT ,MSG050
.DSABL CRF
          ;WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND
7456          ;CREATE & CHECK BACKGROUND
7457 046240 004737 024370          CALL MT0027
7458 046244 012737 000012 002434          MOV #10,TIME
7459 046252 104401 114523          TYPE MSG052          ;REMOVE SYSTEM POWER FOR
TYPEIT ,MSG052
.DSABL CRF
TYPDEC TIME
7460 046256 013746 002434          MOV TIME,-(SP)          ;;SAVE TIME FOR TYPEOUT
046262 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
.DSABL CRF
7461 046264 104401 114554          TYPE MSG053          ;SECONDS MAX!
TYPEIT ,MSG053
.DSABL CRF
7462 046270 104401 114231          TYPE MSG046          ;TO ESCAPE TYPE ANY KEY
TYPEIT ,MSG046
.DSABL CRF
7463 046274 012737 046336 000060          MOV #CMD7A,TKVEC
7464 046302 012737 000340 000062          MOV #340,TKVEC+2
7465 046310 017700 134460          MOV @STKB,R0          ;KILL ANY OLD INTERRUPT
7466 046314 042737 000200 177776          BIC #BIT7,PSW          ;LOWER CPU PRIORITY TO 140
7467 046322 052777 000100 134442          BIS #BIT6,@STKS          ;ENABLE KEYBOARD INTERRUPTS
7468 046330 010637 002374          MOV SP,FSSTACK
7469
7470 046334 000001          WAIT
7471
7472 046336 013706 002374          CMD7A: MOV FSSTACK,SP          ;RECOVER OLD STACK POINTER
7473 046342 042777 000100 134422          BIC #BIT6,@STKS
7474 046350          POP TKVEC+2,TKVEC
046350 012637 000062          MOV (SP)+,TKVEC+2
046354 012637 000060          MOV (SP)+,TKVEC
7475 046360 117700 134410          MOVB @STKB,R0          ;GET CHARACTER TO GET RID OF FLAG
  
```

```

7476 046364 104401 114572 TYPE MSG054 ;NOW STARTING READ TEST OF MEMORY BANKS
      046364 005037 002106 TYPEIT ,MSG054
      046374 004737 042422 .DSABL CRF
      046400 005737 002124 FOR BANK := #0 TO #167
      046404 001416
      046406 005737 002132 CALL EXBANK
      046412 001013 IF ACFLAG IS TRUE AND RRFLAG IS FALSE
7480 046414 104511 INVALIDATE
7481 046416 013702 002106 LET R2 := BANK
      046416 012700 060000 MOV #FIRST,R0
7482 046422 012700 060000 MOV R0,R4
7483 046426 010004 MOV #SIZE,R1
7484 046430 012701 040000 MOV R1,R3
7485 046434 010103 CALL SUPD03
7486 046436 004737 026220 END ;OF IF ACFLAG
7487 046442 046442 L525:::
7488 046442 005237 002106 END ;OF FOR BANK
      046442 023727 002106 000167 INC BANK
      046446 003747 CMP BANK,#167
      046454 BLE B64
      046456 E64:::
7489 046456 104401 114264 TYPE MSG047 ;TEST COMPLETE
      046456 005037 002546 TYPEIT ,MSG047
7490 046462 005037 002546 .DSABL CRF
7491 046466 012637 002256 CLR FS7FLAG
      046466 012637 002112 POP CSRNO,MARGIN,PATTERN,BANK
      046472 012637 002120 MOV (SP)+,CSRNO
      046476 012637 002106 MOV (SP)+,MARGIN
      046502 000207 MOV (SP)+,PATTERN
      046506 000207 MOV (SP)+,BANK
7492
7493 RETURN
  
```

```

7496 046510      FSCMD8: SUBTST <<FS      COMMAND 8      SOB-A-LONG TEST>>
:*****
:*SUBTEST      FS      COMMAND 8      SOB-A-LONG TEST
:*****
7497 046510      PUSH      BANK,PATTERN,MARGIN,TKVEC,TKVEC+2,NOPAR
046510 013746 002106      MOV BANK,-(SP)
046514 013746 002120      MOV PATTERN,-(SP)
046520 013746 002112      MOV MARGIN,-(SP)
046524 013746 000060      MOV TKVEC,-(SP)
046530 013746 000062      MOV TKVEC+2,-(SP)
046534 013746 002100      MOV NOPAR,-(SP)
7498 046540 010637 002374      MOV      SP,FSSTACK      ;SAVE LAST GOOD STACK POINTER
7499 046544      TYPE      MSG055      ;SOB-A-LONG TEST
046544 104401 114642      TYPEIT    ,MSG055
                                .DSABL    CRF
7500 046550      CMD8A:      TYPE      MSG044      ;MARGIN(0,2,3,4,5)?
046550 104401 114150      TYPEIT    ,MSG044
                                .DSABL    CRF
7501 046554 104413      RDOCT
7502 046556      POP      MARGIN      ;READ AN OCTAL NUMBER ONTO THE STACK
046556 012637 002112      ;PUT IN IN MARGIN
7503 046562      IF MARGIN GT #5 OR MARGIN EQ #1      MOV (SP)+,MARGIN
046562 023727 002112 000005      CMP MARGIN,#5
046570 003004      BGT L526
046572 023727 002112 000001      CMP MARGIN,#1
046600 001001      BNE L527
046602      L526:*****
7504 046602      GOTO CMD8A
046602 000762      BR CMD8A
7505 046604      END ;OF IF
046604      L527:*****
7506 046604      IF MARGIN NE #0 AND #SW11 SET.IN @SWR
046604 005737 002112      TST MARGIN
046610 001407      BEQ L530
046612 032777 004000 134146      BIT #SW11,@SWR
046620 001403      BEQ L530
7507 046622      ;MARGINS INHIBITED BY SW11
046622 104401 114175      TYPE      MSG045
                                TYPEIT    ,MSG045
                                .DSABL    CRF
                                GOTO CMD8A
7508 046626      BR CMD8A
046626 000750      END ;OF IF MARGIN
7509 046630      L530:*****
046630
7510
7511 046630      IF #SW0 SET.IN @SWR
046630 032777 000001 134130      BIT #SW0,@SWR
046636 001402      BEQ L531
7512 046640 104470      ECCDIS      ;DISABLE ERROR CORRECTION
7513 046642      ELSE
046642 000401      BR L532
046644      L531:*****
7514 046644 104502      CLRCR      ;CLEAR MK11 CSRS
7515 046646      END ;OF IF
046646      L532:*****
7516 046646      TYPE      MSG056      ;BELL = EACH PASS COMPLETE
046646 104401 114663      TYPEIT    ,MSG056
                                .DSABL    CRF
    
```

```

7517
7518 046652          TYPE      MSG046          ;TO ESCAPE TYPE ANY KEY!
      046652 104401 114231      TYPEIT    ,MSG046
      .DSABL    CRF
7519 046656 012737 047014 000060      MOV      #CMD8C,TKVEC
7520 046664 012737 000340 000062      MOV      #340,TKVEC+2
7521 046672 017700 134076          MOV      @$TKB,R0          ;KILL ANY OLD INTERRUPT
7522 046676 042737 000200 177776      BIC      #BIT7,PSW        ;LOWER CPU PRIORITY TO 140
7523 046704 052777 000100 134060      BIS      #BIT6,@$TKS     ;ENABLE KEYBOARD INTERRUPTS
7524
7525 046712 013700 002112          MOV      MARGIN,R0       ;SET MARGINS
7526 046716 006300          ASL      R0              ;SET MARGINS
7527 046720 010037 177750          MOV      R0,MAINT
7528
7529 046724          SET      HEADER,MUT
      046724 012737 177777 002724      MOV      #-1,HEADER
      046732 012737 177777 002116      MOV      #-1,MUT
7530
7531 046740          CMD8B:  FOR BANK := #0 TO #167
      046740 005037 002106          CLR      BANK
      046744          B65:.....
7532 046744 004737 042422          CALL    EXBANK
7533 046750          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
      046750 005737 002124          TST     ACFLAG
      046754 001406          BEQ     L533
      046756 005737 002132          TST     RRFLAG
      046762 001003          BNE     L533
7534 046764 104511          INVALIDATE
7535 046766 004737 025214          CALL    MTO031
7536 046772          END ;OF IF ACFLAG
      046772          L533:.....
7537 046772          END ;OF FOR BANK
      046772 005237 002106          INC     BANK
      046776 023727 002106 000167      CMP     BANK,#167
      047004 003757          BLE     B65
      047006          B65:.....
7538 047006          TYPE      $BELL          ;RING BELL
      047006 104401 003003      TYPEIT    , $BELL
      .DSABL    CRF
7539 047012          GOTO     CMD8B
      047012 000752          BR      CMD8B
7540
7541 047014 013706 002374          CMD8C:  MOV      FSSTACK,SP          ;RECOVER OLD STACK POINTER
7542 047020 042777 000100 133744      BIC      #BIT6,@$TKS
7543 047026 117700 133742          MOV     @$TKB,R0          ;READ CHAR TO KILL FLAG
7544 047032          POP      NOPAR,TKVEC+2,TKVEC,MARGIN,PATTERN,BANK
      047032 012637 002100          MOV     (SP)+,NOPAR
      047036 012637 000062          MOV     (SP)+,TKVEC+2
      047042 012637 000060          MOV     (SP)+,TKVEC
      047046 012637 002112          MOV     (SP)+,MARGIN
      047052 012637 002120          MOV     (SP)+,PATTERN
      047056 012637 002106          MOV     (SP)+,BANK
7545 047062          MAP      BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      047062 010346          MOV     BANK,R3
      047064 013703 002106          CALL    MAPPER
      047070 004737 040700          .DSABL  CRF
      MOV     R3,-(SP)

```

047074 012603
7546 047076 004737 042422
7547 047102 000207

CALL EXBANK
RETURN

MOV (SP)+,R3

```

7550 047104      FSCMD9: SUBTST <<FS      COMMAND 9      SUPER TIGHT SCOPE LOOP>>
:*****
:*SUBTEST      FS      COMMAND 9      SUPER TIGHT SCOPE LOOP
:*****
7551 047104      PUSH      BANK,MARGIN,TKVEC,TKVEC+2,NOPAR
047104 013746 002106      MOV BANK,-(SP)
047110 013746 002112      MOV MARGIN,-(SP)
047114 013746 000060      MOV TKVEC,-(SP)
047120 013746 000062      MOV TKVEC+2,-(SP)
047124 013746 002100      MOV NOPAR,-(SP)
7552 047130 010637 002374      MOV      SP,FSSTACK      ;SAVE LAST GOOD STACK
7553 047134      TYPE      MSG059      ;SUPER TIGHT SCOPE LOOP
047134 104401 114766      TYPEIT    ,MSG059
      .DSABL   CRF
      IF #SWO SET.IN @SWR
7554 047140      ECCDIS      ;DISABLE ERROR CORRECTION
047140 032777 000001 133620      ELSE
047146 001402      BIT #SWO,@SWR
7555 047150 104470      BEQ L534
7556 047152 000401      BR L535
047154      L534:::
7557 047154 104502      CLRCRSR      ;CLEAR MK11 CSRS
7558 047156      END ;OF IF
047156      L535:::
7559 047156 104401 113473      1$: TYPE      MSG031      ;PHYSICAL ADDRESS (0-1677776)?
047156      TYPEIT    ,MSG031
      .DSABL   CRF
7560 047162 104413      RDOCT      ;READ OCTAL NUMBER ONTO STACK & $HIOCT
7561 047164 013737 074434 002106      MOV      $HIOCT,BANK      ;PUT MSB'S IN BANK
7562 047172 012600      POP      RO      ;PUT LSB'S IN RO
047172      MOV (SP)+,RO
7563 047174 000241      CLC
7564 047176 006100      ROL      RO
7565 047200 006137 002106      ROL      BANK
7566 047204 000241      CLC
7567 047206 006000      ROR      RO
7568 047210      IF BANK GT #167 THEN GOTO 1$      ;CHECK FOR BANK TOO HIGH
047210 023727 002106 000167      CMP BANK,#167
047216 003357      BGT 1$
7569 047220 062700 060000      ADD      #FIRST,RO
7570 047224 032700 000001      IF #BITO SET.IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
047224 001352      BIT #BITO,RO
047230      BNE 1$
7571 047232 020027 157776      IF RO HI #LAST THEN GOTO 1$      ;CHECK FOR ADDRESS OVER 16K
047232 101347      CMP RO,#LAST
047236      BHI 1$
7572 047240      CMD9A: TYPE      MSG044      ;MARGIN(0,2,3,4,5)?
7573 047240 104401 114150      TYPEIT    ,MSG044
      .DSABL   CRF
7574 047244 104413      RDOCT      ;READ AN OCTAL NUMBER ONTO THE STACK
7575 047246 012637 002112      POP      MARGIN      ;PUT IN IN MARGIN
047246      MOV (SP)+,MARGIN
7576 047252 023727 002112 000005      IF MARGIN GT #5 OR MARGIN EQ #1
047252 003004      CMP MARGIN,#5
047260 023727 002112 000001      BGT L536
047262      CMP MARGIN,#1
  
```

```

047270 001001
047272
7577 047272          GOTO CMD9A
047272 000762
7578 047274          END ;OF IF
047274
7579 047274          IF MARGIN NE #0 AND #SW11 SET.IN @SWR
047274 005737 002112
047300 001407
047302 032777 004000 133456
047310 001403
7580 047312          TYPE MSG045          ;MARGINS INHIBITED BY SW11
047312 104401 114175  TYPEIT ,MSG045
      .DSABL CRF
      GOTO CMD9A
7581 047316          BR CMD9A
047316 000750
7582 047320          END ;OF IF MARGIN
047320
7583
7584 047320 012737 000003 002100  MOV #3,NOPAR          ;PARITY ACTION
7585 047326 012737 047444 002372  MOV #CMD9LOOP,PARHERE
7586 047334 012737 047456 000004  SET4 #CMD9B          ;TRAPS TO 4 GOTO CMD9B
      MOV #CMD9B,4
      .DSABL CRF
      MAP BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7587 047342          MOV R3,-(SP)
047342 010346
047344 013703 002106
047350 004737 040700
      MOV BANK,R3
      CALL MAPPER
      .DSABL CRF
      MOV (SP)+,R3
7588 047354 012603
7589 047356 104511          INVALIDATE
047360 104401 115016  TYPE MSG060          ;SWR ----> MEM, MEM ----> DISPLAY
      TYPEIT ,MSG060
      .DSABL CRF
7590
7591 047364          TYPE MSG046          ;TO ESCAPE TYPE ANY KEY!
047364 104401 114231  TYPEIT ,MSG046
      .DSABL CRF
7592 047370 012737 047464 000060  MOV #CMD9C,TKVEC
7593 047376 012737 000340 000062  MOV #340,TKVEC+2
7594 047404 017701 133364          MOV @STKB,R1          ;KILL ANY OLD INTERRUPT
7595 047410 042737 000200 177776  BIC #BIT7,PSW          ;LOWER CPU PRIORITY TO 140
7596 047416 052777 000100 133346  BIS #BIT6,@STKS          ;ENABLE KEYBOARD INTERRUPTS
7597
7598 047424 013701 002112          MOV MARGIN,R1          ;SET MARGINS
7599 047430 006301          ASL R1          ;SET MARGINS
7600 047432 010137 177750          MOV R1,MAINT
7601
7602 047436          SUPERVISOR          ;ENTER SUPERVISOR MODE
047436 052737 040000 177776  BIS #BIT14,PSW          ;DO IT TO IT!
      .DSABL CRF
7603 047444 017710 133316  CMD9LOOP:MOV @SWR,(R0)          ;WRITE SWITCH REGISTER INTO MEMORY
7604 047450 011077 133314  MOV (R0),@DISPLAY          ;READ MEMORY INTO DISPLAY REGISTER
7605 047454 000773          BR CMD9LOOP          ;LOOP TILL KEYBOARD INTERRUPT
7606
7607 047456 012716 047444  CMD9B: MOV #CMD9LOOP,(SP)
7608 047462 000002          RTI
  
```

```

7609
7610 047464 013706 002374      CMD9C:  MOV    FSSTACK,SP           ;RECOVER OLD STACK POINTER
7611 047470 042777 000100      BIC    #BIT6,@$TKS
7612 047476 117700 133272      MOVSB  @$TKB,R0           ;READ CHAR TO KILL FLAG
7613 047502      POP    NOPAR,TKVEC+2,TKVEC,MARGIN,BANK
      MOV (SP)+,NOPAR
      MOV (SP)+,TKVEC+2
      MOV (SP)+,TKVEC
      MOV (SP)+,MARGIN
      MOV (SP)+,BANK
      MOV R3,-(SP)
      047502 012637 002100
      047506 012637 000062
      047512 012637 000060
      047516 012637 002112
      047522 012637 002106
7614 047526      MAP    BANK           ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      MOV R3,-(SP)
      047526 010346
      047530 013703 002106
      047534 004737 040700
      MOV    BANK,R3
      CALL  MAPPER
      .DSABL CRF
      MOV (SP)+,R3
      047540 012603
7615 047542 004737 042422      CALL  EXBANK
7616 047546 000207      RETURN
7617
7618 047550      FCMD10: SUBTST <<FS  COMMAND 10  ERROR SUMMARY>>
      ;*****
      ;*SUBTEST    FS  COMMAND 10  ERROR SUMMARY
      ;*****
      PUSH  R0,R2,R3,BANK
      MOV R0,-(SP)
      MOV R2,-(SP)
      MOV R3,-(SP)
      MOV BANK,-(SP)
7619 047550      047550 010046
      047552 010246
      047554 010346
      047556 013746 002106
7620 047562      TYPDEC $ERTTL
      047562 013746 002754      MOV    $ERTTL,-(SP)      ;;SAVE $ERTTL FOR TYPEOUT
      047566 104405      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
      TYPE    MSG079           ;ERROR(S) DETECTED
      TYPEIT ,MSG079
      .DSABL CRF
      IF $ERTTL NE #0
7621 047570      104401 115474
7622 047574      047574 005737 002754      TST $ERTTL
      047600 001447      BEQ L541
7623 047602 005037 002424      CLR SUCCESS
7624 047606      FOR BANK := #0 TO #167
      CLR BANK
      B66:;;;;;
7625 047612      047612 013703 002106      MOV BANK,R3
7626 047616 070327 000006      MUL #6,R3
7627 047622 062703 000005      ADD #5,R3
7628 047626      IFB CONFIG(R3) NE #0
      TSTB CONFIG(R3)
      BEQ L542
      047626 105763 003016
      047632 001424
7629 047634      IF SUCCESS IS FALSE
      TST SUCCESS
      047634 005737 002424      BNE L543
      047640 001005
7630 047642      TYPE    MSG076           ;BANK ERRORS
      047642 104401 115373      TYPEIT ,MSG076
      .DSABL CRF
7631 047646      SET SUCCESS
      047646 012737 177777 002424      MOV #-1,SUCCESS
7632 047654      END ;OF IF SUCCESS
  
```

L543:::~::~

7633	047654	013746	002106		MOV	TYPOCS BANK,3		
	047654					BANK,-(SP)	:::SAVE BANK FOR TYPEOUT	
	047660	104403					:::3	
	047662	006			TYPOS		:::GO TYPE--OCTAL ASCII	
	047663	000			.BYTE	6	:::TYPE 6 DIGITS	
					.BYTE	0	:::SUPPRESS LEADING ZEROS	
					.DSABL	CRF		
7634	047664	116300	003016		MOVB	CONFIG(R3),R0		
7635	047670	042700	177400		BIC	#^C377,R0		
7636	047674				TYPDEC	R0		
	047674	010046			MOV	R0,-(SP)	:::SAVE R0 FOR TYPEOUT	
	047676	104405			TYPDS		:::GO TYPE--DECIMAL ASCII WITH SIGN	
					.DSABL	CRF		
7637	047700					TYPE \$CRLF		
	047700	104401	003010		TYPEIT	\$CRLF		
					.DSABL	CRF		
7638	047704					END ;OF IFB CONFIG(R3)		
	047704							L542:::~::~
7639	047704					END ;OF FOR BANK		
	047704	005237	002106				INC BANK	
	047710	023727	002106	000167			CMP BANK,#167	
	047716	003735					BLE B66	
	047720							E66:::~::~
7640	047720					END ;OF IF \$ERTTL		
	047720							L541:::~::~
7641	047720				POP	BANK,R3,R2,R0		
	047720	012637	002106				MOV (SP)+,BANK	
	047724	012603					MOV (SP)+,R3	
	047726	012602					MOV (SP)+,R2	
	047730	012600					MOV (SP)+,R0	
7642	047732	000207			RETURN			


```

7666
7667 050076 104401 114231          TYPE MSG046          ;TO ESCAPE TYPE ANY KEY!
                                TYPEIT ,MSG046
                                .DSABL CRF
7668 050102 012737 050240 000060  MOV #CMD11C,TKVEC
7669 050110 012737 000340 000062  MOV #340,TKVEC+2
7670 050116 017700 132652          MOV @TKB,R0          ;KILL ANY OLD INTERRUPT
7671 050122 042737 000200 177776  BIC #BIT7,PSW        ;LOWER CPU PRIORITY TO 140
7672 050130 052777 000100 132634  BIS #BIT6,@TKS      ;ENABLE KEYBOARD INTERRUPTS
7673
7674 050136 013700 002112          MOV MARGIN,R0        ;SET MARGINS
7675 050142 006300          ASL R0               ;SET MARGINS
7676 050144 010037 177750          MOV R0,MAINT
7677
7678 050150          SET HEADER,MUT
                                MOV #-1,HEADER
                                MOV #-1,MUT
7679
7680 050164 005037 002106          CMD11B: FOR BANK := #0 TO #167
                                CLR BANK
                                B67:.....
7681 050170 004737 042422          CALL EXBANK
7682 050174 005737 002124          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
                                TST ACFLAG
                                BEQ L551
                                TST RRFLAG
                                BNE L551
7683 050210 104511          INVALIDATE
7684 050212 004737 023734          CALL MT0022
7685 050216          END ;OF IF ACFLAG
                                L551:.....
7686 050216          END ;OF FOR BANK
                                INC BANK
                                CMP BANK,#167
                                BLE B67
                                B67:.....
7687 050232 104401 003003          TYPE $BELL          ;RING BELL
                                TYPEIT ,SBELL
                                .DSABL CRF
7688 050236 000752          GOTO CMD11B
                                BR CMD11B
7689
7690 050240 013706 002374 132520  CMD11C: MOV FSSTACK,SP          ;RECOVER OLD STACK POINTER
7691 050244 042777 000100          BIC #BIT6,@TKS
7692 050252 117700 132516          MOV @TKB,R0          ;READ CHAR TO KILL FLAG
7693 050256 012637 002100          POP NOPAR,TKVEC+2,TKVEC,MARGIN,PATTERN,BANK
                                MOV (SP)+,NOPAR
                                MOV (SP)+,TKVEC+2
                                MOV (SP)+,TKVEC
                                MOV (SP)+,MARGIN
                                MOV (SP)+,PATTERN
                                MOV (SP)+,BANK
7694 050306 010346          MAP BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
                                MOV R3,-(SP)
                                MOV BANK,R3
                                CALL MAPPER
                                .DSABL CRF
  
```

7695 050320 012603
7696 050322 004737 042422
7697 050326 000207

CALL EXBANK
RETURN

MOV (SP),R3

```
7700 050330 FCMD12: SUBTST <<FS COMMAND 12 SET FILL COUNT>>
:*****
:*SUBTEST FS COMMAND 12 SET FILL COUNT
:*****
PUSH R0 MOV R0,-(SP)
7701 050330 010046
050330 104401 115661 TYPE MSG085 ;FILL COUNT(OCTAL)?
TYPEIT ,MSG085
.DSABL CRF
RDOCT
POP R0 MOV (SP)+,R0
7703 050336 104413
7704 050340 012600
050340 042700 177760 BIC #^C17,R0
7705 050342 110037 002451 MOVB R0,$FILLS
7706 050346 012600 POP R0 MOV (SP)+,R0
7707 050352 000207 RETURN
7708 050354
7709
7710 050356 FCMD13: SUBTST <<FS COMMAND 13 ENTER KAMIKAZE MODE>>
:*****
:*SUBTEST FS COMMAND 13 ENTER KAMIKAZE MODE
:*****
TYPE MSG101 ;ENTERING KAMIKAZE MODE
TYPEIT ,MSG101
.DSABL CRF
SET KAMIKAZE,SKIPKAMI MOV #-1,KAMIKAZE
MOV #-1,SKIPKAMI
7711 050356 104401 116305 RETURN
7712 050362 012737 177777 002010
050362 012737 177777 002012
050370 000207
7713 050376
7714
7715 050400 FCMD14: SUBTST <<FS COMMAND 14 EXIT KAMIKAZE MODE>>
:*****
:*SUBTEST FS COMMAND 14 EXIT KAMIKAZE MODE
:*****
TYPE MSG102 ;LEAVING KAMIKAZE MODE
TYPEIT ,MSG102
.DSABL CRF
CLR KAMIKAZE
SET SKIPKAMI MOV #-1,SKIPKAMI
7716 050400 104401 116335 RETURN
7717 050404 005037 002010
7718 050410 012737 177777 002012
050410 000207
7719 050416
7720
7721 050420 FCMD15: SUBTST <<FS COMMAND 15 TURN CACHE OFF>>
:*****
:*SUBTEST FS COMMAND 15 TURN CACHE OFF
:*****
TYPE MSG106 ;CACHE IS OFF
TYPEIT ,MSG106
.DSABL CRF
MOV #BIT0!BIT2!BIT3!BIT8!BIT9,CACHK
CACHOFF ;TURN CACHE OFF
RETURN
7722 050420 104401 116527
050420 001415 002700
7723 050424 012737 001415 002700
7724 050432 104424
7725 050434 000207
7726
7727 050436 FCMD16: SUBTST <<FS COMMAND 16 TURN CACHE ON>>
:*****
:*SUBTEST FS COMMAND 16 TURN CACHE ON
:*****
```

7728 050436 104401 116545 TYPE MSG107 ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
050436 TYPEIT ,MSG107
.DSABL CRF
7729 050442 012737 000001 002700 MOV #BIT0,CACHK
7730 050450 104423 CACHON ;TURN CACHE ON
7731 050452 000207 RETURN

7732
7733 050454 FCMD17: SUBTST <<FS COMMAND 17 RUN ONLY MULTIPOST TESTS>>
:*****
:*SUBTEST FS COMMAND 17 RUN ONLY MULTIPOST TESTS
:*****

7734 050454 104401 116622 TYPE MSG108 ;ONLY MULTIPOST TESTS WILL BE RUN
050454 TYPEIT ,MSG108
.DSABL CRF
7735 050460 012737 177777 002004 SET MULTIONLY
050460 MOV #-1,MULTIONLY
7736 050466 000207 RETURN

7739 050470

FCMD18: SUBTST <<FS COMMAND 18 RESUME RUNNING BOTH SINGLE & MULTIPOST TESTS>>

 :*SUBTEST FS COMMAND 18 RESUME RUNNING BOTH SINGLE & MULTIPOST TESTS

7740 050470 104401 116664

TYPE MSG109 ;BOTH SINGLE & MULTIPOST TESTS WILL BE RUN
 TYPEIT ,MSG109
 .DSABL CRF
 CLR MULTIONLY
 RETURN

7741 050474 005037 002004

7742 050500 000207

7743

7744 050502

FCMD19: SUBTST <<FS COMMAND 19 TEST ONLY SELECTED BANKS>>

 :*SUBTEST FS COMMAND 19 TEST ONLY SELECTED BANKS

7745 050502 104401 116423

TYPE MSG105 ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
 TYPEIT ,MSG105

7746 050506 004737 050576

.DSABL CRF
 CALL CMD20A ;ERASE OLD SELECTIONS
 BEGIN CMD19LOOP

7747 050512

B70:*****

7748 050512

REPEAT

B71:*****

7749 050512

TYPE MSG030 ;BANK(0-167)?

7750 050516 104413

TYPEIT ,MSG030

7751 050520 012601

.DSABL CRF
 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK

7752 050522 020127 000167

POP R1 ;PUT IT IN R1

7753 050534 000406

IF R1 GT #167 OR R1 LT #0 MOV (SP)+,R1

7754 050536 070127 000006

MUL #6,R1 CMP R1,#167

7755 050542 052761 040000 003020

BIS #BIT14,CONFIG+2(R1) BGT L552

7756 050550 000760

END ;OF REPEAT TST R1

7757 050552

LEAVE CMD19LOOP BGE L553

7758 050552

END CMD20LOOP L552:*****

7759 050552 104401 116737

TYPE MSG110 ;ONLY SELECTED BANKS WILL BE TESTED
 TYPEIT ,MSG110

7760 050556 012737 177777 002002

.DSABL CRF BR E70

7761 050564 000207

SET SELONLY MOV #-1,SELONLY

7762

RETURN

7763 050566

FCMD20: SUBTST <<FS COMMAND 20 RESUME TESTING ALL BANKS>>

 :*SUBTEST FS COMMAND 20 RESUME TESTING ALL BANKS

7764 050566

TYPE MSG111 ;ALL BANKS WILL BE TESTED

K 3

```
050566 104401 117003          TYPEIT ,MSG111
                                .DSABL CRF
7765 050572 005037 002002      CLR SELONLY
7766
7767                                :ENTRY POINT FROM CMD19
7768 050576                                CMD20A: FOR R1 := #0 TO #167*3*2 BY #6
050576 005001
050600
7769 050600 042761 040000 003020      BIC #BIT14,CONFIG+2(R1)
7770 050606 042761 020000 003020      BIC #BIT13,CONFIG+2(R1)
7771 050614                                END ;OF FOR R1
050614 062701 000006
050620 020127 001312
050624 003765
050626
7772 050626 000207          RETURN
```

```
                                CLR R1
                                B72::::::::::
:DESELECT BANK
:INVALIDATE BACKGROUND PATTERN
                                ADD #6,R1
                                CMP R1,#167*3*2
                                BLE B72
                                E72::::::::::
```

```

7775 050630          WHICHCSR:SUBTST <<SUBR DETERMINE CORRECT CSR>>
:*****
:*SUBTEST          SUBR DETERMINE CORRECT CSR
:*****
7776 050630 013700 002334          MOV MKCSRS,R0          :GET CSR'S FLAG
7777 050634 122700 000200          CMPB #BIT7,R0         :CSR 0?
7778 050640 001003          BNE 1$                :NO - SKIP
7779 050642 005037 002256          CLR CSRNO             :YES - SET IT UP
7780 050646 000207          RETURN
7781
7782 050650          1$: TYPE MSG022          :WHICH CSR(0-7)
      050650 104401 113252          TYPEIT ,MSG022
      .DSABL CRF
      RDOCT          :GET OCTAL NUMBER
      POP R0         :PUT IN R0
7783 050654 104413          :CHECK LIMIT
7784 050656 012600          MOV (SP)+,R0
7785 050660 020027 000007          CMP R0,#7
7786 050664 101371          BHI 1$                :IF BAD LOOP TILL HE TYPES IT RIGHT
7787 050666 012701 177774          MOV #-4,R1
7788 050672 005200          INC R0
7789 050674 062701 000004          2$: ADD #4,R1
7790 050700 077003          SOB R0,2$
7791 050702 010137 002256          MOV R1,CSRNO
7792 050706 000207          RETURN
7793
    
```

```

1          .TITLE  CEMKABO 1170 MAIN MEMORY DIAG 2
2          .IDENT  /X01.60/
3          .SBTTL  MULTIPROCESSOR SLAVE DISPATCHER
4 050710    TASKRET:IF #MASTER NE CPUBIT
5 050710    022737 000020 002114          CMP #MASTER,CPUBIT
6 050716    001402          BEQ L554
7 050720    000402          BR    TASKDO
8 050722    000401          ELSE
9 050724    000207          RETURN
10 050726    104424          END; OF IF #MASTER
11 050730    010346          TASKDO: CACHOFF          ;TURN CACHE OFF
12 050732    012703 000000          MAP #0          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
13 050736    004737 040700          MOV #0,R3          MOV R3,-(SP)
14 050742    012603          CALL MAPPER
15 050744    012700 177777          .DSABL CRF
16 050750    005001          SET R0          MOV (SP)+,R3
17 050752    012603          FOR R1 := #0 TO #6 BY #2          MOV #-1,R0
18 050752    020037 002726          IF R0 EQ ONES          CLR R1
19 050756    001024          IF NOIIST IS TRUE          B73:::
20 050760    005737 002756          MOV SYSTID,R2          CMP R0,ONES
21 050764    001403          ELSE          BNE L556
22 050766    013702 177764          MOV @IISTACR,R2          TST NOIIST
23 050772    000406          BR L560          BEQ L557
24 050774    017702 131736          MOV #-8,R2
25 051000    072227 177770          BIC #^C3,R2
26 051004    042702 177774          END ;OF IF NOIIST
27 051010    051010          SUPERVISOR          ;ENTER SUPERVISOR MODE
28 051010    052737 040000 177776          BIS #BIT14,PSW          ;DO IT TO IT!
29 051016    026102 062606          .DSABL CRF
30 051022    001001          IF FIRST+PORTDIR(R1) EQ R2 THEN LET R0 := R1
31 051024    010100          CMP FIRST+PORTDIR(R1),R2
32 051026    104417          BNE L561
33 051030    051030          MOV R1,R0
34 051030    062701 000002          KERNEL          ;ENTER KERNEL MODE
35 051034    020127 000006          END ;OF IF R0          L561:::
36 051040    003744          END ;OF FOR R1          L556:::
37 051042          ADD #2,R1
38          CMP R1,#6
39          BLE B73
40          E73:::
    
```

```

27 051042          IF R0 EQ ONES
    051042 020037 002726          CMP R0,ONES
    051046 001003          BNE L562
28 051050          FATAL 41
    051050 005237 002070          INC FATAL$          ;SET FATAL INDICATOR
    051054 104041          ERROR +41
    .DSABL CRF
    END ;OF IF R0
29 051056          L562:::
    051056          SUPERVISOR          ;ENTER SUPERVISOR MODE
30 051056 052737 040000 177776          BIS #BIT14,PSW          ;DO IT TO IT!
    051056          .DSABL CRF
31 051064 005760 062616          1$: TST FIRST+TASK(R0)          ;WAIT FOR COMMAND
32 051070 003775          BLE 1$
33 051072 013737 062676 002676          MOV FIRST+MDISPLAY,MDISPLAY
34 051100 013777 002676 131662          MOV MDISPLAY,@DISPLAY
35 051106 116001 062616          MOVB FIRST+TASK(R0),R1          ;GET COMMAND
36 051112 104417          KERNEL          ;ENTER KERNEL MODE
37 051114          PUSH R0
    051114 010046          MOV R0,-(SP)
38 051116          CASE R1
    051116 010146          MOV R1,-(SP)
    051120 006316          ASL @SP
    051122 004737 051170          JSR PC,L563
39 051126 051234          TASK00          ;0 IS NOT A COMMAND
40 051130 051236          TASK01          ;MOVE PROGRAM TO BANK
41 051132 051330          TASK02          ;WRITE ONLY PATTERN IN HI BYTE
42 051134 051430          TASK03          ;READ ONLY PATTERN IN HI BYTE
43 051136 051664          TASK04          ;EXECUTION CONTENTION TEST
44 051140 051774          TASK05          ;PORT ARBITRATION SYMMETRY TEST
45 051142 052300          TASK06          ;STOP PROGRAM
46 051144 052326          TASK07          ;WRITE ONLY ADDRESS PATTERN
47 051146 052336          TASK10          ;READ ONLY ADDRESS PATTERN
48 051150 052536          TASK11          ;ASYNCHRONOUS ADDRESS UNIQUENESS TEST
49 051152 052644          TASK12          ;SKEWED UP ADDRESS TEST
50 051154 052726          TASK13          ;SETUP FOR ASRB TEST
51 051156 053276          TASK14          ;RESET FROM ASRB TEST
52 051160 053310          TASK15          ;RUN CSR TESTS
53 051162 053316          TASK16          ;SET MARGIN 6, CPU HAS I/O PRIORITY
54 051164 053334          TASK17          ;CREATE STALE DATA TEST
55 051166 053422          TASK20          ;CACHE FLUSH TEST
56 051170          END ;OF CASE R1
    051170          L563:::
    051170 062616          ADD (SP)+,@SP
    051172 013646          MOV @ (SP)+,-(SP)
    051174 004736          JSR PC,@(SP)+
57 051176          POP R0
    051176 012600          MAP #0          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
58 051200          MOV (SP)+,R0
    051200 010346          MOV R3,-(SP)
    051202 012703 000000          MOV #0,R3
    051206 004737 040700          CALL MAPPER
    .DSABL CRF
    051212 012603          MOV (SP)+,R3
59 051214          SUPERVISOR          ;ENTER SUPERVISOR MODE
    051214 052737 040000 177776          BIS #BIT14,PSW          ;DO IT TO IT!
    .DSABL CRF
    
```

60	051222	052760	100000	062616	BIS	#BIT15, FIRST+TASK(R0)	:INDICATE DONE
61	051230	104417			KERNEL		:ENTER KERNEL MODE
62	051232	000626			BR	TASKRET	
63	051234	000207			TASK00:	RETURN	

C 4

```

66 051236          TASK01: SUBTST <<MOVE PROGRAM TO NEW BANK>>
:*****
:*SUBTEST          MOVE PROGRAM TO NEW BANK
:*****
67 051236          SUPERVISOR          ;ENTER SUPERVISOR MODE
051236 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
68 051244 013700 062106      MOV FIRST+BANK,R0
69 051250 104417          KERNEL          ;ENTER KERNEL MODE
70 051252 004737 041456      CALL RELOC1
71 051256 012737 002000 002706      MOV #STACK,KSTACK
72 051264          POP R1,R0          ;GET OLD RETURN & RO
                                MOV (SP)+,R1
                                MOV (SP)+,R0
73 051270 013706 002706      MOV KSTACK,SP
74 051274          PUSH RO,R1          ;PUT BACK RO & RETURN
                                MOV RO,-(SP)
                                MOV R1,-(SP)
75 051300          MAP #0          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
                                MOV R3,-(SP)
                                MOV #0,R3
                                CALL MAPPER
                                .DSABL CRF
                                MOV (SP)+,R3
                                MOV R0,-(SP)
                                MOV R1,-(SP)
                                MOV R3,-(SP)
                                MOV #0,R3
                                CALL MAPPER
                                .DSABL CRF
                                MOV (SP)+,R3
76 051312 012603          CLR RFLAG
77 051314 005037 002134      SET NOSCOPE          ;NO SCOPE LOOPS ALLOWED IN MULTIPROCESSOR TESTS
                                MOV #-1,NOSCOPE
78 051320 012737 177777 002540      RETURN
79 051326 000207
80 051330          TASK02: SUBTST <<WRITE ONLY PATTERN IN HI BYTE>>
:*****
:*SUBTEST          WRITE ONLY PATTERN IN HI BYTE
:*****
81 051330          SUPERVISOR          ;ENTER SUPERVISOR MODE
051330 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
82 051336 116005 062617          MOVB FIRST+TASK+1(R0),R5
83 051342 013737 062106 002106      MOV FIRST+BANK,BANK
84 051350 104417          KERNEL          ;ENTER KERNEL MODE
85 051352          SET WRITEONLY
                                MOV #-1,WRITEONLY
86 051360 012737 177777 002502      MOV #FIRST,R0
87 051364 012701 040000      MOV #SIZE,R1
88 051370          CASE R5
                                MOV R5,-(SP)
                                ASL @SP
                                JSR PC,L564
                                MOV R5,-(SP)
                                ASL @SP
                                JSR PC,L564
89 051400          SLPAT0          ;ZEROS
90 051402          SLPAT1          ;ONES
91 051404          SLPAT2          ;125252
92 051406          SLPAT3          ;52525
93 051410          SLPAT4          ;377
94 051412          SLPAT5          ;177400
95 051414          END ;OF CASE R5
                                L564:::
                                ADD (SP)+,@SP
                                MOV @ (SP)+,-(SP)
051414 062616
051416 013646
    
```

051420 004736
96 051422 005037 002502
97 051426 000207

CLR WRITEONLY
RETURN

JSR PC,@(SP)+

```

100 051430          TASK03: SUBTST <<READ ONLY PATTERN IN HI BYTE>>
:*****
:*SUBTEST          READ ONLY PATTERN IN HI BYTE
:*****
101 051430          SUPERVISOR          ;ENTER SUPERVISOR MODE
    051430 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
102 051436 116005 062617          MOVB FIRST+TASK+1(R0),R5
103 051442 013737 062106 002106      MOV FIRST+BANK,BANK
104 051450 104417          KERNEL          ;ENTER KERNEL MODE
105 051452          SET READONLY
                                MOV #-1,READONLY
    051452 012737 177777 002504
106 051460 012700 060000          MOV #FIRST,R0
107 051464 012701 040000          MOV #SIZE,R1
108 051470          CASE R5
                                MOV R5,-(SP)
    051470 010546          ASL @SP
    051472 006316          JSR PC,L565
    051474 004737 051514
109 051500 051530          SLPAT0          ;ZEROS
110 051502 051534          SLPAT1          ;ONES
111 051504 051542          SLPAT2          ;125252
112 051506 051550          SLPAT3          ;52525
113 051510 051556          SLPAT4          ;377
114 051512 051564          SLPAT5          ;177400
115 051514          END ;OF CASE R5
                                L565:::
    051514 062616          ADD (SP)+,@SP
    051516 013646          MOV @((SP)+,-(SP)
    051520 004736          JSR PC,@((SP)+
116 051522 005037 002504          CLR READONLY
117 051526 000207          RETURN
118
119 051530 005002          SLPAT0: CLR R2
120 051532 000416          BR SLPAT
121 051534          SLPAT1: SET R2
                                MOV #-1,R2
    051534 012702 177777          BR SLPAT
122 051540 000413          SLPAT2: MOV #125252,R2
123 051542 012702 125252          BR SLPAT
124 051546 000410          SLPAT3: MOV #52525,R2
125 051550 012702 052525          BR SLPAT
126 051554 000405          SLPAT4: MOV #377,R2
127 051556 012702 000377          BR SLPAT
128 051562 000402          SLPAT5: MOV #177400,R2
129 051564 012702 177400          BR SLPAT
130 051570          SLPAT: SUBTST <<SLAVE PATTERNS>>
:*****
:*SUBTEST          SLAVE PATTERNS
:*****
131 051570          CLEAR MARGIN,MAINT
                                CLR MARGIN
    051570 005037 002112          CLR MAINT
    051574 005037 177750
132 051600 104502          CLRCR          ;CLEAR MK11 CSRS
133 051602          BMOV MTP000
    051602 004537 043710          JSR R5,BLOCK1
    051606 026376          MTP000
                                .DSABL CRF
134 051610 012737 000207 177644      MOV #207,U!PAR2          ;PUT RETURN INST AFTER WRITE ROUTINE
    
```

```
135 051616 012737 177646 002360      MOV    #L_PAR3,SUPDOADD
136 051624 004737 035204      CALL  REGCOPY
137 051630 104415      SAVREG
138 051632      IF WRITEONLY IS TRUE THEN $CALL SUPD01
    051632 005737 002502
    051636 001402
    051640 004737 026042
    051644
139 051644      IF READONLY IS TRUE THEN $CALL SUPD03
    051644 005737 002504
    051650 001402
    051652 004737 026220
    051656
140 051656 104416      RESREG
141 051660 104424      CACHOFF      ;TURN CACHE OFF
142 051662 000207      RETURN
```

TST WRITEONLY
BEQ L566
JSR PC,SUPD01
L566:::~::~

TST READONLY
BEQ L567
JSR PC,SUPD03
L567:::~::~

```

145 051664          TASK04: SUBTST <<EXECUTION CONTENTION TEST>>
:*****
:SUBTST          EXECUTION CONTENTION TEST
:*****
146 051664          SUPERVISOR          ;ENTER SUPERVISOR MODE
    051664 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
147 051672 013737 062106 002106      MOV FIRST+BANK,BANK
148 051700 104417          KERNEL          ;ENTER KERNEL MODE
149 051702 005060 002636          CLR PTYBUF(R0)
150 051706 012737 000037 002366      MOV #37,REALPAT
151 051714 012737 060000 002360      MOV #FIRST,SUPDOADD
152 051722          PUSH R0
                                MOV R0,-(SP)
153 051724 010046          ADD #PTYBUF,R0          ;R0 = PTYBUF + INDEX FOR CPU
154 051730 062700 002636          CALL SUPD03
155 051734 004737 026220          CACHOFF
156 051736          MAP #0          ;TURN CACHE OFF
                                ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
                                MOV R3,-(SP)
                                051736 010346
                                051740 012703 000000
                                051744 004737 040700
                                MOV #0,R3
                                CALL MAPPER
                                .DSABL CRF
                                MOV (SP)+,R3
157 051750 012603          POP R0
                                MOV (SP)+,R0
158 051752 012600          SUPERVISOR          ;ENTER SUPERVISOR MODE
    051754 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
159 051762 016060 002636 062636      MOV PTYBUF(R0),FIRST+PTYBUF(R0)
160 051770 104417          KERNEL          ;ENTER KERNEL MODE
161 051772 000207          RETURN
    
```

```

164 051774          TASK05: SUBTST <<PORT ARBITRATION SYMMETRY TEST>>
:*****
: *SUBTEST          PORT ARBITRATION SYMMETRY TEST
:*****
165 051774          SUPERVISOR          ;ENTER SUPERVISOR MODE
    051774 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
166 052002 013737 062106 002106      MOV FIRST+BANK,BANK
167 052010 104417          KERNEL          ;ENTER KERNEL MODE
168 052012          MAP BANK
                                MOV R3,-(SP)
    052012 010346          MOV BANK,R3
    052014 013703 002106      CALL MAPPER
    052020 004737 040700      .DSABL CRF
                                MOV (SP)+,R3
169 052024 012603          SUPERVISOR          ;ENTER SUPERVISOR MODE
    052026 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
170 052034          BMOV TSK05A,FIRST+2
    052034 004537 043730          JSR R5,BLOCK2
    052040 060002          FIRST+2
    052042 052240          TSK05A
                                .DSABL CRF
171 052044 104417          KERNEL          ;ENTER KERNEL MODE
172 052046 005060 002646          CLR PORTCOUNT(R0)
173 052052 010004          MOV R0,R4
174 052054          PUSH 100,102
                                MOV 100,-(SP)
                                MOV 102,-(SP)
    052054 013746 000100          MOV #1$,100
    052060 013746 000102          MOV #340,102
175 052064 012737 052224 000100          MOV #BIT6,LKS          ;ENABLE INTERRUPTS
176 052072 012737 000340 000102          CLR R0
177 052100 012737 000100 177546          MOV #FIRST,R1
178 052106 005000          SPL 5
179 052110 012701 060000          MOV #FIRST+2,SUPDOADD
180 052114 000235          CALL SUPDO4
181 052116 012737 060002 002360          CACHOFF          ;TURN CACHE OFF
182 052124 004737 026234          CLR LKS
183 052130 104424          SPL 7
184 052132 005037 177546          CALL SUPDO4          ;DO TEST AS BACKGND WHILE OTHER CPU'S FINISH
185 052136 000237          CACHOFF
186 052140 004737 026234          MAP #0          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
187 052144 104424          MOV R3,-(SP)
188 052146          MOV #0,R3
    052146 010346          CALL MAPPER
    052150 012703 000000      .DSABL CRF
    052154 004737 040700          MOV (SP)+,R3
    052160 012603          MOV R4,R0
189 052162 010400          SUPERVISOR          ;ENTER SUPERVISOR MODE
190 052164          BIS #BIT14,PSW          ;DO IT TO IT!
    052164 052737 040000 177776      .DSABL CRF
191 052172 016060 002646 062646      MOV PORTCOUNT(R0),FIRST+PORTCOUNT(R0)
192 052200 104417          KERNEL          ;ENTER KERNEL MODE
193 052202          CLEAR MARGIN,MAINT
    052202 005037 002112          CLR MARGIN
    052206 005037 177750          CLR MAINT
    
```

194	052212			POP	102,100	
	052212	012637	000102			MOV (SP)+,102
	052216	012637	000100			MOV (SP)+,100
195	052222	000207		RETURN		
196	052224			1\$: PUSH	R0	
	052224	010046				MOV R0,-(SP)
197	052226	010400		MOV	R4,R0	
198	052230	005260	002646	INC	PORTCOUNT(R0)	
199	052234			POP	R0	
	052234	012600				MOV (SP)+,R0
200	052236	000002		RTI		
201	052240	010505		TSK05A: MOV	R5,R5	
202	052242	010505		MOV	R5,R5	
203	052244	010505		MOV	R5,R5	
204	052246	010505		MOV	R5,R5	
205	052250	010505		MOV	R5,R5	
206	052252	010505		MOV	R5,R5	
207	052254	010505		MOV	R5,R5	
208	052256	010505		MOV	R5,R5	
209	052260	010505		MOV	R5,R5	
210	052262	010505		MOV	R5,R5	
211	052264	010505		MOV	R5,R5	
212	052266	010505		MOV	R5,R5	
213	052270	010505		MOV	R5,R5	
214	052272	010505		MOV	R5,R5	
215	052274	077017		SUB	R0,TSK05A	
216	052276	000207		RETURN		

219 052300

TASK06: SUBTST <<STOP SLAVE PROGRAM>>

 :*SUBTEST STOP SLAVE PROGRAM

220 052300 012706 000700
 221 052304 042737 001001 177572
 222 052312 005037 002676
 223 052316 005077 130446
 224 052322 000137 011720
 225
 226 052326

MOV #USESTK,SP
 BIC #BIT9!BIT0,MMRO ;DEENERGIZE MEMORY MANAGEMENT
 CLR MDISPLAY
 CLR @DISPLAY
 JMP STOPCPU

TASK07: SUBTST <<WRITE ONLY ADDRESS PATTERN IN HI BYTE>>

 :*SUBTEST WRITE ONLY ADDRESS PATTERN IN HI BYTE

227 052326
 052326 012737 177777 002502
 228 052334 000403
 229
 230 052336

SET WRITEONLY
 BR TASK7A
 MOV #-1,WRITEONLY

TASK10: SUBTST <<READ ONLY ADDRESS PATTERN IN HI BYTE>>

 :*SUBTEST READ ONLY ADDRESS PATTERN IN HI BYTE

231 052336
 052336 012737 177777 002504
 232 052344
 052344 052737 040000 177776

SET READONLY
 MOV #-1,READONLY

TASK7A: SUPERVISOR ;ENTER SUPERVISOR MODE

BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF
 MOVB FIRST+TASK+1(R0),R1
 MOV FIRST+BANK,BANK
 KERNEL ;ENTER KERNEL MODE
 IF R1 EQ #0

233 052352 116001 062617
 234 052356 013737 062106 002106
 235 052364 104417
 236 052366

TST R1
 BNE L570

052366 005701
 052370 001011
 237 052372
 052372 004537 043710
 052376 026422

BMOV MTP001 ;ADDRESS PATTERN
 JSR R5,BLOCK1
 MTP001
 .DSABL CRF
 MOV #FIRST,R0
 CLR R2
 MOV R0,R4
 MOV R2,R5
 ELSE

238 052400 012700 060000
 239 052404 005002
 240 052406 010004
 241 052410 010205
 242 052412
 052412 000413

BR L571
 L570:::~::~

243 052414
 052414 004537 043710
 052420 026454

BMOV MTP002 ;COMPLEMENT ADDRESS PATTERN
 JSR R5,BLOCK1
 MTP002
 .DSABL CRF

244 052422 012700 160000
 245 052426 012702 000001
 246 052432 012704 060000
 247 052436 012705 100001
 248 052442

MOV #LAST+2,R0
 MOV #1,R2
 MOV #FIRST,R4
 MOV #100001,R5
 END ;OF IF R1

052442
 249 052442 012701 040000
 250 052446 010103
 251 052450 012737 000207 177650

L571:::~::~
 MOV #SIZE,R1
 MOV R1,R3
 MOV #207,UIPAR4 ;RETURN INSTRUCTION

```
252 052456 012737 177652 002360      MOV    #UIPAR5,SUPDOADD      ;READ ONLY STARTING ADDRESS
253 052464                                CLEAR  MARGIN,MAINT
      052464 005037 002112                                CLR   MARGIN
      052470 005037 177750                                CLR   MAINT
254 052474 104502                                CLRCSR      ;CLEAR MK11 CSRS
255 052476                                IF WRITEONLY IS TRUE THEN $CALL SUPD01
      052476 005737 002502                                TST  WRITEONLY
      052502 001402                                BEQ  L572
      052504 004737 026042                                JSR  PC,SUPD01
      052510                                L572:::
256 052510                                IF READONLY IS TRUE THEN $CALL SUPD03
      052510 005737 002504                                TST  READONLY
      052514 001402                                BEQ  L573
      052516 004737 026220                                JSR  PC,SUPD03
      052522                                L573:::
257 052522 104424                                CACHOFF    ;TURN CACHE OFF
258 052524                                CLEAR  READONLY,WRITEONLY
      052524 005037 002504                                CLR   READONLY
      052530 005037 002502                                CLR   WRITEONLY
259 052534 000207                                RETURN
```

262 052536
263 052536 052737 040000 177776
264 052544 116001 062617
265 052550 013737 062106 002106
266 052556 104417
267 052560 072127 000011
268 052564 010100
269 052566 012701 010000
270 052572 012737 177777 002502
052600 012737 177777 002504
271 052606 004737 051530
272 052612 004737 051534
273 052616 004737 051542
274 052622 004737 051550
275 052626 004737 051556
276 052632 005037 002502
052636 005037 002504
277 052642 000207
278
279 052644

```
TASK11: SUBTST <<RUN FIVE PATTERNS IN DESIGNATED 1/4 BANK>>
:*****
:*SUBTEST      RUN FIVE PATTERNS IN DESIGNATED 1/4 BANK
:*****
SUPERVISOR      ;ENTER SUPERVISOR MODE
BIS #BIT14,PSW  ;DO IT TO IT!
.DSABL CRF
MOVB FIRST+TASK+1(R0),R1
MOV FIRST+BANK,BANK
KERNEL          ;ENTER KERNEL MODE
ASH #9,R1
MOV R1,R0
MOV #SIZE/4,R1
SET WRITEONLY,READONLY
MOV #-1,WRITEONLY
MOV #-1,READONLY
CALL SLPAT0
CALL SLPAT1
CALL SLPAT2
CALL SLPAT3
CALL SLPAT4
CLEAR WRITEONLY,READONLY
CLR WRITEONLY
CLR READONLY
RETURN
```

280 052644 052737 040000 177776
281 052652 116001 062617
282 052656 104417
283
284
285
286
287 052660 010004
288 052662 060104
289
290 052664 012705 010000
291 052670 012703 060000
292 052674 004537 043710
052700 034772
293 052702 104415
294 052704 004737 026042
295 052710 004537 043710
052714 035020
296 052716 104416
297 052720 004737 026056
298 052724 000207

```
TASK12: SUBTST <<WRITE & READ A SKEWED UP ADDRESS TEST>>
:*****
:*SUBTEST      WRITE & READ A SKEWED UP ADDRESS TEST
:*****
SUPERVISOR      ;ENTER SUPERVISOR MODE
BIS #BIT14,PSW  ;DO IT TO IT!
.DSABL CRF
MOVB FIRST+TASK+1(R0),R1
KERNEL          ;ENTER KERNEL MODE
;PAY ATTENTION NOW!
;R0 = MULTIPOINT ID # (0,2,4,6)
;R1 = OFFSET STARTING ADDRESS (0,2,4,6)
MOV R0,R4
ADD R1,R4      ;R4 IS NOW UNIQUE AMONST THE 4 CPU'S
MOV #SIZE/4,R5
MOV #FIRST,R3
BMOV MTPA32
JSR R5,BLOCK1
MTPA32
.DSABL CRF
SAVREG
CALL SUPD01
BMOV MTPB32
JSR R5,BLOCK1
MTPB32
.DSABL CRF
RESREG
CALL SUPD02
RETURN
```

301 052726

TASK13: SUBTST <<SETUP FOR ASRB TEST>>

 :*SUBTEST SETUP FOR ASRB TEST

302 052726

052726 004537 043710
 052732 053100

BMOV TSK13A
 JSR R5,BLOCK1
 TSK13A
 .DSABL CRF

303 052734 012777 177640 130000

304 052742 052737 040000 177776
 052742 052737 040000 177776

MOV #UIPAR0,@IISTVEC ;CAUSE WE SIMULATE TRAPS
 SUPERVISOR ;ENTER SUPERVISOR MODE
 BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF

305 052750 013737 062106 002106

306 052756 052760 100000 062616
 307 052764 104417

MOV FIRST+BANK,BANK
 BIS #BIT15,FIRST+TASK(R0) ;ACK
 KERNEL ;ENTER KERNEL MODE
 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 MOV R3,-(SP)

052766 010346
 052770 013703 002106
 052774 004737 040700

MOV BANK,R3
 CALL MAPPER
 .DSABL CRF

053000 012603

309 053002 023727 002114 000020
 053010 001432

IF CPUBIT NE #MASTER

MOV (SP)+,R3

310 053012 012702 060000

053016 012702 060000

FOR R2 := #FIRST TO #FIRST+34 BY #4

CMP CPUBIT,#MASTER
 BEQ L574

311 053016 012737 000001 002602

053016 012737 000001 002602
 053024 052737 040000 177776

FOR I := #1 TO #1000.

MOV #FIRST,R2
 B74:::~::~

312 053024 052737 040000 177776

053024 052737 040000 177776

SUPERVISOR ;ENTER SUPERVISOR MODE
 BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF

MOV #1,I
 B75:::~::~

313 053032 005737 060002

314 053036 001775

315

316 053040 017701 127676

317 053044 004711

318 053046 005237 002602 001750

053052 023727 002602 001750
 053060 003761

1\$:

TST FIRST+2
 BEQ 1\$
 ;DISPATCH THRU THE IIST VECTOR
 MOV @IISTVEC,R1
 CALL (R1)
 END ;OF FOR I

B75:::~::~

INC I
 CMP I,#1000.
 BLE B75
 E75:::~::~

319 053062 062702 000004 060034

053062 062702 000004 060034
 053066 020227 060034
 053072 003751

END ;OF FOR R2

ADD #4,R2
 CMP R2,#FIRST+34
 BLE B74
 E74:::~::~

320 053074 104417

321 053076 000207

322 053076 000207

323

324 053100 000240

325 053102 000240

326 053104 000240

327 053106 000240

KERNEL ;ENTER KERNEL MODE
 END ;OF IF CPUBIT

L574:::~::~

RETURN

TSK13A: NOP ;V177640
 NOP ;V177642
 NOP ;V177644
 NOP ;V177646

328	053110	000240			NOP		:V177650
329	053112	000240			NOP		:V177652
330	053114	000240			NOP		:V177654
331	053116	000240			NOP		:V177656
332	053120	000240			NOP		:V177660
333	053122				SUPERVISOR		:V177662
	053122	052737	040000	177776	BIS	#BIT14.PSW	:ENTER SUPERVISOR MODE
					.DSABL	CRF	:DO IT TO IT!
334	053130	106212			ASRB	(R2)	:V177670
335	053132	000137	053136		JMP	TSK13B	:V177672

```

338 053136      TSK13B: SUBTST <<RECOVER FROM TASK13 - TRAP FROM IIST>>
:*****
:*SUBTEST      RECOVER FROM TASK13 - TRAP FROM IIST
:*****
339 053136      ON.ERROR
053136 103021          SET      SLFLAG(R0)          ;I LOCKED IT!!!          BCC L575
340 053140          SET      SLFLAG(R0)          ;I LOCKED IT!!!          MOV #-1,SLFLAG(R0)
053140 012760 177777 002656
341 053146 163777 002730 127566      SUB      SLAVES,@IISTVEC
342 053154 163777 002730 127560      SUB      SLAVES,@IISTVEC
343 053162          IF @IISTVEC LO #UIPAR0 THEN LET @IISTVEC := #UIPAR0
053162 027727 127554 177640          CMP @IISTVEC,#UIPAR0
053170 103003          BHS L576
053172 012777 177640 127542          MOV #UIPAR0,@IISTVEC
053200          L576:~~~~~
344 053200          ELSE
053200 000414          BR L577
053202          L575:~~~~~
345 053202 005060 002656          CLR      SLFLAG(R0)          ;IT WAS LOCKED BY SOMEONE ELSE
346 053206 062777 000002 127526      ADD      #2,@IISTVEC
347 053214          IF @IISTVEC HI #UDPAR1 THEN LET @IISTVEC := #UDPAR1
053214 027727 127522 177662          CMP @IISTVEC,#UDPAR1
053222 101403          BLOS L600
053224 012777 177662 127510          MOV #UDPAR1,@IISTVEC
053232          L600:~~~~~
348 053232          END ;OF ON.NOERROR
053232          L577:~~~~~
349 053232          MAP      #0          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
053232 010346          MOV R3,-(SP)
053234 012703 000000          MOV      #0,R3
053240 004737 040700          CALL    MAPPER
          .DSABL CRF
          MOV (SP)+,R3
350 053244 012603          MOV      SLFLAG(R0),FIRST+SLFLAG(R0)
351 053254 016060 002656 062656      MAP      BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
          MOV R3,-(SP)
053254 010346          MOV      BANK,R3
053256 013703 002106          CALL    MAPPER
053262 004737 040700          .DSABL CRF
          MOV (SP)+,R3
352 053266 012603          CLR      FIRST+2
353 053270 005037 060002          RETURN
053274 000207
    
```

```

356 053276          TASK14: SUBTST <<RESET FROM ASRB TEST>>
:*****
:*SUBTEST          RESET FROM ASRB TEST
:*****
357 053276 000237          SPL          7
358 053300 012777 061216 127434      MOV          #SLAVUP,@IISTVEC
359 053306 000207          RETURN
360
361 053310          TASK15: SUBTST <<RUN SLAVE IN CSR TESTS>>
:*****
:*SUBTEST          RUN SLAVE IN CSR TESTS
:*****
362 053310 004737 016664          CALL          MKCONTROL
363 053314 000207          RETURN
364
365 053316          TASK16: SUBTST <<SET MARGIN 6, CPU HAS I/O PRIORITY>>
:*****
:*SUBTEST          SET MARGIN 6, CPU HAS I/O PRIORITY
:*****
366 053316 012737 000014 177750      MOV          #14,MAINT
367 053324 012737 000006 002112      MOV          #6,MARGIN
368 053332 000207          RETURN
369
370 053334          TASK17: SUBTST <<CREATE STALE DATA TEST>>
:*****
:*SUBTEST          CREATE STALE DATA TEST
:*****
371 053334          SUPERVISOR
053334 052737 040000 177776      BIS          #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
372 053342 013737 062106 002106      MOV          FIRST+BANK,BANK
373 053350 104417          KERNEL
374 053352          MAP          BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
                                MOV R3,-(SP)
                                053352 010346          MOV          BANK,R3
                                053354 013703 002106      CALL          MAPPER
                                053360 004737 040700      .DSABL CRF
                                MOV (SP)+,R3
375 053366          SUPERVISOR          ;ENTER SUPERVISOR MODE
053366 052737 040000 177776      BIS          #BIT14,PSW          ;DO IT TO IT!
                                .DSABL CRF
376 053374 012700 060000          MOV          #FIRST,R0
377 053400 012701 060004          MOV          #FIRST+4,R1
378 053404 022711 047507          CMP          #'GO,(R1)
379 053410 001375          BNE          1$
380 053412 012710 177777          MOV          #-1,(R0)
381 053416 104417          KERNEL
382 053420 000207          RETURN
1$:

```

D 5

385 053422

TASK20: SUBTST <<CACHE FLUSH TEST>>

 :SUBTEST CACHE FLUSH TEST

386 053422 004537 043710
 053422 053602

BMOV TSK20A
 JSR R5,BLOCK1
 TSK20A
 .DSABL CRF

387 053430 052737 040000 177776
 053430

SUPERVISOR ;ENTER SUPERVISOR MODE
 BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF

388 053436 013737 062106 002106
 389 053444 104417

MOV FIRST+BANK,BANK ;RETURN TO KERNEL MODE
 KERNEL
 MAP BANK

390 053446 010346
 053450 013703 002106
 053454 004737 040700

MOV BANK,R3 MOV R3,-(SP)
 CALL MAPPER
 .DSABL CRF

391 053462 012700 060000
 392 053466 012701 060004
 393 053472 005002

MOV #FIRST,R0
 MOV #FIRST+4,R1
 CLR R2

394
 395
 396 053474 077201

1\$: ;WAIT FOR ANY FLUSH TO FINISH
 SOB R2,1\$

398 053476 052737 040000 177776
 053476

SUPERVISOR ;ENTER SUPERVISOR MODE
 BIS #BIT14,PSW ;DO IT TO IT!
 .DSABL CRF

399 053504 004737 177640
 400 053510 005060 000004
 401 053514 104417

CALL FASTCITY
 CLR 4(R0) ;KILL THE GO SIGNAL
 KERNEL ;ENTER KERNEL MODE

402
 403
 404 053516 022701 000000
 405 053522 001411
 406 053524 012737 000000 002050
 407 053532 010137 002056
 408 053536 012737 060000 002040
 409 053544 104055

;VERIFY STALE DATA
 CMP #0,R1 ;CHECK FOR STALE DATA
 BEQ 2\$;OK - SKIP
 MOV #0,GOOD ;REPORT ERROR
 MOV R1,BAD
 MOV #FIRST,ADDRESS
 ERROR +55

410
 411
 412 053546 022702 177777
 413 053552 001411
 414 053554 012737 177777 002050
 415 053562 010237 002056
 416 053566 012737 060000 002040
 417 053574 104056

2\$: ;VERIFY THAT FLUSHING GETS RID OF STALE DATA
 CMP #-1,R2 ;CHECK FOR GOOD DATA
 BEQ 5\$;OK - SKIP
 MOV #-1,GOOD ;REPORT ERROR
 MOV R2,BAD
 MOV #FIRST,ADDRESS
 ERROR +56

418
 419 053576 104424
 420 053600 000207

5\$: CACHOFF ;TURN CACHE OFF
 RETURN

421
 422 053602

TSK20A: SUBTST <<CACHE FLUSH TEST (FOR PAR'S)>>

 :SUBTEST CACHE FLUSH TEST (FOR PAR'S)

423 053602 012737 000401 177746

MOV #BIT8!BIT0,CONTRL ;V177640 ;TURN ON CACHE & FLUSH IT!

424	053610	012710	000000		MOV	#0,(R0)		:V177646	:WRITE FRESH DATA
425	053614	005710			TST	(R0)		:V177652	:READ - CREATING A CACHE HIT
426	053616	012711	047507		MOV	#'GO,(R1)		:V177654	:SIGNAL SLAVES TO MAKE STALE DATA
427	053622	077201		1\$:	SOB	R2,1\$:V177660	:WAIT FOR SLAVES TO FINISH
428	053624	011001			MOV	(R0),R1		:V177662	:STALE DATA ---> R1
429	053626	012737	000401	177746	MOV	#BIT8!BIT0,CONTRL		:V177664	:FLUSH CACHE AGAIN
430	053634	011002			MOV	(R0),R2		:V177672	:FRESH DATA ---> R2
431	053636	000207			RETURN			:V177674	

```

434 053640 MRUNSETUP:SUBTST <<SETUP EXECUTION CONTENTION TEST>>
*****
: *SUBTEST SETUP EXECUTION CONTENTION TEST
*****
435 053640 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
053640 010346 MOV R3,-(SP)
053642 013703 002106 MOV BANK,R3
053646 004737 040700 CALL MAPPER
.DSABL CRF
MOV (SP)+,R3
436 053652 012603 MOV #FIRST,R0
437 053654 012700 060000 WHILE RO LO #LAST
053660 B76:.....
053660 020027 157776 CMP RO,#LAST
053664 103023 BHIS L601
438 053666 004737 074720 CALL $RAND
439 053672 013701 002714 MOV SEEDHI,R1
440 053676 012702 000020 MOV #16.,R2
441 053702 SUPERVISOR ;ENTER SUPERVISOR MODE
053702 052737 040000 177776 BIS #BIT14,PSW ;DO IT TO IT!
.DSABL CRF
THRU R2
442 053710 B77:.....
053710 ROL R1
443 053710 006101 ON.ERROR
444 053712 ON.ERROR
053712 103003 BCC L602
445 053714 012720 005210 MOV #5210,(R0)+ ;INC (R0) INSTRUCTION
446 053720 ELSE BR L603
053720 000402 L602:.....
053722 L603:.....
447 053722 012720 005710 MOV #5710,(R0)+ ;TST (R0) INSTRUCTION
448 053726 END ;OF ON.ERROR
053726 END ;OF THRU R2
449 053726 077210 SOB R2,B77 ; **** EIS INST.
053730 E77:.....
450 053730 104417 KERNEL ;ENTER KERNEL MODE
451 053732 END ;OF WHILE RO
053732 000752 BR B76
053734 L601:.....
053734 E76:.....
452 053734 SUPERVISOR ;ENTER SUPERVISOR MODE
053734 052737 040000 177776 BIS #BIT14,PSW ;DO IT TO IT!
.DSABL CRF
453 053742 012740 000207 MOV #207,-(R0) ;RETURN INSTRUCTION
454 053746 104417 KERNEL ;ENTER KERNEL MODE
455 053750 000207 RETURN
    
```

```

458 053752
459
460 053752 012737 177765 002676
461 053760 013777 002676 127002
462 053766 012737 000001 002106
    053774
463 053774 013737 002510 002114
464 054002 004737 042422
465 054006 012737 000020 002114
466 054014 005737 002130
    054020 001153
    054022 005737 002124
    054026 001550
467 054030 010346
    054032 013703 002106
    054036 004737 040700
    054042 012603
468 054044 052737 040000 177776
469 054052 005037 002646
    054056 005037 002650
    054062 005037 002652
    054066 005037 002654
    054072 005037 060002
470 054076 104417
471 054100 005000
    054102
472 054102 010001
473 054104 006301
474 054106 012761 000013 002616
475 054114 005200
    054116 020037 002730
    054122 003767
    054124
476 054124 004737 050726
477 054130 010346
    054132 013703 002106
    054136 004737 040700
    054142 012603
478 054144 005000
    054146
479 054146 010001
480 054150 006301
    
```

```

SUPM13: SUBTST <<MULTIPOINT ASRB TEST (IN SUPERVISOR SPACE)>>
:*****
:*SUBTEST MULTIPOINT ASRB TEST (IN SUPERVISOR SPACE)
:*****
.ENABL LSB
MOV #-13,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR BANK := #1 TO #167
    MOV #1,BANK
    B100:*****
MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE
    TST PFLAG
    BNE L604
    TST ACFLAG
    BEQ L604
    MOV R3,-(SP)
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
MOV BANK,R3
CALL MAPPER
.DSABL CRF
MOV (SP)+,R3
SUPERVISOR ;ENTER SUPERVISOR
BIS #BIT14,PSW ;DO IT TO IT!
.DSABL CRF
CLEAR PORTCOUNT,PORTCOUNT+2,PORTCOUNT+4,PORTCOUNT+6,FIRST+2
CLR PORTCOUNT
CLR PORTCOUNT+2
CLR PORTCOUNT+4
CLR PORTCOUNT+6
CLR FIRST+2
KERNEL ;ENTER KERNEL MODE
FOR R0 := #0 TO SLAVES
    CLR R0
    B101:*****
MOV R0,R1
ASL R1
MOV #13,TASK(R1) ;ORDERS TO SETUP FOR ASRB TEST
END ;OF FOR R0
    INC R0
    CMP R0,SLAVES
    BLE B101
    E101:*****
CALL TASKDO
MAP BANK
MOV R3,-(SP)
MOV BANK,R3
CALL MAPPER
.DSABL CRF
MOV (SP)+,R3
FOR R0 := #0 TO SLAVES
    CLR R0
    B102:*****
MOV R0,R1
ASL R1
    
```

```

481 054152 012737 061165 002574      MOV      #5*SECONDS,HUNGTIME
482 054160 004737 072154      2$:      CALL      QUE
483 054164      103005      ON.ERROR      BCC L605
484 054166 010037 002056      MOV      R0,BAD
485 054172 104060      ERROR      +60      ;SLAVE HUNG SYSTEM!
486 054174      000261      $RETURN ERROR
487 054200      000207      SEC
                                RTS PC
                                L605:.....
488 054200 005761 002616      TST      TASK(R1)
489 054204 100365      BPL      2$
490 054206      005200      END ;OF FOR R0
                                INC R0
                                CMP R0,SLAVES
                                BLE B102
                                E102:.....
                                ;I ASSUME THEY HAVE FINISHED COMPLETELY AFTER THIS DELAY
491 054216 005000      CLR R0
492 054216 005000      1$:      SOB R0,1$
493 054220 077001      FOR R2 := #FIRST TO #FIRST+34 BY #4
494 054222 012702 060000      MOV #FIRST,R2
                                B103:.....
495 054226 012737 000001 002602      FOR I := #1 TO #1000.
                                MOV #1,I
                                B104:.....
496 054234 052737 040000 177776      BIS      SUPERVISOR      ;ENTER SUPERVISOR MODE
                                .DSABL #BIT14,PSW      ;DO IT TO IT!
                                CRF
                                MOV #1,(R2)      ;UNLOCK TEST LOCATION
                                CALL WAKEUP
500      ;RECORD WINNER(S)
501 054252 005000      CLR R0
502 054254 005001      FOR R1 := #0 TO #6 BY #2
                                CLR R1
                                B105:.....
503 054256      IF SLFLAG(R1) IS TRUE
                                TST SLFLAG(R1)
                                BEQ L606
504 054264 005261 002646      INC PORTCOUNT(R1)
505 054270 005200      INC R0
506 054272      END ;OF IF SLFLAG
                                L606:.....
507 054272      END ;OF FOR R1
                                ADD #2,R1
                                CMP R1,#6
                                BLE B105
                                E105:.....
508
509      ;CHECK FOR ONE & ONLY ONE LOCK
510 054304 020027 000001      IF R0 NE #1
                                CMP R0,#1
                                BEQ L607
511 054312 104033      ERROR +33
512 054314      END ;OF IF R0
    
```

513	054314				END ;OF FOR I		L607:::~::~
	054314	005237	002602				INC I
	054320	023727	002602	001750			CMP I,#1000.
	054326	003742					BLE B104
	054330				END ;OF FOR R2		E104:::~::~
514	054330						ADD #4,R2
	054334	062702	000004				CMP R2,#FIRST+34
	054340	020227	060034				BLE B103
	054342	003732					E103:::~::~
515	054342	104417			KERNEL		
516	054344	004737	054472		CALL M13BAL		;CALL BALANCE TEST
517	054350				END ;OF IF PFLAG		
	054350				END ;OF FOR BANK		L604:::~::~
518	054350	005237	002106				INC BANK
	054354	023727	002106	000167			CMP BANK,#167
	054362	003604					BLE B100
	054364				FOR R0 := #1 TO SLAVES		.E100:::~::~
519	054364	012700	000001				MOV #1,R0
	054370						B106:::~::~
520	054370	010001			MOV R0,R1		
521	054372	006301			ASL R1		
522	054374	012761	000014	002616	MOV #14,TASK(R1)		;ORDERS TO RESET FROM ASRB TEST
523	054402				END ;OF FOR R0		
	054402	005200					INC R0
	054404	020037	002730				CMP R0,SLAVES
	054410	003767					BLE B106
	054412				FOR R0 := #1 TO SLAVES		E106:::~::~
524	054412	012700	000001				MOV #1,R0
	054416						B107:::~::~
525	054416	010001			MOV R0,R1		
526	054420	006301			ASL R1		
527	054422	012737	061165	002574	MOV #5*SECONDS,HUNGTIME		
528	054430	004737	072154	3\$:	CALL QUE		
529	054434				ON.ERROR		
	054434	103005					BCC L610
530	054436	010037	002056		MOV R0,BAD		
531	054442	104060			ERROR +60		;SLAVE HUNG SYSTEM!
532	054444				\$RETURN ERROR		
	054444	000261					SEC
	054446	000207					RTS PC
533	054450				END ;OF ON.ERROR		
	054450						L610:::~::~
534	054450	005761	002616		TST TASK(R1)		;WAIT FOR ALL CPU'S DONE
535	054454	100365			BPL 3\$		
536	054456				END ;OF FOR R0		
	054456	005200					INC R0
	054460	020037	002730				CMP R0,SLAVES
	054464	003754					BLE B107
	054466						E107:::~::~
537	054466				\$RETURN NOERROR		
	054466	000241					CLC
	054470	000207					RTS PC

538

.DSABL LSB

```

541 054472      M13BAL: SUBTST <<BALANCE TEST>>
:*****
:*SUBTEST      BALANCE TEST
:*****
542 054472      BEGIN ASRBALANCE
543 054472      FOR R0 := #0 TO SLAVES
544 054474      005000
545 054476      010001      MOV R0,R1
546 054500      006301      ASL R1
547 054504      013702      002646      MOV PORTCOUNT,R2
548 054510      166102      002646      SUB PORTCOUNT(R1),R2
549 054514      062702      000400      ADD #256.,R2
550 054514      005702      IF R2 LT #0 OR R2 GT #256.*2
551 054516      002403
552 054520      020227      001000
553 054524      003402
554 054526      104053      ERROR +53
555 054530      000404      LEAVE ASRBALANCE
556 054532      END ;OF IF R2
557 054532      END ;OF FOR R0
558 054532      005200
559 054534      020037      002730
560 054540      003755
561 054542      END ASRBALANCE
562 054542      RETURN
563 054542      000207
564 054544      012737      177777      060002      WAKEUP: ;WAKE UP EVERY CPU
565 054544      012737      177777      060002      WAKEUP: SET FIRST+2
566 054552      000240      ;INCLUDING MYSELF
567 054554      000240      NOP
568 054556      005000      CLR R0
569 054560      017701      126156      MOV @IISTVEC,R1
570 054564      004711      CALL (R1)
571 054566      000207      RETURN

```

B110:::~::~

CLR R0
B111:::~::~

;# HERE & IN NEXT INST IS + OR - COUNT SPEC

TST R2
BLT L611
CMP R2,#256.*2
BLE L612

L611:::~::~

BR E110

L612:::~::~

INC R0
CMP R0,SLAVES
BLE B111

E111:::~::~

E110:::~::~

MOV #-1,FIRST+2

```

568 .SBTTL ERROR DATA (SUPERVISOR) SETUP STUFF
569 054570 $PER25: LET ADDRESS := R1 - #2
    054570 010137 002040
    054574 162737 000002 002040
570 054602 IF ABORTFLAG IS FALSE
    054602 005737 002142
    054606 001007
571 054610 SUPERVISOR ;ENTER SUPERVISOR MODE
    054610 052737 040000 177776 BIS #BIT14,PSW ;DO IT TO IT!
    .DSABL CRF
    LET BAD := -2(R1)
572 054616 MOV -2(R1),BAD
    054616 016137 177776 002056
573 054624 KERNEL ;ENTER KERNEL MODE
    104417 END ;OF IF ABORTFLAG
574 054626 L613:::
575 054626 IF 177654 EQ #0
    054626 005737 177654 TST 177654
    054632 001003 BNE L614
576 054634 LET GOOD := R2
    054634 010237 002050 MOV R2,GOOD
577 054640 ELSE
    054640 000402 BR L615
    054642 L614:::
578 054642 LET GOOD := R3
    054642 010337 002050 MOV R3,GOOD
579 054646 END ;OF IF
    054646 L615:::
580 054646 JMP PERRAW
581
582 054652 PERRA3: SUBTST <<DATA WAS 3 WORDS>>
    ;*****
    ;*SUBTEST DATA WAS 3 WORDS
    ;*****
583 054652 IF BADPC EQ #0 THEN $CALL BADSTACK
    054652 005737 002026 TST BADPC
    054656 001002 BNE L616
    054660 004737 036770 JSR PC,BADSTACK
    054664 L616:::
584 054664 PUSH R0
    054664 010046 MOV R0,-(SP)
585 054666 CHK1DIS ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
    104505 READCSR
586 054670 MOV CSR,R0
    104426 SWAB R0
587 054672 MOV CSR,R0
    013700 002144 BIC #^C177,R0
588 054676 CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
    000300
589 054700 LET ADDRESS := R1
    042700 177600
590 054704 MOV R1,ADDRESS
    104503
591 054706
    010137 002040
592 054712 CLR GOOD
    005037 002050
593 054716 SUPERVISOR ;ENTER SUPERVISOR MODE
    054716 052737 040000 177776 BIS #BIT14,PSW ;DO IT TO IT!
    .DSABL CRF
594 054724 MOV (R1),BAD
    011137 002056
595 054730 MOV 2(R1),BAD2
    016137 000002 002060
596 054736 KERNEL ;ENTER KERNEL MODE
    104417
597 054740 MOVB R0,BAD3
    110037 002062
598 054744 CLRB BAD3+1
    105037 002063
    
```

599 054750 004737 065564
600 054754 104034
601 054756 012600
602 054760 000002

CALL PERBNK
ERROR +34
POP R0
RTI

MOV (SP)+,R0

```

605 054762          $PER30: LET GOOD := R1
    054762 010137 002050
606 054766          LET ADDRESS := (SP) - 16
    054766 011637 002040
    054772 163737 000016 002040
607 055000          IF ABORTFLAG IS FALSE
    055000 005737 002142
    055004 001007
608 055006          SUPERVISOR          ;ENTER SUPERVISOR MODE
    055006 052737 040000 177776      BIS #BIT14,PSW          ;DO IT TO IT!
    .DSABL CRF
    LET BAD := @ADDRESS
609 055014          KERNEL
    055014 017737 125020 002056      END ;OF IF ABORTFLAG
610 055022          JMP PERRAW
    055022 104417
611 055024
    055024
612 055024          GETDATA:SUBTST <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
    055024 000137 065330
613
614 055030          :*****
          :*SUBTEST GET DATA FROM ABORTED AREA IF POSSIBLE
          :*****
          PUSH R0,4,114
615 055030          MOV R0,-(SP)
    055030 010046          MOV 4,-(SP)
    055032 013746 000004          MOV 114,-(SP)
    055036 013746 000114
616 055042          MOV SP,GETDA1
    055042 010637 055156          MOV #1$,4
617 055046          MOV #1$,114
    055046 012737 055122 000004
618 055054          MOV ADDRESS,R0
    055054 012737 055122 000114
619 055062          BIS #BIT0,MAINT
    055062 013700 002040
620 055066          BIS #BIT11,CONTRL          ;DISABLE ABORTS
    055066 052737 000001 177750
621 055074          SUPERVISOR
    055074 052737 004000 177746      BIS #BIT14,PSW          ;DO IT TO IT!
    .DSABL CRF
622 055102          MOV (R0),BAD
    055102 052737 040000 177776
623 055110          KERNEL
    055110 011037 002056
624 055114          CLR ABORTFLAG
    055114 104417
625 055116          MOV GETDA1,SP          ;RESTORE KNOWN GOOD STACK POINTER
    055116 005037 002142          BIC #BIT11,CONTRL          ;ENABLE ABORTS
626 055122          BIC #BIT0,MAINT
    055122 013706 055156
627 055126          POP 114,4,R0
    055126 042737 004000 177746
628 055134          MOV (SP)+,114
    055134 042737 000001 177750
629 055142          MOV (SP)+,4
    055142 012637 000114
    055146 012637 000004
    055152 012600
630 055154          MOV (SP)+,R0
    055154 000207
631 055156          RETURN
    055156 000000
          GETDA1: 0
    
```

```

634 .SBTTL POWER FAIL AUTO RESTART
635 .SBTTL ROUTINE POWER DOWN AND UP
636 :*****
637 :POWER DOWN ROUTINE
638 055160 $PWRDN: IF CPUBIT NE #MASTER
                                CMP CPUBIT,#MASTER
                                BEQ L620
055160 023727 002114 000020
055166 001402
639 055170 DNSLAV: HALT
640 055172 BR DNSLAV
641 055174 END ;OF IF CPUBIT
                                L620:::
055174 012737 000376 002760
642 MOV #376,PWLOCK
643 ;SAVE CACHE & MARGIN STATUS
644 055202 PUSH CONTRL,MAINT
                                MOV CONTRL,-(SP)
                                MOV MAINT,-(SP)
055202 013746 177746
055206 013746 177750
645 055212 CACHON ;TURN CACHE ON
646 055214 MOV #SILLUP,PWRVEC ;SET FOR FAST UP
647 055222 MOV #340,PWRVEC+2 ;:PRIO:7
648 055230 PUSH R0,R1,R2,R3,R4,R5,CSRNO
                                MOV R0,-(SP)
                                MOV R1,-(SP)
                                MOV R2,-(SP)
                                MOV R3,-(SP)
                                MOV R4,-(SP)
                                MOV R5,-(SP)
                                MOV CSRNO,-(SP)
055230 010046
055232 010146
055234 010246
055236 010346
055240 010446
055242 010546
055244 013746 002256
649 ;SAVE USER PAR'S & PDR7
650 055250 MOV #177700,R0
651 055254 MOV #17,R1
652 055260 1$: PUSH -(R0)
                                MOV -(R0),-(SP)
055260 014046
653 055262 077102
654 SOB R1,1$
655 055264 ;SAVE SUPERVISOR PAR'S
656 055270 MOV #172300,R0
657 055274 2$: MOV #16,R1
                                PUSH -(R0)
                                MOV -(R0),-(SP)
055274 014046
658 055276 077102
659 055300 SOB R1,2$
                                IF RLFLAG IS TRUE THEN $CALL WOOPS
055300 005737 002134
055304 001402
055306 004737 056160
055312
660 ;COPY KERNEL MAP TO USER & SUPERVISOR
661 055312 MOV #KIPDR0,R0
662 055316 MOV #UIPDR0,R1
663 055322 MOV #SIPDR0,R2
664 055326 MOV #32,R3
665 055332 3$: MOV (R0),(R1)+
666 055334 MOV (R0)+,(R2)+
667 055336 SOB R3,3$
                                L621:::

```



```

055504 013746 170202
055510 013746 170200
699
700 055514
055514 017746 125246
701
702 055520 010637 056140
703
704 055524 012737 055536 000024
705 055532 000000
706 055534 000776

```

```

:SAVE POSSIBLE SOFTWARE SWITCH REGISTER
PUSH @SWR
:SAVE STACK POINTER
MOV SP,$SAVR6 ::SAVE SP
:NOW SET UP REAL VECTOR
MOV #SPWRUP,PWRVEC ::SET UP VECTOR
$DOWN: HALT
BR $DOWN ::HANG UP

```

```

MOV MAPH0,-(SP)
MOV MAPL0,-(SP)
MOV @SWR,-(SP)

```

```

709 .....
710 ;POWER UP ROUTINE
711 055536 005737 003014 $PWRUP: IF MULTIPOINT IS TRUE THEN JUMPTO MULTIUP
    055536 005737 003014
    055542 001402
    055544 000137 056142
    055550
    055550 012737 056134 000024
712 MOV $SILLUP,PWRVEC ;;SET FOR FAST DOWN
713 ;RESTORE STACK POINTER
714 055556 013706 056140 MOV $SAVR6,SP ;;GET SP
715 055562 005037 056140 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
716 055566 005237 056140 1$: INC $SAVR6 ;;WAIT FOR THE INC
717 055572 001375 BNE 1$ ;;OF A WORD
718 ;RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
719 055574 POP @SWR
    055574 012677 125166 MOV (SP)+,@SWR
720 ;RESTORE UNIBUS MAP
721 055600 POP MAPLO,MAPHO,MAPL36,MAPH36
    055600 012637 170200 MOV (SP)+,MAPLO
    055604 012637 170202 MOV (SP)+,MAPHO
    055610 012637 170370 MOV (SP)+,MAPL36
    055614 012637 170372 MOV (SP)+,MAPH36
722 055620 004737 042026 CALL LOWMAP ;SETUP LOWER 16K OF UNIBUS MAP
723 ;RESTORE KERNEL PAR'S & PDR'S
724 055624 012700 172340 MOV #172340,R0
725 055630 012702 172300 MOV #KIPDRO,R2
726 055634 012701 000020 MOV #16.,R1
727 055640 6$: POP (R0)+
    055640 012620 MOV (SP)+,(R0)+
728 055642 012722 077406 MOV #77406,(R2)+
729 055646 077104 SOB R1,6$
730 ;RESTORE MMR3,2,1,0
731 055650 POP MMR3,MMR2,MMR1,MMR0
    055650 012637 172516 MOV (SP)+,MMR3
    055654 012637 177576 MOV (SP)+,MMR2
    055660 012637 177574 MOV (SP)+,MMR1
    055664 012637 177572 MOV (SP)+,MMR0
732 ;RESTORE MK11 REGISTERS
733 055670 013701 002334 MOV MKCSRS,R1 ;GET CSR'S BYTE
734 055674 042701 177400 BIC #177400,R1
735 055700 BEGIN LCSRRESTORE
    055700
736 055700 FOR CSRNO := #34 DOWNT0 #0 BY #4
    055700 012737 000034 002256
    055706
737 055706 106201 ASRB R1
738 055710 ON.ERROR
    055710 103005 BCC L624
739 055712 POP CSR+2,CSR
    055712 012637 002146 MOV (SP)+,CSR+2
    055716 012637 002144 MOV (SP)+,CSR
740 055722 104425
741 055724 LOADCSR
    055724
742 055724 END ;OF ON.ERROR
    055724 105701 IFB R1 EQ #0 THEN LEAVE LCSRRESTORE
    055726 001407 L624:
    TSTB R1
    BEQ E114
    
```

743 055730
055730 162737 000004 002256
055736 023727 002256 000000
055744 002360
055746

END ;OF FOR CSRNO

SUB #4,CSRNO
CMP CSRNO,#0
BGE B115

E115:::~::~:

744 055746
055746

END LCSRRESTORE

E114:::~::~:

745
746 055746 012700 172300
747 055752 012701 177600
748 055756 012702 172200
749 055762 012703 000040
750 055766 011021
751 055770 012022
752 055772 077303

;COPY KERNEL MAP TO USER & SUPERVISOR

MOV #KIPDRO,R0
MOV #UIPDRO,R1
MOV #SIPDRO,R2
MOV #32,R3
3\$: MOV (R0),(R1)+
MOV (R0)+,(R2)+
SOB R3,3\$

ROUTINE POWER DOWN AND UP

```

754                                     ;RESTORE SUPERVISOR & USER STACK POINTERS
755 055774 POP RO                                     MOV (SP)+,RO
    055774 012600
756 055776 SUPERVISOR ;ENTER SUPERVISOR MODE
    055776 052737 040000 177776 BIS #BIT14,PSW ;DO IT TO IT!
    .DSABL CRF
    MOV RO,SSP
757 056004 010006                                     ;ENTER KERNEL MODE
758 056006 104417 KERNEL
759 056010 POP RO
    056010 012600                                     MOV (SP)+,RO
760 056012 USER
    056012 052737 140000 177776 BIS #BIT15!BIT14,PSW ;DO IT TO IT!
    .DSABL CRF
    MOV RO,USP
761 056020 010006                                     ;ENTER KERNEL MODE
762 056022 104417 KERNEL
763 ;RESTORE SUPERVISOR PAR'S
764 056024 012700 172240 MOV #172240,RO
765 056030 012701 000020 MOV #16.,R1
766 056034 012620 7$: POP (RO)+                                     MOV (SP)+,(RO)+
    056034 077102 SOB R1,7$
767 056036 077102 ;RESTORE USER PAR'S & PDR7
768 MOV #177636,RO
769 056040 012700 177636 MOV #17.,R1
770 056044 012701 000021 8$: POP (RO)+                                     MOV (SP)+,(RO)+
771 056050 012620 SOB R1,8$
772 056052 077102 ;RESTORE POSSIBLE SOFTWARE DISPLAY REGISTER
773 MOV $PATMAR,@DISPLAY
774 056054 013777 002014 124706 MOV $PATMAR,DISPREG
775 056062 013737 002014 000174 POP CSRNO,R5,R4,R3,R2,R1,RO
776 056070 012637 002256 MOV (SP)+,CSRNO
    056074 012605 MOV (SP)+,R5
    056076 012604 MOV (SP)+,R4
    056100 012603 MOV (SP)+,R3
    056102 012602 MOV (SP)+,R2
    056104 012601 MOV (SP)+,R1
    056106 012600 MOV (SP)+,RO
777 056110 012737 055160 000024 MOV #PWRDN,PWRVEC ;;SET UP THE POWER DOWN VECTOR
778 056116 104401 114503 TYPE MSG051 ;REPORT THE POWER FAILURE
    .DSABL CRF
    TYPEIT ,MSG051
779 ;RESTORE MARGIN & CACHE STATUS
780 056122 POP MAINT,CONTRL
    056122 012637 177750 MOV (SP)+,MAINT
    056126 012637 177746 MOV (SP)+,CONTRL
781 056132 000002 RTI
782 056134 000000 $ILLUP: HALT
783 056136 000776 BR $ILLUP
784 056140 000000 $SAVR6: 0
785 .EVEN ;:THE POWER UP SEQUENCE WAS STARTED
    ;: BEFORE THE POWER DOWN WAS COMPLETE
    ;:PUT THE SP HERE

```


820 056336

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

 :*SUBTEST POWER UP FROM BANK 0 TO RELOCATION

821 056336 012700 056512
 822 056342 012701 172340
 823 056346 012703 172300
 824 056352 012702 000010
 825 056356 012021
 826 056360 012723 077406
 827 056364 077204
 828 056366 012037 172516
 829 056372 012037 177576
 830 056376 012037 177574
 831 056402 012037 177572
 832 056406 013706 056140
 833 056412
 834 056412 013746 002106
 835 056416 005037 002106
 056422 010346
 056424 013703 002106
 056430 004737 040700
 056434 012603
 836 056436
 056436 052737 040000 177776
 837 056444 013737 056542 060024
 838 056452 013737 056544 060026
 839
 840
 841 056460 012700 056546
 842 056464 012701 000102
 843 056470 012702 136336
 844 056474 012022
 845 056476 077102
 846
 847 056500 104417
 848 056502
 056502 012637 002106
 849 056506 000137 055536
 850 056512 000014
 853 056542 000104

```

MOV #WOOPEND,R0
MOV #KIPAR0,R1
MOV #KIPDR0,R3
MOV #8.,R2
1$: MOV (R0)+,(R1)+
    MOV #77406,(R3)+
    SOB R2,1$
    MOV (R0)+,MMR3
    MOV (R0)+,MMR2
    MOV (R0)+,MMR1
    MOV (R0)+,MMR0
    MOV $SAVR6,SP
    PUSH BANK
                                           MOV BANK,-(SP)
    CLR BANK
    MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
                                           MOV R3,-(SP)
    MOV BANK,R3
    CALL MAPPER
    .DSABL CRF
                                           MOV (SP)+,R3
SUPERVISOR ;ENTER SUPERVISOR MODE
BIS #BIT14,PSW ;DO IT TO IT!
.DSABL CRF
MOV WOOPSAV,FIRST+PWRVEC
MOV WOOPSAV+2,FIRST+PWRVEC+2
;SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
;BMOV WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
MOV #WOOPSAV+4,R0
MOV #WOOPEND-WOOPUP/2+12.,R1
MOV #FIRST+WOOPUP,R2
2$: MOV (R0)+,(R2)+
    SOB R1,2$
    KERNEL ;ENTER KERNEL MODE
    POP BANK
                                           MOV (SP)+,BANK
    JMP $PWRUP
WOOPEND: .REPT 12.
WOOPSAV: .REPT WOOPEND-WOOPUP/2+12.+2
    
```

```

858          .SBTTL IO SUBROUTINES
859
860          .SBTTL ROUTINE TYPE
861
862          ;*****
863          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
864          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
865          ;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
866          ;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
867          ;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
868          ;*
869          ;*CALL:
870          ;*1) USING A TRAP INSTRUCTION
871          ;*      TYPE      MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
872          ;*OR
873          ;*      TYPE
874          ;*      MESADR
875          ;*
876
877 056752 105737 002452  $TYPE:  TSTB      $TPFLG          ;;IS THERE A TERMINAL?
878 056756 100407          BMI        6$          ;;BR IF NO
879 056760 010046          1$:      MOV        RO,-(SP)          ;;SAVE RO
880 056762 017600 000002  MOV        @2(SP),RO          ;;GET ADDRESS OF ASCIZ STRING
881 056766 112046          4$:      MOVVB    (RO)+,-(SP)          ;;PUSH CHARACTER TO BE TYPED ONTO STACK
882 056770 001005          BNE        7$          ;;BR IF IT ISN'T THE TERMINATOR
883 056772 005726          TST        (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
884 056774 012600          5$:      MOV        (SP)+,RO          ;;RESTORE RO
885 056776 062716 000002  6$:      ADD        #2,(SP)          ;;ADJUST RETURN PC
886 057002 000002          RTI
887 057004 122716 000011  7$:      CMPB     #HT,(SP)          ;;BRANCH IF NOT <HT>
888 057010 001002          BNE        11$
889 057012 112716 000040  11$:     MOVVB    #' ,(SP)          ;;REPLACE TAB WITH SPACE
890 057016 122716 000200  CMPB     #CRLF,(SP)          ;;BRANCH IF NOT <CRLF>
891 057022 001006          BNE        8$
892 057024 005726          TST        (SP)+          ;;POP <CR><LF> EQUIV
893 057026          TYPE          ;;TYPE A CR AND LF
894 057026 104401          TYPEIT
895 057030 003010          .DSABL   CRF
896 057032 105037 057356  $CRLF
897 057036 000753          CLRB     $CHARCNT          ;;CLEAR CHARACTER COUNT
898 057040 004737 057076  BR        4$          ;;GET NEXT CHARACTER
899 057044 123726 003002  8$:      CALL     $TYPEC          ;;GO TYPE THIS CHARACTER
900 057050 001346          9$:      CMPB     $FILLC,(SP)+          ;;IS IT TIME FOR FILLER CHARS.?
901 057052 013746 002450  BNE        4$          ;;IF NO GO GET NEXT CHAR.
902 057056 105366 000001  MOV        $NULL,-(SP)          ;;GET # OF FILLER CHARS. NEEDED
903 057062 002770          AND     AND THE NULL CHAR.
904 057064 004737 057076  10$:     DECB     1(SP)          ;;DOES A NULL NEED TO BE TYPED?
905 057070 105337 057356  BLT        9$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
906 057074 000770          CALL     $TYPEC          ;;GO TYPE A NULL
907 057076          DECB     $CHARCNT          ;;DO NOT COUNT AS A COUNT
908 057076 010146          BR        10$          ;;LOOP
909 057100 116601 000004  $TYPEC:  PUSH     R1          MOV R1,-(SP)
910 057104 010046          MOVVB    4(SP),R1
911 057106 013746 177746  PUSH     RO,CONTRL          MOV RO,-(SP)
912                                MOV CONTRL,-(SP)
  
```

```

910 057112 104424          CACHOFF          ;TURN CACHE OFF
911 057114          IF MULTIPOINT IS TRUE          TST MULTIPOINT
      057114 005737 003014          BEQ L625
      057120 001414
912 057122          IF NOIIST IS TRUE          TST NOIIST
      057122 005737 002756          BEQ L626
      057126 001403
913 057130 013700 177764          MOV SYSTID,R0
914 057134          ELSE
      057134 000406          BR L627
      057136          L626:::
915 057136 017700 123574          MOV @IISTACR,R0
916 057142 072027 177770          ASH #-8,R0
917 057146 042700 177774          BIC #^C3,R0
918 057152          END ;OF IF NOIIST          L627:::
      057152          END ;OF IF MULTIPOINT          L625:::
919 057152          IF PORTDIR NE R0 AND MULTIPOINT IS TRUE ;IF I,M NOT THE MASTER IN A MULTIPOINT SYSTEM
      057152 023700 002606          CMP PORTDIR,R0
      057156 001456          BEQ L630
      057160 005737 003014          TST MULTIPOINT
      057164 001453          BEQ L630
921 057166          IF CPUBIT EQ #SLAVE1 THEN LET R0 := #2          CMP CPUBIT,#SLAVE1
      057166 023727 002114 000040          BNE L631
      057174 001002          MOV #2,R0
      057176 012700 000002          L631:::
      057202          IF CPUBIT EQ #SLAVE2 THEN LET R0 := #4          CMP CPUBIT,#SLAVE2
922 057202 023727 002114 000100          BNE L632
      057210 001002          MOV #4,R0
      057212 012700 000004          L632:::
      057216          IF CPUBIT EQ #SLAVE3 THEN LET R0 := #6          CMP CPUBIT,#SLAVE3
923 057216 023727 002114 000200          BNE L633
      057224 001002          MOV #6,R0
      057226 012700 000006          L633:::
924 057232          MAP #0          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
      057232 010346          MOV R3,-(SP)
      057234 012703 000000          MOV #0,R3
      057240 004737 040700          CALL MAPPER
      .DSABL CRF          MOV (SP)+,R3
925 057244 012603          SUPERVISOR          ;ENTER SUPERVISOR MODE
      057246 052737 040000 177776          BIS #BIT14,PSW          ;DO IT TO IT!
      .DSABL CRF
926 057254 005760 062626          2$: TST FIRST+PTYFLAG(R0)
927 057260 100775          BMI 2$
928 057262 110160 062636          MOVB R1,FIRST+PTYBUF(R0)
929 057266          SET FIRST+PTYFLAG(R0)
      057266 012760 177777 062626          MOV #-1,FIRST+PTYFLAG(R0)
930 057274          MAP BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      057274 010346          MOV R3,-(SP)
      057276 013703 002106          MOV BANK,R3
      057302 004737 040700          CALL MAPPER
      .DSABL CRF
    
```

```

ROUTINE TYPE
057306 012603
931 057310 104417
932 057312 000405
057314
933 057314 105777 123456 3$: TSTB @$TPS
934 057320 100375 BPL 3$
935 057322 110177 123452 MOVB R1,$STPB
936 057326 END ;OF IF PORTDIR
057326
937 057326 122766 000015 000002 CMPB #CR,2(SP)
938 057334 001003 BNE 1$
939 057336 105037 057356 CLRB $CHARCNT
940 057342 000406 BR $TYPEX
941 057344 122766 000012 000002 1$: CMPB #LF,2(SP)
942 057352 001402 BEQ $TYPEX
943 057354 105227 INCB (PC)+
944 057356 000000 $CHARCNT: .WORD 0
945 057360 $TYPEX: POP CONTRL,R0,R1
057360 012637 177746 MOV (SP)+,CONTRL
057364 012600 MOV (SP)+,R0
057366 012601 MOV (SP)+,R1
946 057370 000207
947 057372
RETURN
SUPLIMIT:;!!!!!!!!!!!!!!!!THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE

```

```

MOV (SP)+,R3
:ENTER KERNEL MODE
:IF I AM THE MASTER OR WE'RE NOT MULTIPORTING
BR L634
L630:;:;:;:;:;
::WAIT UNTIL PRINTER IS READY
::LOAD CHAR TO BE TYPED INTO DATA REG.
L634:;:;:;:;:;
::IS CHARACTER A CARRIAGE RETURN?
::BRANCH IF NO
::YES--CLEAR CHARACTER COUNT
::EXIT
::IS CHARACTER A LINE FEED?
::BRANCH IF YES
::COUNT THE CHARACTER
::CHARACTER COUNT STORAGE

```

949 057372

MAST01: SUBSTST <<MULTIPOINT SLAVE STARTER>>

 :*SUBTEST MULTIPOINT SLAVE STARTER

```

950          .ENABL  LSB
951 057372 104424  CACHOFF
952 057374 012737 177777 002540  SET  NOSCOPE          ;TURN CACHE OFF
                                ;NO SCOPE LOOPS IN MULTIPROCESSOR TESTS
                                MOV #-1,NOSCOPE
953 057402 012737 177777 002676  MOV  #-1,MDISPLAY
954 057410 013777 002676 123352  MOV  MDISPLAY,@DISPLAY
955          ;PASS PSEUDO SWR TO SLAVE
956 057416 013737 002766 002674  MOV  SWR,MASWR          ;SAVE REAL SWR
957 057424 012737 000176 002766  MOV  #SWREG,SWR         ;SETUP SOFTWARE SWR
958 057432 017777 123236 123326  MOV  @MASWR,@SWR       ;FILL IT WITH REAL SWR STUFF
959 057440          IF NOIIST IS TRUE
                                TST NOIIST
                                BEQ L635
960 057446          FOR R0 := #1 TO SLAVES
                                MOV #1,R0
                                B116:;:;:;:;:
961 057452          CLR LOCK
962 057456 162737 000200 002706  SUB  #200,KSTACK
963 057464 010001          MOV  R0,R1
964 057466 006301          ASL  R1
965 057470          FOR R2 := #10. DOWNT0 #0
                                MOV #10.,R2
                                B117:;:;:;:;:
966 057474 010261 002636          MOV  R2,PTYBUF(R1)
967 057500 012703 000200          MOV  #200,R3 ;DELAY FOR THE SLAVE TO GAIN ACCESS TO CNT-DOWN
968 057504 077301 1$:          SOB  R3,1$
969 057506          END ;OF FOR R2
                                DEC  R2
                                CMP  R2,#0
                                BGE  B117
                                E117:;:;:;:;:
970 057516 012703 000200          MOV  #200,R3 ;DELAY FOR THE SLAVE TO SETUP HIS STACK
971 057522 077301 2$:          SOB  R3,2$
972 057524 012737 023542 002574  MOV  #2*SECONDS,HUNGTIME
973 057532 004737 072154 4$:          CALL  QUE
974 057536          ON.ERROR
                                BCC  L636
975 057540 010037 002056          MOV  R0,BAD
976 057544 104060          ERROR  +60          ;SLAVE HUNG SYSTEM!
977 057546 005037 003014          CLR  MULTIPOINT
978 057552 012737 002000 002706  MOV  #STACK,KSTACK
979 057560 000137 060746          JMP  MULTIEXIT
980 057564          END ;OF ON.ERROR
                                L636:;:;:;:;:
981 057564 005761 002616          TST  TASK(R1)
982 057570 100360          BPL  4$
983 057572          END ;OF FOR R0
                                INC  R0
                                CMP  R0,SLAVES
                                BLE  B116
                                E116:;:;:;:;:
984 057602          ELSE
                                BR   L637
057600 003724
057602 000516
    
```

```

057604
985 057604 013700 002742      MOV  IISTVEC,R0
986 057610 012720 061216      MOV  #SLAVUP,(R0)+
987 057614 012710 000340      MOV  #340,(R0)
988 057620      FOR R0 := #0 TO #3
    057620 005000
    057622
989 057622 005037 002670      CLR  LOCK
990 057626 162737 000200 002706      SUB  #200,KSTACK
991 057634 012701 000001      MOV  #1,R1
992 057640 072100      ASH  R0,R1
993 057642 004737 061174      CALL IISTOFF          ;INIT IIST & DISABLE BOOTS & INTERRUPTS
994 057646 012777 000000 123062      MOV  #0,@IISTACR
995 057654 010177 123060      MOV  R1,@IISTADR      ;INTERRUPT SOME CPU
996 057660 012777 000001 123052      MOV  #1,@IISTADR      ;NOW!
997
998
999 057666 012737 061165 002574      ;WAIT FOR READY FLAG
1000 057674 012777 000001 123034 6$:      MOV  #5*SECONDS,HUNGTIME
1001 057702      MOV  #1,@IISTACR
    057702 103010      ON.ERROR
1002 057704 104061      ERROR +61          ;IIST HUNG SYSTEM!
1003 057706 005037 003014      CLR  MULTIPOINT
1004 057712 012737 002000 002706      MOV  #STACK,KSTACK
1005 057720 000137 060746      JMP  MULTIEXIT
1006 057724      END ;OF ON.ERROR
    057724
1007 057724 032777 004000 123006      BIT  #B1111,@IISTADR
1008 057732 001760      BEQ  6$
1009 057734 012703 000200      MOV  #200,R3 ;DELAY FOR THE IIST TO REALLY BE READY
1010 057740 077301 3$:      SOB  R3,3$ ;AND FOR SLAVE TO GO THRU VECTOR AND SETUP HIS STACK
1011 057742      END ;OF FOR R0
    057742 005200      INC  R0
    057744 020027 000003      CMP  R0,#3
    057750 003724      BLE  B120
    057752
1012 057752      FOR R0 := #1 TO SLAVES
    057752 012700 000001
    057756
1013 057756 010001      MOV  R0,R1
1014 057760 006301      ASL  R1
1015 057762 012737 061165 002574      MOV  #5*SECONDS,HUNGTIME
1016 057770 004737 072154 5$:      CALL QUE
1017 057774      ON.ERROR
    057774 103012      BCC  L641
1018 057776 010037 002056      MOV  R0,BAD
1019 060002 104060      ERROR +60          ;SLAVE HUNG SYSTEM!
1020 060004 005037 003014      CLR  MULTIPOINT
1021 060010 012737 002000 002706      MOV  #STACK,KSTACK
1022 060016 000137 060746      JMP  MULTIEXIT
1023 060022      END ;OF ON.ERROR
    060022
1024 060022 005761 002616      TST  TASK(R1)
1025 060026 100360      BPL  5$
1026 060030      END ;OF FOR R0
    060030 005200      INC  R0
    060032 020037 002730      CMP  R0,SLAVES

```

L635:::~::~

CLR R0
B120:::~::~

BCC L640

L640:::~::~

E120:::~::~

B121:::~::~

L641:::~::~

060036 003747
060040
1027 060040
060040
1028 060040 012737 002000 002706
1029

END ;OF IF NOIIST
MOV #STACK,KSTACK
.DSABL LSB

BLE B121
E121:::~::~
L637:::~::~

1032 060046

MAST02: SUBTST <<HAVE SLAVES RUN CSR TESTS>>

:SUBTEST HAVE SLAVES RUN CSR TESTS

1033
1034 060046 012737 000000 002676
1035 060054 012777 177776 122706
1036 060062 005037 002670
1037 060066 012700 000001
060066
060072
1038 060072 012701 000020
1039 060076 072100
1040 060100 010137 002114
1041 060104 010001
1042 060106 006301
1043 060110 012761 000015 002616
1044 060116 012737 000031 002572
1045 060124 012737 177777 002574
1046 060132 004737 072154
1047 060136 103016
060136 162737 000001 002572
1048 060140 103012
060146 010037 002056
1050 060150 012737 000020 002114
1051 060154 104060
1052 060162 005037 003014
1053 060164 000137 060746
1054 060170 060174
060174
1055 060174 060174
060174
1056 060174 005761 002616
1057 060174 100354
1058 060200 060202
060202 005200
060204 020037 002730
060210 003730
060212
1060 060212 012737 000020 002114
1061

```

.ENABL  LSB
MOV     #0,MDISPLAY ;LET SLAVES SHOW INDIVIDUAL CSR TESTS IN DISPLAY
MOV     #-2,@DISPLAY ;FAKE OUT MASTER'S DISPLAY (NOT SLAVES)
CLR     LOCK
FOR RO := #1 TO SLAVES
                                MOV #1,R0
                                B122:::
MOV     #BIT4,R1
ASH     R0,R1
MOV     R1,CPUBIT
MOV     R0,R1
ASL     R1
MOV     #15,TASK(R1) ;RUN CSR TESTS COMMAND
MOV     #25,HUNGMSB
MOV     #-1,HUNGTIME ;13 SECONDS
CALL    QUE
ON.ERROR
                                BCC L642
SUB     #1,HUNGMSB
ON.ERROR
                                BCC L643
MOV     R0,BAD
MOV     #MASTER,CPUBIT
ERROR   +60 ;SLAVE HUNG SYSTEM!
CLR     MULTIPOST
JMP     MULTIEXIT
END ;OF ON.ERROR
                                L643:::
END ;OF ON.ERROR
                                L642:::
TST     TASK(R1)
BPL     1$
END ;OF FOR RO
                                INC R0
                                CMP R0,SLAVES
                                BLE B122
MOV     #MASTER,CPUBIT ;SET UP AS MASTER AGAIN
.DSABL  LSB
                                E122:::

```

1\$:

D 7

```

1064 060220      MAST03: SUBTST <<MOVE SLAVES TO THERE OWN BANKS>>
:*****
:*SUBTEST      MOVE SLAVES TO THERE OWN BANKS
:*****
1065             .ENABL  LSB
1066 060220      MOV      #-3,MDISPLAY
1067 060226      MOV      MDISPLAY,@DISPLAY
1068 060234      CLR      LOCK
1069 060240      FOR R0 := #1 TO SLAVES
                                MOV #1,R0
                                B123:*****
1070 060244      MOV      #BIT4,R1
1071 060250      ASH      R0,R1
1072 060252      MOV      R1,CPUBIT
1073 060256      BEGIN MOVELOOP
                                B124:*****
1074 060256      FOR BANK := #1 TO #167
                                MOV #1,BANK
                                B125:*****
1075 060264      CALL EXBANK
1076 060270      IF PFLAG IS FALSE AND ACFLAG IS TRUE AND BMFLAG IS FALSE
                                TST PFLAG
                                BNE L644
                                TST ACFLAG
                                BEQ L644
                                TST BMFLAG
                                BNE L644
1077 060312      INVALIDATE
1078 060314      MOV      R0,R1
1079 060316      ASL      R1
                                ;R1 = 2 OR 4 OR 6
1080 060320      MOV      #1,TASK(R1)
                                ;MOVE COMMAND
1081 060326      MOV      #5*SECONDS,HUNGTIME
1082 060334      CALL      QUE
1083 060340      ON.ERROR
                                BCC L645
1084 060342      MOV      R0,BAD
1085 060346      MOV      #MASTER,CPUBIT
1086 060354      ERROR +60
                                ;SLAVE HUNG SYSTEM!
1087 060356      CLR      MULTIPOINT
1088 060362      JMP      MULTIEXIT
1089 060366      END ;OF ON.ERROR
                                L645:*****
1090 060366      TST      TASK(R1)
1091 060372      BPL      1$
1092 060374      MOV      BANKINDEX,R1
1093 060400      BIS      #BIT15,CONFIG(R1)
1094 060406      BIC      #BIT13,CONFIG+2(R1)
                                ;INVALIDATE BACKGROUND PATTERN
1095 060414      LEAVE MOVELOOP
                                BR E124
1096 060416      END ;OF IF PFLAG
                                L644:*****
1097 060416      END ;OF FOR BANK
                                INC BANK
                                CMP BANK,#167
                                BLE B125
                                E125:*****
1098 060432      MOV R0,BAD
    
```

```
1099 060436 104062          ERROR          +62          ;NOT ENOUGH GOOD MEMORY TO TEST SLAVES
1100                                     ;ABORTING MULTIPROCESSOR TESTS
1101 060440 005037 003014          CLR MULTIPOINT
1102 060444 000137 060746          JMP MULTIEXIT
1103 060450          END MOVELOOP
1104 060450          END ;OF FOR RO          E124::::::::::
      060450 005200          INC RO
      060452 020037 002730          CMP RO,SLAVES
      060456 003672          BLE B123
1105 060460 012737 002000 002706          MOV #STACK,KSTACK          E123::::::::::
1106          ;POINT THE MASTER BACK TO THE NORMAL SWR
1107 060466 013737 002674 002766          MOV MASWR,SWR
1108          .DSABL LSB
```

```

1111 060474          SUBSTST <<RUN NORMAL MULTIPROCESSOR TESTS>>
:*****
:*SUBTEST          RUN NORMAL MULTIPROCESSOR TESTS
:*****
1112 060474 012737 000020 002114      MOV    #MASTER,CPUBIT
1113 060502 052737 000020 002510      BIS    #BIT4,ALLCPUS
1114 060510          FOR R0 := #1 TO SLAVES
                                MOV #1,R0
                                B126:;:;:;:;
1115 060514 012700 000001          ASL    ALLCPUS
1116 060520 052737 000020 002510      BIS    #BIT4,ALLCPUS
1117 060526          END ;OF FOR R0
                                INC R0
                                CMP R0,SLAVES
                                BLE B126
                                E126:;:;:;:;
1118 060536 004737 061322          CALL   MAST04          ;ACCESS PATHS TEST
1119 060542          ON.ERROR
                                BCC L646
1120 060544 103004          CLR    MULTIPORT
1121 060550 005037 003014          JMP    MULTIEXIT
1122 060554 000137 060746          END ;OF ON.ERROR
                                L646:;:;:;:;
1123 060554 004737 061706          CALL   MAST05          ;ADDRESS TEST
1124 060560          ON.ERROR
                                BCC L647
1125 060562 103004          CLR    MULTIPORT
1126 060566 005037 003014          JMP    MULTIEXIT
1127 060572 000137 060746          END ;OF ON.ERROR
                                L647:;:;:;:;
1128 060572 004737 062270          CALL   MAST06          ;ASYNCHRONOUS ADDRESS UNIQUENESS TEST
1129 060576          ON.ERROR
                                BCC L650
1130 060600 103004          CLR    MULTIPORT
1131 060604 005037 003014          JMP    MULTIEXIT
1132 060610 000137 060746          END ;OF ON.ERROR
                                L650:;:;:;:;
1133 060610 004737 062550          CALL   MAST07          ;SKEWED UP ADDRESS TEST
1134 060614          ON.ERROR
                                BCC L651
1135 060616 103004          CLR    MULTIPORT
1136 060622 005037 003014          JMP    MULTIEXIT
1137 060626 000137 060746          END ;OF ON.ERROR
                                L651:;:;:;:;
1138 060626 004737 062776          CALL   MAST10          ;PORT ARBITRATION SYMMETRY TEST
1139 060632          ON.ERROR
                                BCC L652
1140 060634 103004          CLR    MULTIPORT
1141 060640 005037 003014          JMP    MULTIEXIT
1142 060644 000137 060746          END ;OF ON.ERROR
                                L652:;:;:;:;
1143 060644 004737 063260          CALL   MAST11          ;I/O PRIORITY TEST
1144 060650          ON.ERROR
                                BCC L653
1145 060652 103004          CLR    MULTIPORT
1146 060656 005037 003014          JMP    MULTIEXIT
1147 060662 000137 060746          END ;OF ON.ERROR
    
```

```

1148 060662 004737 063732          CALL  MAST12          ;EXECUTION CONTENTION TEST L653:.....
1149 060666          ON.ERROR
1150 060666 103004          CLR  MULTIPOST      BCC L654
1151 060670 005037 003014      JMP  MULTIEXIT
1152 060700 000137 060746      END ;OF ON.ERROR
1153 060700 004737 064200      CALL  MAST13          ;ASRB TEST L654:.....
1154 060704          ON.ERROR
1155 060704 103004          CLR  MULTIPOST      BCC L655
1156 060706 005037 003014      JMP  MULTIEXIT
1157 060712 000137 060746      END ;OF ON.ERROR
1158 060716 004737 064204      CALL  MAST14          ;CACHE FLUSH TEST L655:.....
1159 060722          ON.ERROR
1160 060722 103004          CLR  MULTIPOST      BCC L656
1161 060724 005037 003014      JMP  MULTIEXIT
1162 060730 000137 060746      END ;OF ON.ERROR
1163 060734 004737 064472      CALL  MAST15          ;STOP SLAVES L656:.....
1164 060740          ON.ERROR
1165 060740 103002          CLR  MULTIPOST      BCC L657
1166 060742 005037 003014      END ;OF ON.ERROR
1167 060746          L657:.....
1167 060746 012737 000001 002106      MULTIEXIT:FOR BANK := #1 TO #167      MOV #1,BANK
1168 060754          B127:.....
1169 060760 004737 042422          CALL  EXBANK
1170 060764 013701 002110          MOV  BANKINDEX,R1
1171 060772 042761 100000 003016      BIC  #BIT15,CONFIG(R1)
1172 060772 005237 002106          END ;OF FOR BANK
1173 060776 023727 002106 000167          INC BANK
1174 061004 003763          CMP BANK,#167
1175 061006          BLE B127
1176 061006          E127:.....
1177 061006          ;MARK THE LOADERS AS PROTECTED AGAIN
1178 061006 013737 002710 002106      MOV  LOADHOME,BANK
1179 061014 004737 042422          CALL  EXBANK
1180 061020 013701 002110          MOV  BANKINDEX,R1
1181 061024 052761 100000 003016      BIS  #BIT15,CONFIG(R1)
1182 061032 005037 002676          CLR  MDISPLAY
1183 061036 005077 121726          CLR  @DISPLAY
1184 061042          SET  WRITEONLY
1185 061042 012737 177777 002502          MOV  #-1,WRITEONLY
1186 061050 004737 015104          CALL  DOBACK          ;REWRITE BACKGROUND PATTERN
1187 061054          CLEAR NOSCOPE,LOCK,WRITEONLY
1188 061054 005037 002540          CLR  NOSCOPE
1189 061060 005037 002670          CLR  LOCK
1190 061064 005037 002502          CLR  WRITEONLY
1191 061070 000137 014656          JMP  FLUSH
    
```


1221 061244

SLAVEMAP:SUBTST <<SLAVES MAP>>

.....
: *SUBTEST SLAVES MAP
:

1222

:MAP KERNEL

1223 061244 104415
1224 061246 012700 002550
1225 061252 012701 172340
1226 061256 012702 077406
1227 061262 012703 172300
1228 061266 012705 000006
1229 061272 012021
1230 061274 010223
1231 061276 077503
1232 061300 012721 177400
1233 061304 010223
1234 061306 012711 177600
1235 061312 010213
1236 061314 104420
1237 061316 104416
1238 061320 000207

SAVREG
MOV #KIO,R0
MOV #KIPAR0,R1
MOV #77406,R2
MOV #KIPDR0,R3
MOV #6,R5
1\$: MOV (R0)+,(R1)+
MOV R2,(R3)+
SOB R5,1\$
MOV #177400,(R1)+
MOV R2,(R3)+
MOV #177600,(R1)
MOV R2,(R3)
ENERGIZE
RESREG
RETURN

1241 061322

MAST04: SUBTST <<MULTIPOINT ACCESS PATHS TEST>>

 :*SUBTEST MULTIPOINT ACCESS PATHS TEST

1242
 1243 061322 012737 177774 002676
 1244 061330 013777 002676 121432
 1245 061336
 061336 005000
 061340
 1246 061340 010001
 1247 061342 006301
 1248 061344
 061344 012737 000001 002106
 061352
 1249 061352 013737 002510 002114
 1250 061360 004737 042422
 1251 061364 012737 000020 002114
 1252 061372
 061372 005737 002130
 061376 001127
 061400 005737 002124
 061404 001524
 1253 061406 104511
 1254 061410
 061410 005005
 061412
 1255 061412 010502
 1256 061414 000302
 1257 061416 062702 000002
 1258 061422 010261 002616
 1259 061426
 061426 005701
 061430 001012
 1260 061432
 061432 010046
 061434 010146
 061436 010246
 061440 010546
 1261 061442 004737 050726
 1262 061446
 061446 012605
 061450 012602
 061452 012601
 061454 012600
 1263 061456
 061456
 1264 061456 012737 061165 002574
 1265 061464 004737 072154 1\$:
 1266 061470
 061470 103005
 1267 061472 010037 002056
 1268 061476 104060
 1269 061500
 061500 000261
 061502 000207
 1270 061504

.ENABL LSB
 MOV #-4,MDISPLAY
 MOV MDISPLAY,@DISPLAY
 FOR R0 := #0 TO SLAVES

MOV R0,R1
 ASL R1
 FOR BANK := #1 TO #167

MOV ALLCPUS,CPUBIT
 CALL EXBANK
 MOV #MASTER,CPUBIT
 IF PFLAG IS FALSE AND ACFLAG IS TRUE

INVALIDATE
 FOR R5 := #0 TO #5 ;FOR PATTERNS 0-5

MOV R5,R2
 SWAB R2
 ADD #2,R2
 MOV R2,TASK(R1)
 IF R1 EQ #0

PUSH R0,R1,R2,R5

CALL TASKDO
 POP R5,R2,R1,R0

END ;OF IF R1

MOV #5*SECONDS,HUNGTIME
 CALL QUE
 ON.ERROR

MOV R0,BAD
 ERROR +60
 \$RETURN ERROR

END ;OF ON.ERROR

CLR R0
 B130:*****

MOV #1,BANK
 B131:*****

TST PFLAG
 BNE L662
 TST ACFLAG
 BEQ L662

CLR R5
 B132:*****

;WRITE ONLY PATTERN COMMAND
 ;SET TASK ORDER

TST R1
 BNE L663

MOV R0,-(SP)
 MOV R1,-(SP)
 MOV R2,-(SP)
 MOV R5,-(SP)

MOV (SP)+,R5
 MOV (SP)+,R2
 MOV (SP)+,R1
 MOV (SP)+,R0

L663:*****

BCC L664

;SLAVE HUNG SYSTEM!

SEC
 RTS PC

```

1271 061504 005761 002616          TST     TASK(R1)
1272 061510 100365                   BPL     1$
1273 061512 062702 000001          ADD     #1,R2          ;CHANGE ORDER TO READ ONLY
1274 061516 005003                   FOR R3 := #0 TO SLAVES
                                CLR R3
                                B133:;
1275 061520 010304                   MOV     R3,R4
1276 061522 006304                   ASL     R4
1277 061524 010264 002616          MOV     R2,TASK(R4)
1278 061530 005203                   END ;OF FOR R3
                                INC R3
                                CMP R3,SLAVES
                                BLE B133
                                E133:;
1279 061540 010046                   PUSH    R0,R1,R3,R4,R5
                                MOV R0,-(SP)
                                MOV R1,-(SP)
                                MOV R3,-(SP)
                                MOV R4,-(SP)
                                MOV R5,-(SP)
1280 061552 004737 050726          CALL    TASKDO
1281 061556 012605                   POP     R5,R4,R3,R1,R0
                                MOV (SP)+,R5
                                MOV (SP)+,R4
                                MOV (SP)+,R3
                                MOV (SP)+,R1
                                MOV (SP)+,R0
1282 061570 005003                   FOR R3 := #0 TO SLAVES
                                CLR R3
                                B134:;
1283 061572 010304                   MOV     R3,R4
1284 061574 006304                   ASL     R4
1285 061576 020401                   IF R4 NE R1
                                CMP R4,R1
                                BEQ L665
1286 061602 012737 061165 002574    MOV #5*SECONDS,HUNGTIME
1287 061610 004737 072154          2$:   CALL    QUE
1288 061614 103005                   ON.ERROR
                                BCC L666
1289 061616 010337 002056          MOV     R3,BAD
1290 061622 104060                   ERROR   +60
1291 061624 000261                   $RETURN ERROR
                                ;SLAVE HUNG SYSTEM!
                                SEC
                                RTS PC
1292 061630 000207                   END ;OF ON.ERROR
                                L666:;
1293 061630 005764 002616          TST     TASK(R4)
1294 061634 100365                   BPL     2$
1295 061636 000261                   END ;OF IF R4
                                L665:;
1296 061636 005203                   END ;OF FOR R3
                                INC R3
                                CMP R3,SLAVES
                                BLE B134
                                E134:;
1297 061646 003752                   END ;OF FOR R5

```

```
061646 005205
061650 020527 000005
061654 003656
1298 061656          END ;OF IF PFLAG
061656
1299 061656          END ;OF FOR BANK
061656 005237 002106
061662 023727 002106 000167
061670 003630
061672
1300 061672          END ;OF FOR R0
061672 005200
061674 020037 002730
061700 003617
061702
1301 061702          $RETURN NOERROR
061702 000241
061704 000207
1302          .DSABL LSB
```

```
INC R5
CMP R5,#5
BLE B132
E132:::
L662:::
INC BANK
CMP BANK,#167
BLE B131
E131:::
INC R0
CMP R0,SLAVES
BLE B130
E130:::
CLC
RTS PC
```

1306 061706

MAST05: SUBTST <<MULTIPOINT ADDRESS TEST>>

 :*SUBTEST MULTIPOINT ADDRESS TEST

1307
 1308 061706 012737 177773 002676
 1309 061714 013777 002676 121046
 1310 061722
 061722 005000
 061724
 1311 061724 010001
 1312 061726 006301
 1313 061730
 061730 012737 000001 002106
 061736
 1314 061736 013737 002510 002114
 1315 061744 004737 042422
 1316 061750 012737 000020 002114
 1317 061756
 061756 005737 002130
 061762 001126
 061764 005737 002124
 061770 001523
 1318 061772
 061772 005005
 061774
 1319 061774 010502
 1320 061776 000302
 1321 062000 062702 000007
 1322 062004 010261 002616
 1323 062010
 062010 005701
 062012 001012
 1324 062014
 062014 010046
 062016 010146
 062020 010246
 062022 010546
 1325 062024 004737 050726
 1326 062030
 062030 012605
 062032 012602
 062034 012601
 062036 012600
 1327 062040
 062040
 1328 062040 012737 061165 002574
 1329 062046 004737 072154
 1330 062052
 062052 103005
 1331 062054 010037 002056
 1332 062060 104060
 1333 062062
 062062 000261
 062064 000207
 1334 062066
 062066

```

.ENABL  LSB
MOV     #-5,MDISPLAY
MOV     MDISPLAY,@DISPLAY
FOR R0 := #0 TO SLAVES

                                CLR R0
                                B135:*****

MOV     R0,R1
ASL     R1
FOR BANK := #1 TO #167

                                MOV #1,BANK
                                B136:*****

MOV ALLCPUS,CPUBIT
CALL    EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE

                                TST PFLAG
                                BNE L667
                                TST ACFLAG
                                BEQ L667

FOR R5 := #0 TO #1 ;FOR ADDRESS & COMPLEMENT ADDRESS PATTERNS
                                CLR R5
                                B137:*****

MOV     R5,R2
SWAB   R2
ADD    #7,R2 ;WRITE ONLY ADDRESS PATTERN
MOV    R2,TASK(R1) ;SET TASK ORDER
IF R1 EQ #0

                                TST R1
                                BNE L670

                                PUSH R0,R1,R2,R5

                                MOV R0,-(SP)
                                MOV R1,-(SP)
                                MOV R2,-(SP)
                                MOV R5,-(SP)

                                CALL TASKDO
                                POP  R5,R2,R1,R0

                                MOV (SP)+,R5
                                MOV (SP)+,R2
                                MOV (SP)+,R1
                                MOV (SP)+,R0

                                END ;OF IF R1
                                L670:*****

MOV     #5*SECONDS,HUNGTIME
CALL    QUE
ON.ERROR

                                BCC L671

MOV     R0,BAD
ERROR +60 ;SLAVE HUNG SYSTEM!
$RETURN ERROR

                                SEC
                                RTS PC

                                END ;OF ON.ERROR
                                L671:*****
    
```

1\$:

```

MULTIPOINT ADDRESS TEST

1335 062066 005761 002616      TST    TASK(R1)
1336 062072 100365              BPL    1$
1337 062074 062702 000001      ADD    #1,R2      ;CHANGE ORDER TO READ ADDRESS PATTERNS ONLY
1338 062100 005003              FOR R3 := #0 TO SLAVES
                                CLR R3
                                B140:.....
1339 062102 010304              MOV    R3,R4
1340 062104 006304              ASL    R4
1341 062106 010264 002616      MOV    R2,TASK(R4)
1342 062112 005203              END ;OF FOR R3
                                INC R3
                                CMP R3,SLAVES
                                BLE B140
                                E140:.....
1343 062122 010046              PUSH   R0,R1,R3,R4,R5
                                MOV R0,-(SP)
                                MOV R1,-(SP)
                                MOV R3,-(SP)
                                MOV R4,-(SP)
                                MOV R5,-(SP)
1344 062134 004737 050726      CALL   TASKDO
1345 062140 012605              POP    R5,R4,R3,R1,R0
                                MOV (SP)+,R5
                                MOV (SP)+,R4
                                MOV (SP)+,R3
                                MOV (SP)+,R1
                                MOV (SP)+,R0
                                FOR R3 := #0 TO SLAVES
1346 062152 005003              CLR R3
                                B141:.....
1347 062154 010304              MOV    R3,R4
1348 062156 006304              ASL    R4
1349 062160 020401              IF R4 NE R1
                                CMP R4,R1
                                BEQ L672
1350 062164 012737 061165 002574      MOV #5*SECONDS,HUNGTIME
1351 062172 004737 072154 2$:      CALL   QUE
1352 062176 103005              ON.ERROR
                                BCC L673
1353 062200 010337 002056      MOV    R3,BAD
1354 062204 104050              ERROR  +60
1355 062206 000261              $RETURN ERROR
                                ;SLAVE HUNG SYSTEM!
                                SEC
                                RTS PC
1356 062212 005764 002616      END ;OF ON.ERROR
                                L673:.....
1357 062212 005764 002616      TST    TASK(R4)
1358 062216 100365              BPL    2$
1359 062220 003752              END ;OF IF R4
                                L672:.....
1360 062220 005203              END ;OF FOR R3
                                INC R3
                                CMP R3,SLAVES
                                BLE B141
                                E141:.....
1361 062230 005205              END ;OF FOR R5
                                INC R5

```

062232 020527 000001
062236 003656
1362 062240
062240
1363 062240
062240 005237 002106
062244 023727 002106 000167
062252 003631
062254
1364 062254
062254 005200
062256 020037 002730
062262 003620
062264
1365 062264
062264 000241
062266 000207
1366

END ;OF IF PFLAG
END ;OF FOR BANK
END ;OF FOR RO
\$RETURN NOERROR
.DSABL LSB

CMP R5,#1
BLE B137
E137:::~::~
L667:::~::~
INC BANK
CMP BANK,#167
BLE B136
E136:::~::~
INC RO
CMP RO,SLAVES
BLE B135
E135:::~::~
CLC
RTS PC

1369 062270

MAST06: SUBTST <<MULTIPOINT ASYNCHRONOUS ADDRESS UNIQUENESS TEST>>

: *SUBTEST MULTIPOINT ASYNCHRONOUS ADDRESS UNIQUENESS TEST

1370
1371 062270 012737 177772 002676
1372 062276 013777 002676 120464
1373 062304
062304 012737 000001 002106
062312
1374 062312 013737 002510 002114
1375 062320 004737 042422
1376 062324 012737 000020 002114
1377 062332
062332 005737 002130
062336 001074
062340 005737 002124
062344 001471
1378 062346
062346 005000
062350
1379 062350 010003
1380 062352 072327 000015
1381 062356 062703 060000
1382 062362
062362 005002
062364
1383 062364 062703 020000
1384 062370
062370 020327 160000
062374 001002
062376 012703 060000
062402
1385 062402
062402 010346
1386 062404 000241
1387 062406 006003
1388 062410 010201
1389 062412 006301
1390 062414 062703 000011
1391 062420 010361 002616
1392 062424
062424 012603
1393 062426
062426 005202
062430 020237 002730
062434 003753
062436
1394 062436
062436 010046
1395 062440 004737 050726
1396 062444
062444 012600
1397 062446
062446 005002
062450
1398 062450 010201

.ENABL LSB
MOV #-6,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR BANK := #1 TO #167

MOV #1,BANK
B142:::~::~

MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE

TST PFLAG
BNE L674
TST ACFLAG
BEQ L674
CLR R0

FOR R0 := #0 TO #3

;R0 := STARTING ADDRESS INDEX
B143:::~::~

MOV R0,R3
ASH #13,R3
ADD #FIRST,R3
FOR R2 := #0 TO SLAVES

CLR R2
B144:::~::~

ADD #SIZE/2,R3
IF R3 EQ #LAST+2 THEN LET R3 := #FIRST

CMP R3,#LAST+2
BNE L675
MOV #FIRST,R3
L675:::~::~

PUSH R3

MOV R3,-(SP)

CLC
ROR R3
MOV R2,R1
ASL R1
ADD #11,R3
MOV R3,TASK(R1)
POP R3

;R1 := SLAVE #

;ORDER FOR ASYNCHRONOUS ADD UNIQUE TEST

END ;OF FOR R2

MOV (SP)+,R3

INC R2
CMP R2,SLAVES
BLE B144

E144:::~::~

PUSH R0

MOV R0,-(SP)
;MASTER ASSUMES THE SLAVES ROLE

CALL TASKDO
POP R0

MOV (SP)+,R0

FOR R2 := #0 TO SLAVES

CLR R2
B145:::~::~

MOV R2,R1

1399	062452	006301			ASL R1	
1400	062454	012737	061165	002574	MOV #5*SECONDS,HUNGTIME	
1401	062462	004737	072154	1\$:	CALL QUE	
1402	062466				ON.ERROR	
	062466	103005				BCC L676
1403	062470	010237	002056		MOV R2,BAD	
1404	062474	104060			ERROR +60	:SLAVE HUNG SYSTEM!
1405	062476				\$RETURN ERROR	
	062476	000261				SEC
	062500	000207				RTS PC
1406	062502				END ;OF ON.ERROR	
	062502					L676::::::::::
1407	062502	005761	002616		TST TASK(R1)	
1408	062506	100365			BPL 1\$	
1409	062510				END ;OF FOR R2	
	062510	005202				INC R2
	062512	020237	002730			CMP R2,SLAVES
	062516	003754				BLE B145
	062520					E145::::::::::
1410	062520				END ;OF FOR R0	
	062520	005200				INC R0
	062522	020027	000003			CMP R0,#3
	062526	003710				BLE B143
	062530					E143::::::::::
1411	062530				END ;OF IF PFLAG	
	062530					L674::::::::::
1412	062530				END ;OF FOR BANK	
	062530	005237	002106			INC BANK
	062534	023727	002106	000167		CMP BANK,#167
	062542	003663				BLE B142
	062544					E142::::::::::
1413	062544				\$RETURN NOERROR	
	062544	000241				CLC
	062546	000207				RTS PC
1414					.DSABL LSB	

```

1417 062550
1418
1419 062550 012737 177771 002676
1420 062556 013777 002676 120204
1421 062564 012737 000001 002106
      062564
      062572
1422 062572 013737 002510 002114
1423 062600 004737 042422
1424 062604 012737 000020 002114
1425 062612 005737 002130
      062616 001057
      062620 005737 002124
      062624 001454
1426 062626 005002
      062630
1427 062630 005000
      062632
1428 062632 010001
1429 062634 006301
1430 062636 010203
1431 062640 000303
1432 062642 062703 000012
1433 062646 010361 002616
1434 062652 005200
      062654 020037 002730
      062660 003764
      062662
1435 062662 010246
      062664 004737 050726
1436 062664 012602
1437 062670 005000
      062672
1438 062672 010001
      062674 006301
1439 062674 012737 061165 002574
1440 062676 004737 072154 1$:
1441 062700 103005
1442 062706 010037 002056
1443 062712 104060
      062714 000261
1444 062714 010037 002056
1445 062720 000207
      062722
1446 062722 000261
      062724 000207
1447 062726
      062726
1448 062726 005761 002616

```

```

MAST07: SUBST <<MULTIPOINT SKEWED UP ADDRESS TEST>>
*****
:SUBTEST MULTIPOINT SKEWED UP ADDRESS TEST
*****
      .ENABL LSB
      MOV #-7,MDISPLAY
      MOV MDISPLAY,@DISPLAY
      FOR BANK := #1 TO #167
      MOV ALLCPUS,CPUBIT
      CALL EXBANK
      MOV #MASTER,CPUBIT
      IF PFLAG IS FALSE AND ACFLAG IS TRUE
      FOR R2 := #0 TO #6 BY #2
      FOR R0 := #0 TO SLAVES
      MOV R0,R1
      ASL R1
      MOV R2,R3
      SWAB R3
      ADD #12,R3
      MOV R3,TASK(R1)
      END ;OF FOR R0
      PUSH R2
      CALL TASKDO
      POP R2
      FOR R0 := #0 TO SLAVES
      MOV R0,R1
      ASL R1
      MOV #5*SECONDS,HUNGTIME
      CALL QUE
      ON.ERROR
      MOV R0,BAD
      ERROR +60
      $RETURN ERROR
      END ;OF ON.ERROR
      TST TASK(R1)

```

```

      MOV #1,BANK
B146:.....
      TST PFLAG
      BNE L677
      TST ACFLAG
      BEQ L677
      CLR R2
B147:.....
      CLR R0
B150:.....
;ADDRESS OFFSET (0,2,4,6)
;ORDER SKEWED UP ADDRESS TEST
      INC R0
      CMP R0,SLAVES
      BLE B150
E150:.....
      MOV R2,-(SP)
      MOV (SP)+,R2
      CLR R0
B151:.....
      BCC L700
;SLAVE HUNG SYSTEM!
      SEC
      RTS PC
L700:.....

```

1449 062732 100365
1450 062734 005200
062736 020037 002730
062742 003754
062744
1451 062744
062744 062702 000002
062750 020227 000006
062754 003725
062756
1452 062756
062756
1453 062756
062756 005237 002106
062762 023727 002106 000167
062770 003700
062772
1454 062772
062772 000241
062774 000207
1455

BPL 1\$
END ;OF FOR R0

END ;OF FOR R2

END ;OF IF PFLAG
END ;OF FOR BANK

\$RETURN NOERROR

.DSABL LSB

INC R0
CMP R0,SLAVES
BLE B151
E151:::~::~

ADD #2,R2
CMP R2,#6
BLE B147
E147:::~::~

L677:::~::~

INC BANK
CMP BANK,#167
BLE B146
E146:::~::~

CLC
RTS PC

1458 062776

```

MAST10: SUBTST <<MULTIPOINT PORT ARBITRATION SYMMETRY TEST>>
:*****
:SUBTEST      MULTIPOINT PORT ARBITRATION SYMMETRY TEST
:*****
    
```

1459

1460 062776
 062776 005737 002104
 063002 001402
 063004 000241
 063006 000207
 063010

```

    .ENABL LSB
    IF NOCLOCK IS TRUE THEN $RETURN NOERROR
    
```

```

TST NOCLOCK
BEQ L701
CLC
RTS PC
L701:*****
    
```

1461 063010 012737 177770 002676
 1462 063016 013777 002676 117744
 1463 063024 012737 000001 002106
 063032

```

MOV #-10,MDISPLAY
MOV MDISPLAY,@DISPLAY
FOR BANK := #1 TO #167
    
```

```

MOV #1,BANK
B152:*****
    
```

1464 063032 013737 002510 002114
 1465 063040 004737 042422
 1466 063044 012737 000020 002114
 1467 063052 063052 005737 002130
 063056 001070
 063060 005737 002124
 063064 001465

```

MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE
    
```

```

TST PFLAG
BNE L702
TST ACFLAG
BEQ L702
    
```

1468 063066 063066 005000
 063070

```

FOR R0 := #0 TO SLAVES
    
```

```

CLR R0
B153:*****
    
```

1469 063070 010001
 1470 063072 006301
 1471 063074 012761 000405 002616
 1472 063102 063102 005200

```

MOV R0,R1
ASL R1
MOV #405,TASK(R1) ;ORDERS FOR PORT ARBITRATION SYMMETRY TEST
END ;OF FOR R0
    
```

```

INC R0
CMP R0,SLAVES
BLE B153
E153:*****
    
```

063104 020037 002730
 063110 003767
 063112

```

CALL TASKDO
FOR R0 := #0 TO SLAVES
    
```

```

CLR R0
B154:*****
    
```

1473 063112 004737 050726
 1474 063116 063116 005000
 063120

```

MOV R0,R1
ASL R1
MOV #5*SECONDS,HUNGTIME
CALL QUE
ON.ERROR
    
```

```

BCC L703
    
```

1475 063120 010001
 1476 063122 006301
 1477 063124 012737 061165 002574
 1478 063132 004737 072154 1\$:
 1479 063136 063136 103005

```

MOV R0,BAD
ERROR +60
$RETURN ERROR
    
```

```

;SLAVE HUNG SYSTEM!
    
```

063140 010037 002056
 1481 063144 104060
 1482 063146 063146 000261
 063150 000207

```

END ;OF ON.ERROR
    
```

```

SEC
RTS PC
L703:*****
    
```

1483 063152 063152
 1484 063152 005761 002616
 1485 063156 100365
 1486 063160 063160 005200
 063162 020037 002730

```

TST TASK(R1)
BPL 1$
END ;OF FOR R0
    
```

```

INC R0
CMP R0,SLAVES
    
```


1506 063260

MAST11: SUBTST <<MULTIPORT PORT I/O PRIORITY TEST>>

:SUBTEST MULTIPORT PORT I/O PRIORITY TEST

1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517

:THIS TESTS THE ABILITY OF THE MEMORY PORT TO GIVE I/O HIGHER
:PRIORITY THEN CPU ACCESSES

:IT DOES THIS BY USING MARGIN #6 IN THE CPU'S MAINT REGISTER
:TO RAISE THE CPU PRIORITY TO THAT OF I/O.

:EACH CPU (EXCEPT ONE) HAS HIS PRIORITY RAISED AND
:A TEST IS PERFORMED TO INSURE THAT THESE CPUS ARE GRANTED MORE
:CYCLES THEN THE OTHER CPU.

:NEED LINE CLOCK
IF NOCLOCK IS TRUE THEN \$RETURN NOERROR

1518 063260
063260 005737 002104
063264 001402
063266 000241
063270 000207
063272

TST NOCLOCK
BEQ L706
CLC
RTS PC

L706:::~::~

1519
1520

:CAN'T DO UNLESS 4 CPU'S
IF SLAVES LT #3 THEN \$RETURN NOERROR

1521 063272
063272 023727 002730 000003
063300 002002
063302 000241
063304 000207
063306

CMP SLAVES,#3
BGE L707
CLC
RTS PC

L707:::~::~

1522

1523 063306 012737 177767 002676
1524 063314 013777 002676 117446

MOV #-11,MDISPLAY
MOV MDISPLAY,@DISPLAY

```

1527
1528 063322      ;FOR EACH BANK THAT IS ACCESSABLE BY ALL CPU'S
      063322 012737 000001 002106      FOR BANK := #1 TO #167
      063330
      MOV #1,BANK
      B157:.....
1529 063330 013737 002510 002114      MOV ALLCPUS,CPUBIT
1530 063336 004737 042422      CALL EXBANK
1531 063342 012737 000020 002114      MOV #MASTER,CPUBIT
1532 063350 013701 002110      MOV BANKINDEX,R1
1533 063354      IF #BIT3 OFF.IN CONFIG(R1) ;IF NO EXTERNAL INTERLEAVE
      063354 032761 000010 003016      BIT #BIT3,CONFIG(R1)
      063362 001077      BNE L710
1534 063364      IF PFLAG IS FALSE AND ACFLAG IS TRUE
      063364 005737 002130      TST PFLAG
      063370 001074      BNE L711
      063372 005737 002124      TST ACFLAG
      063376 001471      BEQ L711
1535
1536      ;EACH CPU HAS LOW PRIORITY ONCE
1537 063400      FOR I := #0 TO SLAVES
      063400 005037 002602      CLR I
      063404      B160:.....
1538 063404      FOR J := #0 TO SLAVES
      063404 005037 002604      CLR J
      063410      B161:.....
1539 063410      IF I NE J ;IF NOT THE LOW PRIORITY CPU
      063410 023737 002602 002604      CMP I,J
      063416 001416      BEQ L712
1540 063420 013701 002604      MOV J,R1
1541 063424 006301      ASL R1
1542 063426 012761 000016 002616      MOV #16,TASK(R1) ;ORDER MARGIN #6
1543 063434      IF J EQ #0
      063434 005737 002604      TST J
      063440 001003      BNE L713
1544 063442 004737 050726      CALL TASKDO
1545 063446      ELSE
      063446 000402      BR L714
      063450      L713:.....
1546 063450 004737 063674      CALL WAIT5
1547 063454      END ;OF IF J EQ #0
      063454      L714:.....
1548 063454      END ;OF IF I NE J
      063454      L712:.....
1549 063454      END ;OF FOR J := #0 TO SLAVES
      063454 005237 002604      INC J
      063460 023737 002604 002730      CMP J,SLAVES
      063466 003750      BLE B161
      063470      E161:.....
1550
1551      ;EVERY CPU WILL DO PORT ARBITRATION TEST
1552 063470      FOR RO := #0 TO SLAVES
      063470 005000      CLR RO
      063472      B162:.....
1553 063472 010001      MOV RO,R1
1554 063474 006301      ASL R1
1555 063476 012761 000005 002616      MOV #5,TASK(R1) ;ORDER PORT ARBITRATION TEST
1556 063504      END ;OF FOR RO
      063504 005200      INC RO

```



```
1575 063602 MST11A: SUBTST <<CHECK FOR PROPER SKEWING OF ACCESSES>>
;*****
;*SUBTEST CHECK FOR PROPER SKEWING OF ACCESSES
;*****
1576 063602 BEGIN NOBALANCE
063602 B164:~::~:~:
1577 ;FOR EACH CPU
1578 063602 FOR RO := #0 TO SLAVES
063602 005000 CLR RO
063604 B165:~::~:~:
1579 063604 010001 MOV RO,R1
1580 063606 006301 ASL R1
1581 063610 013703 002602 MOV I,R3
1582 063614 006303 ASL R3
1583 063616 016302 002646 MOV PORTCOUNT(R3),R2 ;LOW PRIORITY PORT
1584 063622 016104 002646 MOV PORTCOUNT(R1),R4 ;HIGH PRIORITY PORT
1585 063626 160402 SUB R4,R2
1586 063630 006204 ASR R4
1587 063632 006204 ASR R4
1588 063634 006204 ASR R4 ;R4 = 12.5% OF HIGH PRIORITY PORT
1589 063636 IF RO NE I ;IF THE CPU SELECTED IS NOT THE LOW PRIORITY ONE
063636 020037 002602 CMP RO,I
063642 001407 BEQ L715
1590 063644 IF R2 LT R4
063644 020204 CMP R2,R4
063646 002005 BGE L716
1591 063650 013737 002602 002056 MOV I,BAD
1592 063656 104066 ERROR +66
1593 063660 LEAVE NOBALANCE
063660 000404 BR E164
1594 063662 END ;OF IF R2
063662 L716:~::~:~:
1595 063662 END ;OF IF RO NE I
063662 L715:~::~:~:
1596 063662 END ;OF FOR RO
063662 005200 INC RO
063664 020037 002730 CMP RO,SLAVES
063670 003745 BLE B165
063672 E165:~::~:~:
1597 063672 END NOBALANCE
063672 E164:~::~:~:
1598 063672 000207 RETURN
1599
1600 063674 WAITS: SUBTST <<WAIT FOR 5 SECONDS FOR A SLAVE>>
;*****
;*SUBTEST WAIT FOR 5 SECONDS FOR A SLAVE
;*****
1601 .ENABL LSB
1602 063674 012737 061165 002574 MOV #5*SECONDS,HUNGTIME
1603 063702 004737 072154 1$: CALL QUE
1604 063706 ON.ERROR
063706 103005 BCC L717
1605 063710 010037 002056 MOV RO,BAD
1606 063714 104060 ERROR +60 ;SLAVE HUNG SYSTEM!
1607 063716 $RETURN ERROR
063716 000261 SEC
063720 000207 RTS PC
```

1608 063722
063722
1609 063722 005761 002616
1610 063726 100365
1611 063730 000207
1612

END ;OF ON.ERROR
TST TASK(R1)
BPL 1\$
RETURN
.DSABL LSB

L717: :::::

1615 063732

MAST12: SUBTST <<MULTIPOINT EXECUTION CONTENTION TEST>>

 :*SUBTEST MULTIPOINT EXECUTION CONTENTION TEST
 :*****

1616

1617 063732 012737 177766 002676

1618 063740 013777 002676 117022

1619 063746 012737 000001 002106

063754

1620 063754 013737 002510 002114

1621 063762 004737 042422

1622 063766 012737 000020 002114

1623 063774 005737 002130

064000 001067

064002 005737 002124

064006 001464

1624 064010 004737 053640

1625 064014 005000

064016

1626 064016 010001

1627 064020 006301

1628 064022 005061 002636

1629 064026 012761 000004 002616

1630 064034 005200

064036 020037 002730

064042 003765

064044

1631 064044 004737 050726

1632 064050 005000

064052

1633 064052 010001

1634 064054 006301

1635 064056 012737 061165 002574

1636 064064 004737 072154 1\$:

1637 064070 103005

064072 010037 002056

1638 064076 104060

1640 064100 000261

064102 000207

1641 064104

064104 005761 002616

1643 064110 100365

1644 064112 005200

064114 020037 002730

064120 003754

064122

1645 064122

064122

.ENABL LSB
 MOV #-12,MDISPLAY
 MOV MDISPLAY,@DISPLAY
 FOR BANK := #1 TO #167

MOV ALLCPUS,CPUBIT
 CALL EXBANK
 MOV #MASTER,CPUBIT
 IF PFLAG IS FALSE AND ACFLAG IS TRUE

CALL MRUNSETUP
 FOR RO := #0 TO SLAVES

MOV RO,R1
 ASL R1
 CLR PTYBUF(R1)
 MOV #4,TASK(R1)
 END ;OF FOR RO

CALL TASKDO
 FOR RO := #0 TO SLAVES

MOV RO,R1
 ASL R1
 MOV #5*SECONDS,HUNGTIME
 CALL QUE
 ON.ERROR

MOV RO,BAD
 ERROR +60
 \$RETURN ERROR

END ;OF ON.ERROR

TST TASK(R1)
 BPL 1\$
 END ;OF FOR RO

BEGIN RUNCOUNT

MOV #1,BANK
 B166:::~::~

TST PFLAG
 BNE L720
 TST ACFLAG
 BEQ L720

CLR RO
 B167:::~::~

;CLEAR INC COUNTER
 ;ORDER FOR EXECUTION CONTENTION TEST

INC RO
 CMP RO,SLAVES
 BLE B167
 E167:::~::~

CLR RO
 B170:::~::~

BCC L721
 ;SLAVE HUNG SYSTEM!

SEC
 RTS PC
 L721:::~::~

INC RO
 CMP RO,SLAVES
 BLE B170
 E170:::~::~

B171:::~::~

```
1646 064122          FOR R0 := #0 TO SLAVES
      064122 005000
      064124
1647 064124 010001      MOV      R0,R1
1648 064126 006301      ASL      R1
1649 064130 013702 002636  MOV      PTYBUF,R2
1650 064134 166102 002636  SUB      PTYBUF(R1),R2
1651 064140          IF R2 NE #0
      064140 005702          TST R2
      064142 001402          BEQ L722
1652 064144 104052          ERROR +52
1653 064146          LEAVE RUNCOUNT
      064146 000404          BR E171
1654 064150          END ;OF IF R2
      064150          END ;OF FOR R0
1655 064150          INC R0
      064150 005200          CMP R0,SLAVES
      064152 020037 002730  BLE B172
      064156 003762          E172:*****
      064160          END RUNCOUNT
1656 064160          E171:*****
      064160          END ;OF IF PFLAG
1657 064160          L720:*****
      064160          END ;OF FOR BANK
1658 064160          INC BANK
      064160 005237 002106 000167  CMP BANK,#167
      064164 023727 002106 000167  BLE B166
      064172 003670          E166:*****
1659 064174          $RETURN NOERROR
      064174 000241          CLC
      064176 000207          RTS PC
1660          .DSABL LSB
1661
1662 064200          MAST13: SUBTST <<MULTIPOINT ASRB TEST>>
      *****
      *SUBTEST      MULTIPOINT ASRB TEST
      *****
1663 064200 000137 053752  JMP      SUPM13      ;ACTUAL ROUTINE MUST BE IN SUPERVISOR SPACE!
```

1666 064204

MAST14: SUBTST <<MULTIPOINT CACHE FLUSH TEST>>

:SUBTEST MULTIPOINT CACHE FLUSH TEST

1667
1668
1669

:WARNING FAILURES HERE ARE USUALLY DUE TO A CACHE PROBLEM
:NOT A MAIN MEMORY PROBLEM!!!!!!!!!!!!!!!!!!!!!!!!!!!!
:ENABL LSB
IF #BIT3 SET.IN CACHK OR #BIT2 SET.IN CACHK

1670 064204 032737 000010 002700
064204 001004
064214 032737 000004 002700
064222 001402

BIT #BIT3,CACHK
BNE L723
BIT #BIT2,CACHK
BEQ L724

1671 064224 000241
064224 000207

\$RETURN NOERROR ;OPERATOR HAS DISABLED PART OF CACHE

L723:::::
CLC
RTS PC

1672 064230 064230

END ;OF IF #BIT3

L724:::::

1673
1674 064230 012737 177764 002676
1675 064236 013777 002676 116524

MOV #-14,MDISPLAY
MOV MDISPLAY,@DISPLAY

1676
1677 064244 005000
064244
064246

FOR R0 := #0 TO SLAVES

CLR R0
B173:::::

1678 064246 010001
1679 064250 006301

MOV R0,R1
ASL R1
FOR BANK := #1 TO #167

1680 064252 012737 000001 002106
064252
064260

MOV #1,BANK
B174:::::

1681 064260 013737 002510 002114
1682 064266 004737 042422
1683 064272 012737 000020 002114

MOV ALLCPUS,CPUBIT
CALL EXBANK
MOV #MASTER,CPUBIT
IF PFLAG IS FALSE AND ACFLAG IS TRUE

1684 064300 005737 002130
064304 001056
064306 005737 002124
064312 001453

TST PFLAG
BNE L725
TST ACFLAG
BEQ L725

1685 064314 005003
064314
064316

FOR R3 := #0 TO SLAVES

CLR R3
B175:::::

1686 064316 010304
1687 064320 006304
1688 064322 020401

MOV R3,R4
ASL R4
IF R4 EQ R1

CMP R4,R1
BNE L726

1689 064326 012764 000020 002616
1690 064334 000403

MOV #20,TASK(R4) ;CACHE FLUSH TEST COMMAND
ELSE

BR L727

1691 064336 012764 000017 002616
1692 064344 064344

MOV #17,TASK(R4) ;CREATE STALE DATA COMMAND
END ;OF IF R4

L726:::::
L727:::::

1693 064344 005203
064346 020337 002730
064352 003761

END ;OF FOR R3

INC R3
CMP R3,SLAVES
BLE B175

1717 064472

MAST15: SUBTST <<STOP SLAVES>>

 :*SUBTEST STOP SLAVES

1718

1719 064472 012737 177763 002676

1720 064500 013777 002676 116262

1721 064506 012700 000001

064506 012700 000001

064512 010001

1722 064512 010001

1723 064514 006301

1724 064516 012761 000006 002616

1725 064524 012737 023542 002574

1726 064532 004737 072154

1727 064536 103005

1728 064540 010037 002056

1729 064544 104060

1730 064546 000261

064550 000207

1731 064552

064552

1732 064552 005761 002616

1733 064556 100365

1734 064560 005200

064562 020037 002730

064566 003751

064570

1735 064570 000241

064572 000207

1736

.ENABL LSB
 MOV #-15,MDISPLAY
 MOV MDISPLAY,@DISPLAY
 FOR R0 := #1 TO SLAVES

1\$:
 MOV R0,R1
 ASL R1
 MOV #6,TASK(R1)
 MOV #2*SECONDS,HUNGTIME
 CALL QUE
 ON.ERROR

MOV R0,BAD
 ERROR +60
 \$RETURN ERROR

END ;OF ON.ERROR

TST TASK(R1)
 BPL 1\$
 END ;OF FOR R0

\$RETURN NOERROR

.DSABL LSB

MOV #1,R0
 B177:::~::~

;ORDER STOP TASK

BCC L731

;SLAVE HUNG SYSTEM!

SEC
 RTS PC

L731:::~::~

INC R0
 CMP R0,SLAVES
 BLE B177

E177:::~::~

CLC
 RTS PC

1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795

.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

1796		THEN BAD DATA IS POINTED TO BY (R1)
1797		GOOD DATA IS ZERO
1798		
1799	PERR14	TRAP
1800		BAD DATA IN R0 UNLESS ABORTED
1801		THEN BAD DATA IS POINTED TO BY (R1)
1802		GOOD DATA IS ONES
1803		
1804	PERR15	TRAP
1805		BAD DATA IN R0 UNLESS ABORTED
1806		THEN BAD DATA IS POINTED TO BY (R1)
1807		GOOD DATA IN TSTDAT
1808		
1809	PERR16	TRAP
1810		BAD DATA IN R0 UNLESS ABORTED
1811		THEN BAD DATA IS POINTED TO BY (R1)
1812		GOOD DATA IN TSTDAT+2
1813		
1814	PERR17	TRAP
1815		BAD DATA IN R0 UNLESS ABORTED
1816		THEN BAD DATA IS POINTED TO BY (R1)
1817		GOOD DATA IN R2
1818		
1819	PERR20	TRAP
1820		BAD DATA IN R0 UNLESS ABORTED
1821		THEN BAD DATA IS POINTED TO BY (R1)
1822		GOOD DATA IN R3
1823		
1824	PERR21	TRAP
1825		7 BIT BYTE TEST
1826		BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
1827		THEN BAD DATA IS POINTED TO BY (R1)
1828		GOOD DATA IS A 7 BIT BYTE ON ONES
1829		
1830	PERR22	TRAP
1831		BAD DATA IN R2 UNLESS ABORTED
1832		THEN BAD DATA IS POINTED TO BY (R1)
1833		GOOD DATA IN R0
1834		
1835	PERR23	TRAP
1836		BAD DATA IN R0 UNLESS ABORTED
1837		THEN BAD DATA IS POINTED TO BY (R1)
1838		GOOD DATA IN R4
1839		
1840	PERR24	TRAP
1841		BAD DATA IN R0 UNLESS ABORTED
1842		THEN BAD DATA IS POINTED TO BY (R2)
1843		GOOD DATA IN R3
1844		
1845	PERR25	TRAP
1846		BAD DATA POINTED TO BY -(R1)
1847		GOOD DATA IN R2 UNLESS LOC V177654 IS SET
1848		THEN GOOD DATA IS IN R3
1849		
1850	PERR26	TRAP
1851		BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
1852		GOOD DATA IS 000000,,100000,,100

1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880

```
..... 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
```

1883	064574	010437	002040		\$PER01: MOV R4,ADDRESS	
1884	064600	162737	000002	002040	SUB #2,ADDRESS	
1885	064606	010437	002056		MOV R0,BAD	
1886	064612	010537	002050		MOV R5,GOOD	
1887	064616	000137	065330		JMP PERRAW	
1888						
1889	064622	010437	002040		\$PER02: MOV R4,ADDRESS	
1890	064626	162737	000002	002040	SUB #2,ADDRESS	
1891	064634	010137	002056		MOV R1,BAD	
1892	064640	010237	002050		MOV R2,GOOD	
1893	064644	000137	065330		JMP PERRAW	
1894						
1895	064650	010137	002040		\$PER03: MOV R1,ADDRESS	
1896	064654	162737	000002	002040	SUB #2,ADDRESS	
1897	064662	010437	002050		MOV R4,GOOD	
1898	064666	016137	177776	002056	MOV -2(R1),BAD	
1899	064674	000137	065330		JMP PERRAW	
1900						
1901	064700	010037	002040		\$PER04: MOV R0,ADDRESS	
1902	064704	162737	000002	002040	SUB #2,ADDRESS	
1903	064712	010437	002056		MOV R4,BAD	
1904	064716	010237	002050		MOV R2,GOOD	
1905	064722	000137	065330		JMP PERRAW	
1906						
1907	064726	010237	002050		PERR05: MOV R2,GOOD	
1908	064732	014037	002056		PERA05: MOV -(R0),BAD	
1909	064736	010037	002040		MOV R0,ADDRESS	
1910	064742	062700	000002		ADD #2,R0	;RESTORE R0
1911	064746	004737	036770		CALL BADSTACK	
1912	064752	000207			RETURN	
1913						
1914	064754	005037	002050		PERR06: CLR GOOD	
1915	064760	000764			BR PERA05	
1916						
1917	064762	010137	002040		\$PER07: MOV R1,ADDRESS	
1918	064766	010237	002056		MOV R2,BAD	
1919	064772	013737	002340	002050	MOV DATBUF,GOOD	
1920	065000	000137	065330		JMP PERRAW	
1921						
1922	065004				\$PER10: LET ADDRESS := R1 + #2	
	065004	010137	002040			MOV R1,ADDRESS
	065010	062737	000002	002040		ADD #2,ADDRESS
1923	065016				LET BAD := R2	
	065016	010237	002056			MOV R2,BAD
1924	065022				LET GOOD := DATBUF+2	
	065022	013737	002342	002050		MOV DATBUF+2,GOOD
1925	065030	000137	065330		JMP PERRAW	
1926						
1927	065034				\$PER11: LET ADDRESS := R1	
	065034	010137	002040			MOV R1,ADDRESS
1928	065040				LET BAD := R0	
	065040	010037	002056			MOV R0,BAD
1929	065044				LET GOOD := #0	
	065044	005037	002050			CLR GOOD
1930	065050	000137	065402		JMP PERRAB	
1931						
1932	065054				\$PER12: LET ADDRESS := R1	

1933 065054 010137 002040
065060
1934 065060 010037 002056
065064
1935 065064 012737 000377 002050
065072 000137 065402

LET BAD := R0
LET GOOD := #377
JMP PERRAB

MOV R1,ADDRESS
MOV R0,BAD
MOV #377,GOOD

1938	065076			\$PER13: LET ADDRESS := R1	
	065076	010137	002040		MOV R1,ADDRESS
1939	065102			LET BAD := R0	
	065102	010037	002056		MOV R0,BAD
1940	065106			LET GOOD := #0	
	065106	005037	002050		CLR GOOD
1941	065112	000137	065330	JMP PERRAW	
1942					
1943	065116			\$PER14: LET ADDRESS := R1	
	065116	010137	002040		MOV R1,ADDRESS
1944	065122			LET BAD := R0	
	065122	010037	002056		MOV R0,BAD
1945	065126			LET GOOD := ONES	
	065126	013737	002726 002050		MOV ONES,GOOD
1946	065134	000137	065330	JMP PERRAW	
1947					
1948	065140			\$PER15: LET ADDRESS := R1	
	065140	010137	002040		MOV R1,ADDRESS
1949	065144			LET BAD := R0	
	065144	010037	002056		MOV R0,BAD
1950	065150			LET GOOD := TSTDAT	
	065150	013737	002344 002050		MOV TSTDAT,GOOD
1951	065156	000137	065330	JMP PERRAW	
1952					
1953	065162			\$PER16: LET ADDRESS := R1	
	065162	010137	002040		MOV R1,ADDRESS
1954	065166			LET BAD := R0	
	065166	010037	002056		MOV R0,BAD
1955	065172			LET GOOD := TSTDAT+2	
	065172	013737	002346 002050		MOV TSTDAT+2,GOOD
1956	065200	000453		BR PERRAW	
1957					
1958	065202			\$PER17: LET ADDRESS := R1	
	065202	010137	002040		MOV R1,ADDRESS
1959	065206			LET BAD := R0	
	065206	010037	002056		MOV R0,BAD
1960	065212			LET GOOD := R2	
	065212	010237	002050		MOV R2,GOOD
1961	065216	000444		BR PERRAW	
1962					
1963	065220			\$PER20: LET ADDRESS := R1	
	065220	010137	002040		MOV R1,ADDRESS
1964	065224			LET BAD := R0	
	065224	010037	002056		MOV R0,BAD
1965	065230			LET GOOD := R3	
	065230	010337	002050		MOV R3,GOOD
1966	065234	000435		BR PERRAW	
1967					
1968	065236			\$PER21: LET ADDRESS := R1	
	065236	010137	002040		MOV R1,ADDRESS
1969	065242			LET BAD := R0	
	065242	010037	002056		MOV R0,BAD
1970	065246			LET GOOD := #177	
	065246	012737	000177 002050		MOV #177,GOOD
1971	065254	000477		BR PERRAW	
1972					
1973	065256			\$PER22: LET ADDRESS := R1	

1974	065256	010137	002040			MOV R1,ADDRESS
	065262			LET BAD := R2		MOV R2,BAD
1975	065262	010237	002056			
	065266			LET GOOD := R0		MOV R0,GOOD
1976	065266	010037	002050			
	065272	000416		BR PERRAW		
1977						
1978	065274			\$PER23: LET ADDRESS := R1		MOV R1,ADDRESS
	065274	010137	002040			MOV R0,BAD
1979	065300			LET BAD := R0		MOV R4,GOOD
	065300	010037	002056			
1980	065304			LET GOOD := R4		
	065304	010437	002050			
1981	065310	000407		BR PERRAW		
1982						
1983	065312			\$PER24: LET ADDRESS := R2		MOV R2,ADDRESS
	065312	010237	002040			MOV R0,BAD
1984	065316			LET BAD := R0		MOV R3,GOOD
	065316	010037	002056			
1985	065322			LET GOOD := R3		
	065322	010337	002050			
1986	065326	000400		BR PERRAW		

1988 065330
 1989 065330 004737 065564
 1990 065334 005737 002142
 065334 001402
 065340 004737 055030
 065346
 1991 065346 005737 002026
 065346 001002
 065352 004737 036770
 065354 004737 065540
 065360
 1992 065360 004737 065540
 1993 065364 005737 002142
 065364 001002
 065370 104011
 1994 065372 104011
 1995 065374 000401
 065374 000401
 065376
 1996 065376 104012
 1997 065400
 065400
 1998 065400 000002
 1999
 2000 065402

```

PERRAW: SUBTST <<DATA WAS A WORD>>
:*****
:*SUBTEST      DATA WAS A WORD
:*****
CALL PERBNK
IF ABORTFLAG IS TRUE THEN $CALL GETDATA

TST ABORTFLAG
BEQ L732
JSR PC,GETDATA
L732:*****

IF BADPC EQ #0 THEN $CALL BADSTACK

TST BADPC
BNE L733
JSR PC,BADSTACK
L733:*****

CALL PERXOR
IF ABORTFLAG IS FALSE

TST ABORTFLAG
BNE L734

ERROR +11
ELSE

BR L735
L734:*****

ERROR +12
END ;OF IF ABORTFLAG

L735:*****

RTI
    
```

2001 065402 004737 065564
 2002 065406 005737 002142
 065406 001402
 065412 004737 055030
 065414 004737 055030
 065420
 2003 065420 005737 002026
 065420 001002
 065424 004737 036770
 065426 004737 065540
 065432
 2004 065432 004737 065540
 2005 065436 005737 002142
 065436 001002
 065442 104014
 2006 065444 104014
 2007 065446 000401
 065446 000401
 065450
 2008 065450 104015
 2009 065452
 065452
 2010 065452 000002

```

PERRAB: SUBTST <<DATA WAS A BYTE>>
:*****
:*SUBTEST      DATA WAS A BYTE
:*****
CALL PERBNK
IF ABORTFLAG IS TRUE THEN $CALL GETDATA

TST ABORTFLAG
BEQ L736
JSR PC,GETDATA
L736:*****

IF BADPC EQ #0 THEN $CALL BADSTACK

TST BADPC
BNE L737
JSR PC,BADSTACK
L737:*****

CALL PERXOR
IF ABORTFLAG IS FALSE

TST ABORTFLAG
BNE L740

ERROR +14
ELSE

BR L741
L740:*****

ERROR +15
END ;OF IF ABORTFLAG

L741:*****

RTI
    
```

2013 065454

PERRA7: SUBSTST <<DATA WAS A 7 BIT BYTE>>
:*****
:*SUBTEST DATA WAS A 7 BIT BYTE
:*****
IF BADPC EQ #0 THEN \$CALL BADSTACK

2014 065454 005737 002026
065454 001002
065460 004737 036770
065462 004737 065540
065466 004737 065564
2015 065466 004737 065540
2016 065472 004737 065564
2017 065476 104022
2018 065500 000002
2019

TST BADPC
BNE L742
JSR PC,BADSTACK
L742:::~::~

2020 065502 012737 100000 002052
065502 000100 002054
2021 065510 012737 000100 002054
065510 000137 054652
2022 065516 000137 054652
2023
2024 065522 005037 002052
2025 065526 012737 000077 002054
065526 000137 054652
2026 065534 000137 054652
2027
2028 065540

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

MOV #100000,GOOD2
MOV #100,GOOD3
MOV #077,GOOD3

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

2029 065540 010046
065540 013700 002050
2030 065542 013700 002056 002064
2031 065546 013737 002056 002064
2032 065554 074037 002064
2033 065560 012600
065560 000207
2034 065562 000207

PUSH R0
MOV GOOD,R0
MOV BAD,BAD XOR
XOR R0,BAD XOR
POP R0
RETURN

MOV R0,-(SP)
MOV (SP)+,R0

2037 065564

PERBNK: SUBTST<<LOG ERROR ON BAD BANK>>

 :SUBTEST LOG ERROR ON BAD BANK

2038

:WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE

2039 065564

PUSH R0,R1

MOV R0,-(SP)

MOV R1,-(SP)

065564 010046

065566 010146

2040 065570 013701 002106

MOV BANK,R1

2041 065574 070127 000006

MUL #6,R1

2042 065600 052761 004000 003016

BIS #BIT11,CONFIG(R1)

2043 065606 062701 000005

ADD #5,R1

:POINT TO MAX ERROR BYTE

2044 065612 105261 003016

INCB CONFIG(R1)

:BUMP BANK COUNTER

2045 065616 001002

BNE 12\$

:NO OVERFLOW - SKIP

2046 065620 105361 003016

DECB CONFIG(R1)

:SET BACK TO 255.

2047 065624 126137 003016 002702 12\$:

CMPB CONFIG(R1),ERRMAX

:IS IT PAST MAX?

2048 065632 101403

BLOS 11\$

:NO - SKIP

2049 065634

SET TOOMANY

:YES

065634 012737 177777 002472

MOV #-1,TOOMANY

2050 065642

11\$: POP R1,R0

MOV (SP)+,R1

MOV (SP)+,R0

065642 012601

065644 012600

2051 065646 000207

RETURN

2052

2053 065650 010137 002040

PERECC: MOV R1,ADDRESS

2054 065654 010037 002056

MOV R0,BAD

2055 065660

IF R1 EQ TESTADD

CMP R1,TESTADD

BNE L743

065660 020137 002506

065664 001004

2056 065666 013737 002344 002050

MOV TSTDAT,GOOD

2057 065674

ELSE

BR L744

065674 000403

L743:.....

2058 065676 013737 002346 002050

MOV TSTDAT+2,GOOD

2059 065704

END ;OF IF R1

L744:.....

065704

2060 065704 004737 065540

CALL PERXOR

2061 065710

SET HEADER

MOV #-1,HEADER

065710 012737 177777 002724

2062 065716 000207

RETURN

2063

2064 065720 \$PER31: IF BADPC EQ #0 THEN \$CALL BADSTACK

TST BADPC

BNE L745

JSR PC,BADSTACK

065720 005737 002026

065724 001002

065726 004737 036770

065732

L745:.....

2065 065732 004737 065650

CALL PERECC

2066 065736

IF REALPAT EQ #11

CMP REALPAT,#11

BNE L746

065736 023727 002366 000011

065744 001001

2067 065746 104042

ERROR +42

2068 065750

END ;OF IF REALPAT

L746:.....

065750

2069 065750

IF REALPAT EQ #14

CMP REALPAT,#14

BNE L747

065750 023727 002366 000014

065756 001001

2070 065760 104045

ERROR +45


```

2082 066016          $PER32: IF BADPC EQ #0 THEN $CALL BADSTACK
      066016 005737 002026
      066022 001002
      066024 004737 036770
      066030
                                TST BADPC
                                BNE L752
                                JSR PC,BADSTACK
                                L752:.....

2083 066030 010137 002040          MOV R1,ADDRESS
2084 066034 010037 002056          MOV R0,BAD
2085 066040 010237 002050          MOV R2,GOOD
2086 066044          SET HEADER
                                MOV #-1,HEADER
      066044 012737 177777 002724
2087 066052 104043          ERROR +43
2088 066054          SET HEADER
                                MOV #-1,HEADER
      066054 012737 177777 002724
2089 066062 000002          RTI
2090
2091 066064          $PER33: IF BADPC EQ #0 THEN $CALL BADSTACK
      066064 005737 002026
      066070 001002
      066072 004737 036770
      066076
                                TST BADPC
                                BNE L753
                                JSR PC,BADSTACK
                                L753:.....

2092 066076 010137 002040          MOV R1,ADDRESS
2093 066102 010037 002056          MOV R0,BAD
2094 066106 105037 002057          CLRB BAD+1
2095 066112 012737 000377 002050          MOV #377,GOOD
2096 066120 004737 065540          CALL PERXOR
2097 066124          SET HEADER
                                MOV #-1,HEADER
      066124 012737 177777 002724
2098 066132 104044          ERROR +44
2099 066134          SET HEADER
                                MOV #-1,HEADER
      066134 012737 177777 002724
2100 066142 000002          RTI
2101
2102 066144          $PER34:IF BADPC EQ #0 THEN $CALL BADSTACK
      066144 005737 002026
      066150 001002
      066152 004737 036770
      066156
                                TST BADPC
                                BNE L754
                                JSR PC,BADSTACK
                                L754:.....

2103 066156          IF #BIT15!BIT4 OFF.IN CSR
      066156 032737 100020 002144
                                BIT #BIT15!BIT4,CSR
                                BNE L755
      066164 001002
2104 066166 104016          ERROR +16          ;NO SBE OR DBE
2105 066170          ELSE
      066170 000401
                                BR L756
      066172
                                L755:.....
2106 066172 104001          ERROR +1          ;EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
2107 066174          END ;OF IF #BIT15!BIT4
                                L756:.....
      066174
2108 066174 000002          RTI
2109
2110          ;DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG
2111 066176 004737 065564          $PER35: CALL PERBNK
2112 066202 004737 036770          CALL BADSTACK
2113 066206 013737 002036 002056          MOV BADPSW,BAD
2114 066214 012737 000012 002050          MOV #12,GOOD
2115 066222 104054          ERROR +54
2116 066224 062706 000004          ADD #4,SP          ;FIX STACK FROM TRAP
2117 066230 000207          RETURN          ;ABORTING TEST
    
```

ROUTINE SCOPE HANDLER

2120
2121
2122
2123
2124
2125
2126
2127
2128
2129 066232 005237 075144
2130 066236
066236 002004
2131 066240 005037 075144
2132 066244 105237 075146
2133 066250
066250
2134 066250 104410
2135 066252
066252 023727 002114 000020
066260 001016
2136 066262
066262 005737 002626
066266 001011
066270 005737 002630
066274 001006
066276 005737 002632
066302 001003
066304 005737 002634
066310 001402
066312
2137 066312 004737 072154
2138 066316
066316
2139 066316
066316
2140 066316
066316 005737 002520
066322 001412
066324 032777 000400 114434
066332 001406
2141 066334 005037 002520
2142 066340
066340 104401 117035
2143 066344 000137 043522
2144 066350
066350
2145 066350
066350 005737 002540
066354 001401
2146 066356 000002
2147 066360
066360
2148 066360
066360 032777 040000 114400
066366 001045
2149

```

.SBTTL ROUTINE SCOPE HANDLER
*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE DISPLAY DATA INTO THE DISPLAY REGISTER
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW9=1 LOOP ON ERROR
*CALL
*
SCOPE      ;:SCOPE=IOT
$SCOPE: INC  $DEVCT      ;TELL APT WE ARE ALIVE
IF RESULT IS LT
                                BGE L757
                                CLR  $DEVCT
                                INCB $UNIT
                                END ;OF IF RESULT
                                L757:::
                                CKSWR      ;:TEST FOR CHANGE IN SOFT-SWR
                                IF CPUBIT EQ #MASTER ;IF I'M THE MASTER
                                CMP CPUBIT,#MASTER
                                BNE L760
                                IF PTYFLAG IS TRUE OR PTYFLAG+2 IS TRUE OR PTYFLAG+4 IS TRUE OR PTYFLAG+6 IS TRUE
                                TST PTYFLAG
                                BNE L761
                                TST PTYFLAG+2
                                BNE L761
                                TST PTYFLAG+4
                                BNE L761
                                TST PTYFLAG+6
                                BEQ L762
                                L761:::
                                CALL QUE
                                END ;OF IF PTYFLAG
                                L762:::
                                END ;OF IF CPUBIT
                                L760:::
                                IF STOPOK IS TRUE AND #SW8 SET.IN @SWR
                                TST STOPOK
                                BEQ L763
                                BIT #SW8,@SWR
                                BEQ L763
                                CLR  STOPOK
                                TYPE MSG112      ;PROGRAM HALTING BECAUSE OF SWITCH #8
                                TYPEIT ,MSG112
                                .DSABL CRF
                                JMP  EXIT
                                END ;OF IF STOPOK
                                L763:::
                                IF NOSCOPE IS TRUE
                                TST NOSCOPE
                                BEQ L764
                                RTI
                                END ;OF IF NOSCOPE
                                L764:::
                                1$: IF #SW14 SET.IN @SWR THEN GOTO $OVER
                                BIT #SW14,@SWR
                                BNE $OVER
;*****START OF CODE FOR THE XOR TESTER*****

```

```

2150 066370 000421          $XTSTR: BR      2$          :: IF RUNNING ON THE 'XOR' TESTER CHANGE
2151                                :: THIS INSTRUCTION TO A 'NOP' (NOP=240)
2152 066372 013746 000004          MOV      ERRVEC, -(SP)      :: SAVE THE CONTENTS OF THE ERROR VECTOR
2153 066376 012737 066416 000004          MOV      #1$, ERRVEC      :: SET FOR TIMEOUT
2154 066404 005737 177060          TST      177060          :: TIME OUT ON XOR?
2155 066410 012637 000004          MOV      (SP)+, ERRVEC    :: RESTORE THE ERROR VECTOR
2156 066414 000424          BR      $$VLAD          :: GO TO THE NEXT TEST
2157 066416 062706 000004          1$: ADD      #4, SP      :: FIX STACK FROM TRAP
2158 066422 005037 177766          CLR      CPUERR          :: RESET CPU ERROR REGISTER
2159 066426 012637 000004          MOV      (SP)+, ERRVEC    :: RESTORE THE ERROR VECTOR
2160 066432 000407          BR      4$              :: LOOP ON THE PRESENT TEST
2161 066434          2$: ; #####END OF CODE FOR THE XOR TESTER#####
2162 066434 105737 002016          3$: TSTB     $ERFLG      :: HAS AN ERROR OCCURRED?
2163 066440 001412          BEQ      $$VLAD          :: BR IF NO
2164 066442 032777 001000 114316          BIT      #SW9, @SWR      :: LOOP ON ERROR?
2165 066450 001404          BEQ      5$              :: BR IF NO
2166 066452 013737 002750 002746          4$: MOV      $LPERR, $LPADR  :: SET LOOP ADDRESS TO LAST SCOPE
2167 066460 000410          BR      $OVER
2168 066462 105037 002016          5$: CLR      $ERFLG      :: ZERO THE ERROR FLAG
2169 066466 011637 002746          $$VLAD: MOV   (SP), $LPADR  :: SAVE SCOPE LOOP ADDRESS
2170 066472 011637 002750          MOV      (SP), $LPERR    :: SAVE ERROR LOOP ADDRESS
2171 066476 005037 002454          CLR      $ESCAPE        :: CLEAR THE ESCAPE FROM ERROR ADDRESS
2172 066502 004737 066514          $OVER: CALL  GETDIS
2173 066506 013716 002746          MOV      $LPADR, (SP)    :: FUDGE RETURN ADDRESS
2174 066512 000002          RTI                    :: FIXES PS

```

```

2176 066514          GETDIS: SUBTST <<SUBR DISPLAY>>
:*****
: *SUBTEST          SUBR    DISPLAY
:*****
2177 066514 113737 002106 002015      MOVB  BANK,$BANK
2178 066522 113737 002366 002014      MOVB  REALPAT,$PATMAR
2179 066530          PUSH   R0,4
                                MOV R0,-(SP)
                                MOV 4,-(SP)
2180 066536 010046 000004          SET4  #2$          ;TRAPS TO 4 GOTO 2$
                                MOV  #2$,4
                                .DSABL CRF
                                CLR   R0
                                MOV  MAINT,R0
                                ASH  #4,R0
                                BR   3$
2181 066544 005000          ADD  #4,SP          ;FIX STACK FROM TRAP
2182 066546 013700 177750          BIC  #^C340,R0
2183 066552 072027 000004          BISB R0,$PATMAR
2184 066556 000402          TST  RLFLAG        ;ARE WE RELOCATED?
2185 066560 062706 000004          BEQ  1$            ;NC - SKIP
2186 066564 042700 177437          BIS  #BIT15,$PATMAR ;YES - SET MSB
2187 066570 150037 002014          IF  MDISPLAY NE #0 THEN LET $PATMAR := MDISPLAY
2188 066574 005737 002134          TST  MDISPLAY
2189 066600 001403          BEQ  L765
2190 066602 052737 100000 002014      MOV  MDISPLAY,$PATMAR
2191 066610 005737 002676          L765:::
                                BEQ  L765
                                MOV  MDISPLAY,$PATMAR
2192 066624 013777 002014 114136      MOV  $PATMAR,@DISPLAY
2193 066632 013737 002014 000174      MOV  $PATMAR,DISPREG ;SOFTWARE DISPLAY REGISTER
2194 066640 012637 000004          POP  4,R0
                                MOV (SP)+,4
                                MOV (SP)+,R0
2195 066646 012600          RES4          ;RESTORE TRAPS TO 4 TO DEFAULT
2196 066646 012737 036726 000004      MOV  #TIMEOUT,4
                                CLR  CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                .DSABL CRF ;THAT A EXPECTED TRAP COULD HAVE SET
2196 066660 000207      RETURN
    
```

```

2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213 066662 005737 002530
      066662 001027
      066666 104410
2214 066670 105237 002016
2215 066672 001775
2216 066676 004737 066514
2217 066700 013737 002014 075140
2218 066704 032777 002000 114046
2219 066712 001404
2220 066720 104401 003003
2221 066722
      066722
2222 066726 104401 112132
      066726
2223 066732 005237 002754
2224 066736 100003
      066736 012737 077777 002754
2225 066740
2226 066746
      066746
2227 066746
      066746
2228 066746 011637 002024
2229 066752 162737 000002 002024
2230 066760 010637 002030
2231 066764 016637 000002 002034
2232 066772 117737 113026 002017
2233 067000 005737 002530
      067000 001044
2234 067006
      067006 005737 002026
      067012 001416
2235 067014 013737 002026 002024
2236 067022 162737 000002 002024
2237 067030 013737 002032 002030
2238 067036 013737 002036 002034
2239 067044 005037 002026
2240 067050
      067050
2241 067050 013737 002024 075136
    
```

```

.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 TYPEOUTS
*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
    TYPEIT $BELL
    .DSABL CRF
    TYPE MSG014 ;CONTROL Z
    TYPEIT ,MSG014
    .DSABL CRF
2$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
    IF RESULT IS MI BPL L767
    MOV #77777,$ERTTL
    END ;OF IF RESULT
    END ;OF IF NOERROR L767:;;;;;
    MOV (SP),ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
    SUB #2,ERRPC L766:;;;;;
    MOV SP,ERRSP
    MOV 2(SP),ERRPSW
    MOVB @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
    IF NOERROR IS FALSE
    IF BADPC NE #0 TST NOERROR
    MOV BADPC,ERRPC BNE L770
    SUB #2,ERRPC TST BADPC
    MOV BADSP,ERRSP BEQ L771
    MOV BADPSW,ERRPSW
    CLR BADPC
    END ;IF
    MOV ERRPC,$FATAL ;FOR APT
    
```

```
2242 067056          IF #SW13 SET.IN @SWR OR SLAERROR IS TRUE
    067056 032777 020000 113702
    067064 001003
    067066 005737 002570
    067072 001401
    067074
                                BIT #SW13,@SWR
                                BNE L772
                                TST SLAERROR
                                BEQ L773
                                L772:.....
2243 067074 000412          BR 3$
2244 067076          END ;OF IF #SW13
    067076
                                L773:.....
2245 067076          IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE
    067076 032777 000040 113662
    067104 001404
    067106 005737 002472
    067112 001401
                                BIT #SW5,@SWR
                                BEQ L774
                                TST TOOMANY
                                BEQ L774
2246 067114          GOTO 3$
    067114 000402
                                BR 3$
2247 067116          END ;OF IF #SW5
    067116
                                L774:.....
2248 067116          END ;OF IF NOERROR
    067116
                                L770:.....
2249 067116 004737 067324  CALL $ERRTYP          ;:GO TO USER ERROR ROUTINE
```

```

2251 067122          3$:   IF NOERROR IS FALSE
      067122 005737 002530
      067126 001064
2252 067130          TST @SWR          ::HALT ON ERROR
      067130 005777 113632          BPL 7$          ::SKIP IF CONTINUE
2253 067134 100002          $HALT:  HALT          ::HALT ON ERROR!
2254 067136 000000          CKSWR          ::TEST FOR CHANGE IN SOFT-SWR
2255 067140 104410          7$:   IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR
2256 067142          TST NOSCOPE
      067142 005737 002540          BNE L775
      067146 001006          BIT #SW9,@SWR
      067150 032777 001000 113610          BEQ L776
      067156 001402
2257 067160 013716 002750          MOV $LPERR,(SP)  ::FUDGE RETURN FOR LOOPING
2258 067164          END ;OF IF NOSCOPE
      067164          L776:::
2259 067164 005737 002454          TST $ESCAPE    ::CHECK FOR AN ESCAPE ADDRESS
2260 067170 001402          BEQ 9$          ::BR IF NONE
2261 067172 013716 002454          MOV $ESCAPE,(SP) ::FUDGE RETURN ADDRESS FOR ESCAPE
2262 067176          9$:   IF DETFLAG IS FALSE
      067176 005737 002326          TST DETFLAG
      067202 001035          BNE L777
2263 067204          IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL$ IS TRUE
      067204 005737 002442          TST ACTFLAG
      067210 001006          BNE L1000
      067212 005737 002444          TST APTFLAG
      067216 001003          BNE L1000
      067220 005737 002070          TST FATAL$
      067224 001405          BEQ L1001
      067226          L1000:::
2264 067226 012737 000001 075134          MOV #1,$MSGTY  ;FOR APT
2265 067234 000137 043522          JMP EXIT
2266 067240          END ;OF IF ACTFLAG
      067240          L1001:::
2267 067240          IF XXDPCHAIN IS TRUE AND $ERTTL HI #20
      067240 005737 002446          TST XXDPCHAIN
      067244 001414          BEQ L1002
      067246 023727 002754 000020          CMP $ERTTL,#20
      067254 101410          BLOS L1002
2268 067256          ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN
      067256 104401 115142          TYPE MSG066
      TYPEIT ,MSG066
      .DSABL CRF
      MOV 42,R0
      CLR 42
      JMP $ZAP42
2269 067262 013700 000042          END ;OF IF XXDPCHAIN
2270 067266 005037 000042          L1002:::
2271 067272 000137 014760          END ;OF IF DETFLAG
2272 067276          L777:::
2273 067276          ELSE
      067276 000403          BR L1003
      067300          L775:::
2275 067300          SET HEADER
      067300 012737 177777 002724          MOV #-1,HEADER
2276 067306          END ;OF IF NOERROR
      067306          L1003:::
2277 067306          10$:  CLEAR TOOMANY,CPUERR,NOERROR
      067306 005037 002472          CLR TOOMANY
  
```

067312 005037 177766
067316 005037 002530
2278 067322 000002
2279

RTI
.DSABL LSB

:::RETURN

CLR CPUERR
CLR NOERROR

.SBTTL ROUTINE ERROR MESSAGE TYPEOUT

```

2282
2283
2284
2285
2286
2287
2288
2289 067324 104415
2290 067326 104401 003010
2291 067332 005000
2292 067334 153700 002017
2293 067340 001004
2294
2295 067342 013746 002024
      067346 104402
2296 067350 000512
2297 067352 005300
2298 067354 006300
2299 067356 006300
2300 067360 006300
2301 067362 062700 101372
2302 067366 012037 067424
2303 067372 001417
2304 067374 005737 002530
2305 067400 001003
2306 067402 005737 002724
2307 067406 100011
2308 067410 005737 002070
2309 067414 001402
2310 067416 104401 115225
      067416
2311 067422 104401
      067422
2312 067424 000000
2313 067426 104401 003010
      067426
2314 067432 012037 067456
2315 067436 001412
2316 067440 005737 002530
2317 067444 001003
2318 067446 005737 002724
2319 067452 100004
2320 067454 104401
      067454
2321 067456 000000
2322 067460 104401 003010
      067460

```

```

*****
*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"
      TYPEIT    , $CRLF
      .DSABL    CRF
      CLR      R0          ;; PICKUP THE ITEM INDEX
      BISB     $ITEMB,R0
      BNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
                          ;; TYPE THE PC OF THE ERROR
      TYPOCT   ERRPC,<ERROR ADDRESS>
      MOV     ERRPC,-(SP) ;; SAVE ERRPC FOR TYPEOUT
                          ;; ERROR ADDRESS
                          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
      TYPOC
      .DSABL    CRF
      BR      11$        ;; GET OUT
      1$:     DEC      R0          ;; ADJUST THE INDEX SO THAT IT WILL
      ASL     R0          ;; WORK FOR THE ERROR TABLE
      ASL     R0
      ASL     R0
      ADD     #$ERRTB,R0   ;; FORM TABLE POINTER
      MOV     (R0)+,3$    ;; PICKUP "ERROR MESSAGE" POINTER
      BEQ     4$          ;; SKIP TYPEOUT IF NO POINTER
      TST    NOERROR     ;; IS THIS REALLY AN ERROR?
      BNE     12$        ;; YES - SKIP
      TST    HEADER      ;; TYPE HEADER?
      BPL     4$          ;; NO - SKIP
      12$:   TST    FATAL$  ;; WAS IT A FATAL ERROR?
      BEQ     2$          ;; NO - SKIP
      TYPE    MSG067     ;; FATAL
      TYPEIT  ,MSG067
      .DSABL    CRF
      2$:     TYPE      ;; TYPE THE "ERROR MESSAGE"
      TYPEIT
      .DSABL    CRF
      3$:     .WORD    0          ;; "ERROR MESSAGE" POINTER GOES HERE
      TYPE    $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPEIT  , $CRLF
      .DSABL    CRF
      4$:     MOV     (R0)+,5$   ;; PICKUP "DATA HEADER" POINTER
      BEQ     6$          ;; SKIP TYPEOUT IF 0
      TST    NOERROR     ;; IS THIS REALLY AN ERROR?
      BNE     13$        ;; YES - SKIP
      TST    HEADER      ;; TYPE HEADER?
      BPL     6$          ;; NO - SKIP
      13$:   TYPE      ;; TYPE THE "DATA HEADER"
      TYPEIT
      .DSABL    CRF
      5$:     .WORD    0          ;; "DATA HEADER" POINTER GOES HERE
      TYPE    $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPEIT  , $CRLF
      .DSABL    CRF

```

```
2323 067464 012001      6$:  MOV      (R0)+,R1      ::PICKUP 'DATA TABLE' POINTER
2324 067466 001436      BEQ      10$          ::BR IF NO DATA TO BE TYPED
2325 067470 012002      MOV      (R0)+,R2      ::PICKUP 'DATA FORMAT' POINTER
```

```

2328 067472 112203          7$:  MOVB  (R2)+,R3
2329 067474 006303          ASL   R3           ;MAKE IT A WORD ADDRESS
2330 067476 004773 067504    CALL  @8$(R3)
2331 067502 000421          BR    9$
2332 067504 067636          8$:  TAG70$
2333 067506 067646          TAG71$
2334 067510 067656          TAG72$
2335 067512 067722          TAG73$
2336 067514 067730          TAG74$
2337 067516 067742          TAG75$
2338 067520 067754          TAG76$
2339 067522 067776          TAG77$
2340 067524 070000          TAG78$
2341 067526 070006          TAG79$
2342 067530 070066          TAG80$
2343 067532 070100          TAG81$
2344 067534 070122          TAG82$
2345 067536 070324          TAG83$
2346 067540 070410          TAG84$
2347 067542 070502          TAG85$
2348 067544 070612          TAG86$
2349 067546 062701 000002    9$:  ADD   #2,R1       ;UPDATE DATA TABLE POINTER
2350 067552 005711          TST   (R1)        ;;IS THERE ANOTHER NUMBER?
2351 067554 001403          BEQ   10$         ;;BR IF NO
2352 067556 104401 112161    TYPE  MSG018      ;TYPE 2 SPACES
                TYPEIT ,MSG018
                .DSABL CRF
                BR    7$           ;;LOOP
2353 067562 000743
2354
2355 067564 005737 002116    10$: TST   MUT        ;IS THERE A MEMORY UNDER TEST
2356 067570 001402          BEQ   11$         ;NO - SKIP
2357 067572 005237 002724    INC   HEADER      ;YES - BUMP HEADER FLAG
2358 067576 104416          11$: RESREG
2359 067600          IF #SW7 SET.IN @SWR AND DETFLAG IS FALSE AND NOERROR IS FALSE
                BIT #SW7,@SWR
                BEQ L1004
                TST DETFLAG
                BNE L1004
                TST NOERROR
                BNE L1004
                CALL  DETAIL
                END ;OF IF #SW7
                L1004:::
2360 067624 004737 071364          TYPE  MSG104      ;CONTROL Z
2361 067630          TYPEIT ,MSG104
                .DSABL CRF
2362 067630 104401 116421          RETURN
2363 067634 000207

```

```
2366 :*****
2367 :*** OCTAL ***
2368 :*****
2369 067636 TAG70$: TYPOCT @ (R1) ;:TYPE AN OCTAL NUMBER
      067636 017146 000000 MOV @ (R1),-(SP) ;:SAVE @ (R1) FOR TYPEOUT
      067642 104402 TYPOC ;:GO TYPE--OCTAL ASCII (ALL DIGITS)
      .DSABL CRF
      RETURN

2370 067644 000207
2371
2372 :*****
2373 :*** DECIMAL ***
2374 :*****
2375 067646 TAG71$: TYPDEC @ (R1) ;:TYPE A DECIMAL NUMBER
      067646 017146 000000 MOV @ (R1),-(SP) ;:SAVE @ (R1) FOR TYPEOUT
      067652 104405 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
      RETURN

2376 067654 000207
2377
2378 :*****
2379 :*** INTERLEAVE ***
2380 :*****
2381 067656 TAG72$: PUSH R1,R5
      067656 010146 MOV R1,-(SP)
      067660 010546 MOV R5,-(SP)
2382 067662 013701 002106 MOV BANK,R1
2383 067666 070127 000006 MUL #6,R1
2384 067672 SET NOTAB ;INDICATE NO TABLE TO BE PRINTED - NOW
      067672 012737 177777 002464 MOV #-1,NOTAB
2385 067700 004737 036012 CALL TCFIG1
2386 067704 005037 002464 CLR NOTAB
2387 067710 POP R5,R1
      067710 012605 MOV (SP)+,R5
      067712 012601 MOV (SP)+,R1
2388 067714 TYPE MSG014 ;1 SPACE
      067714 104401 112132 TYPEIT ,MSG014
      .DSABL CRF
      RETURN

2389 067720 000207
2390
2391 :*****
2392 :*** BOX ***
2393 :*****
2394 067722 004737 070672 TAG73$: CALL WHOBOX
2395 067726 000207 RETURN

2396
2397 :*****
2398 :*** PATTERN ***
2399 :*****
2400 067730 TAG74$: TYPOCS REALPAT,<TYPE (0-77)>,2,Z
      067730 013746 002366 MOV REALPAT,-(SP) ;:SAVE REALPAT FOR TYPEOUT
      ;:TYPE (0-77)
      TYPOS ;:GO TYPE--OCTAL ASCII
      .BYTE 2 ;:TYPE 2 DIGIT(S)
      .BYTE 1 ;:TYPE LEADING ZEROS
      .DSABL CRF
      RETURN

2401 067740 000207
2402
2403 :*****
```

2404
2405
2406 067742 013746 002106
067746 104403
067750 003
067751 000
2407 067752 000207
2408
2409
2410
2411
2412 067754 005737 002126
2413 067760 001003
2414 067762 104401 115234
067762 104401 115234
2415 067766 000207
2416 067770 104401 115241
067770 104401 115241
2417 067774 000207

```
**** BANK ****  
*****  
TAG75$: TYPOCS BANK,<TYPE (0-167)>,3  
MOV BANK,-(SP) ::SAVE BANK FOR TYPEOUT  
::TYPE (0-167)  
TYPOS ::GO TYPE--OCTAL ASCII  
.BYTE 3 ::TYPE 3 DIGIT(S)  
.BYTE 0 ::SUPPRESS LEADING ZEROS  
.DSABL CRF  
RETURN  
  
*****  
**** MTYPE ****  
*****  
TAG76$: TST MKFLAG  
BNE 1$  
TYPE MSG068 ;MJ11  
TYPEIT ,MSG068  
.DSABL CRF  
RETURN  
1$: TYPE MSG069 ;MK11  
TYPEIT ,MSG069  
.DSABL CRF  
RETURN
```

2420
2421
2422
2423 067776 000207
2424
2425
2426
2427
2428 070000
070000 104401 115060

```

:*****
:*** SPARE FUNCTION ***
:*****
TAG77$: RETURN

:*****
:*** UNKNOWN DATA ***
:*****
TAG78$: TYPE      MSG061
      TYPEIT     ,MSG061
      .DSABL     CRF
      RETURN

```

2429 070004 000207

2430
2431
2432
2433
2434 070006 013737 002040 002044
2435 070014 162737 060000 002044
2436 070022 013737 002106 002046
2437 070030 006237 002046
2438 070034 103003
2439 070036 052737 100000 002044
2440 070044 012746 002044
2441 070050 004737 075014
2442 070054 062706 000002
2443 070060
070060 104401 075122

```

:*****
:*** PHYSICAL ADDRESS ***
:*****
TAG79$: MOV      ADDRESS,PHYADD
      SUB      #FIRST,PHYADD
      MOV      BANK,PHYADD+2
      ASR      PHYADD+2
      BCC      1$
      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
      TYPEIT   , $OCT8
      .DSABL   CRF
      RETURN

```

2444 070064 000207

2445
2446
2447
2448 070066
070066 013746 002112

070072 104403
070074 001
070075 000

```

:*****
:*** MARGIN ***
:*****
TAG80$: TYPOCS  MARGIN,<TYPE (0,2,3,4, OR 5)>,1
      MOV      MARGIN,-(SP)           ;; SAVE MARGIN FOR TYPEOUT
                                           ;; TYPE (0,2,3,4, OR 5)
                                           ;; GO TYPE--OCTAL ASCII
      TYPOS
      .BYTE    1                       ;; TYPE 1 DIGIT(S)
      .BYTE    0                       ;; SUPPRESS LEADING ZEROS
      .DSABL   CRF
      RETURN

```

2449 070076 000207

2450
2451
2452
2453
2454 070100
070100 104401 112161

2455 070104
070104 017146 000000

070110 104403
070112 003
070113 001

```

:*****
:*** OCTAL BYTE ***
:*****
TAG81$: TYPE     MSG018                ; 2 SPACES
      TYPEIT     ,MSG018
      .DSABL     CRF
      TYPOCS     @(R1),<TYPE BYTE>,3,2
      MOV        @(R1),-(SP)           ;; SAVE @(R1) FOR TYPEOUT
                                           ;; TYPE BYTE
                                           ;; GO TYPE--OCTAL ASCII
      TYPOS
      .BYTE     3                       ;; TYPE 3 DIGIT(S)
      .BYTE     1                       ;; TYPE LEADING ZEROS
      .DSABL     CRF
      TYPE       MSG014                ; SPACE
      TYPEIT     ,MSG014
      .DSABL     CRF

```

2456 070114
070114 104401 112132

CEMKABO 1170 MAIN MEMORY DIAG 2 MACRO M1113 03-AUG-79 16:50 PAGE 469-1
ROUTINE ERROR MESSAGE TYPEOUT

E 11

SEQ 0962

2457 070120 000207

RETURN

2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493

070122 012737 037477 115336 TAG82\$:
070130 005737 002126
070134 001470
070136 013746 002144
070142 013746 002146
070146 010046
070150 010146
070152 010246
070154 013746 002256
070160 013737 002000 002256
070166 012702 177777
070172 104426
070174 032737 100020 002144
070202 001414
070204 013702 002144
070210 072227 177774
070214 042702 177761
070220 032737 001000 002146
070226 001402
070230 052702 000001
070234
070234
070234 005702
070236 002416
070240 020227 000011
070244 003406
070246 162702 000012
070252 000302
070254 062702 030061
070260 000403
070262
070262 000302
070264 062702 030040
070270
070270
070270 010237 115336

```

*****
*** ARRAY ***
*****
SOMETIMES THE ARRAY INFORMATION MAY BE MISLEADING
IF YOU ARE LOOKING FOR SBE'S (SWITCH 0 IS UP)
IF AN ERROR IS BEING REPORTED AND THE CODE THAT TYPES THE
ERROR HAS A SINGLE BIT ERROR
THE OLD SBE INFORMATION WILL BE WRITTEN OVER AND THE ARRAY
TYPED MAY BE THE ARRAY THAT HAS THE ERROR REPORT CODE IN IT
AND NOT THE ARRAY THAT WAS UNDER TEST AND THAT THE REST
OF THE ERROR REPORT IS TALKING ABOUT.
HOWEVER THE ARRAY IS STILL BAD (IF YOU'RE LOOKING FOR SBE'S).
MOV #'??,MSG074
IF MKFLAG IS TRUE

TST MKFLAG
BEQ L1005

PUSH CSR,CSR+2,R0,R1,R2,CSRNO

MOV CSR,-(SP)
MOV CSR+2,-(SP)
MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)
MOV CSRNO,-(SP)

MOV BADCSR,CSRNO
SET R2

MOV #-1,R2

READCSR
IF #BIT15!BIT4 SET.IN CSR

BIT #BIT15!BIT4,CSR
BEQ L1006

MOV CSR,R2
ASH #-4,R2
BIC #^C16,R2
IF #BIT9 SET.IN CSR+2 THEN LET R2 := R2 SET.BY #BIT0

BIT #BIT9,CSR+2
BEQ L1007
BIS #BIT0,R2
L1007:::

IF R2 GE #0

TST R2
BLT L1010

IF R2 GT #9.

CMP R2,#9.
BLE L1011

SUB #10.,R2
SWAB R2
ADD #'10,R2
ELSE

BR L1012
L1011:::

SWAB R2
ADD #'0,R2
END ;OF IF #BIT3

MOV R2,MSG074
L1012:::

```

2494 070274
070274
2495 070274 012637 002256
070300 012602
070302 012601
070304 012600
070306 012637 002146
070312 012637 002144
2496 070316
070316
2497 070316 104401 115336
070316
2498 070322 000207

END ;OF IF R2
POP CSRNO,R2,R1,R0,CSR+2,CSR

END ;OF IF MKFLAG

TYPE MSG074
TYPEIT ,MSG074
.DSABL CRF
RETURN

L1010:::~::~

MOV (SP)+,CSRNO
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
MOV (SP)+,CSR+2
MOV (SP)+,CSR

L1005:::~::~

```

2501 .....
2502 :*** BINARY BYTE ***
2503 .....
2504 TAG83$: PUSH R0,R2
      070324 010046
      070324 010246
      070326 010246
      070330 104401 112161
      070330 104401 112161
      070334 017100 000000
      070340 012702 000001
      070340 012702 000001
      070344 106100
      070344 106100
      070346 103004
      070346 103004
      070350 112737 000061 112134
      070356 000403
      070356 000403
      070360 112737 000060 112134
      070366 112737 000060 112134
      070366 112737 000060 112134
      070366 104401 112134
      070366 104401 112134
      070372 005202
      070372 005202
      070374 020227 000010
      070400 003761
      070402 003761
      070402 012602
      070402 012602
      070404 012600
      070406 000207
      070406 000207
      070410 010046
      070410 010046
      070412 010246
      070412 010246
      070414 104401 112161
      070414 104401 112161
      070420 017100 000000
      070424 012702 000004
      070430 072002
      070432 012702 000001
      070432 012702 000001
      070436 106100
      070436 106100
      070440 103004
      070440 103004
      070442 112737 000061 112134
      070450 000403
      070450 000403
  
```

```

      MOV R0,-(SP)
      MOV R2,-(SP)
      TYPE MSG018 ;2 SPACES
      TYPEIT ,MSG018
      .DSABL CRF
      MOV @(R1),R0
      FOR R2 := #1 TO #8.
      MOV #1,R2
      B200:.....
      ROLB R0
      ON.ERROR
      BCC L1013
      MOVB #'1,MSG015
      ELSE
      BR L1014
      L1013:.....
      MOVB #'0,MSG015
      END ;OF ON.ERROR
      L1014:.....
      TYPE MSG015
      TYPEIT ,MSG015
      .DSABL CRF
      END ;OF FOR R2
      INC R2
      CMP R2,#8.
      BLE B200
      E200:.....
      POP R2,R0
      MOV (SP)+,R2
      MOV (SP)+,R0
      RETURN
      .....
      :*** 4 BINARY BITS ***
      .....
      TAG84$: PUSH R0,R2
      MOV R0,-(SP)
      MOV R2,-(SP)
      TYPE MSG018 ;2 SPACES
      TYPEIT ,MSG018
      .DSABL CRF
      MOV @(R1),R0
      MOV #4,R2
      ASH R2,R0 ;SHIFT 4 PLACES LEFT
      FOR R2 := #1 TO #4
      MOV #1,R2
      B201:.....
      ROLB R0
      ON.ERROR
      BCC L1015
      MOVB #'1,MSG015
      ELSE
      BR L1016
  
```

2532	070452	112737	000060	112134	MOVB	#*0,MSG015	L1015:::~::~
2533	070460				END ;OF	ON.ERROR	
2534	070460	104401	112134		TYPE	MSG015	L1016:::~::~
	070460				TYPEIT	MSG015	
	070460				.DSABL	CRF	
2535	070464				END ;OF	FOR R2	
	070464	005202					INC R2
	070466	020227	000004				CMP R2,#4
	070472	003761					BLE B201
	070474						E201:::~::~
2536	070474				POP	R2,R0	
	070474	012602					MOV (SP)+,R2
	070476	012600					MOV (SP)+,R0
2537	070500	000207			RETURN		

```

2540 .....
2541 *** ARRAY FROM SBE FROM OLD CSR ***
2542 .....
2543 070502 TAG85$: PUSH R2
2544 070502 010246 MOV R2,-(SP)
2544 070504 012737 037477 115336 MOV #'??,MSG074
2545 070512 013702 002150 MOV SBECR,R2
2546 070516 072227 177774 ASH #-4,R2
2547 070522 042702 177761 BIC #^C16,R2
2548 070526 IF #BIT9 SET.IN SBECR+2 THEN LET R2 := R2 SET.BY #BIT0
2548 070526 032737 001000 002152 BIT #BIT9,SBECR+2
2548 070534 001402 BEQ L1017
2548 070536 052702 000001 BIS #BIT0,R2
2548 070542 L1017:.....
2549 070542 IF R2 GE #0 TST R2
2549 070542 005702 BLT L1020
2549 070544 002416
2550 070546 IF R2 GT #9. CMP R2,#9.
2550 070546 020227 000011 BLE L1021
2550 070552 003406
2551 070554 162702 000012 SUB #10.,R2
2552 070560 000302 SWAB R2
2553 070562 062702 030061 ADD #'10,R2
2554 070566 ELSE
2554 070566 000403 BR L1022
2554 070570 L1021:.....
2555 070570 000302 SWAB R2
2556 070572 062702 030040 ADD #'0,R2
2557 070576 END ;OF IF #BIT3
2557 070576 L1022:.....
2558 070576 010237 115336 MOV R2,MSG074
2559 070602 END ;OF IF R2
2559 070602 L1020:.....
2560 070602 TYPE MSG074
2560 070602 104401 115336 TYPEIT ,MSG074
2560 070602 .DSABL CRF
2561 070606 POP R2
2561 070606 012602 MOV (SP)+,R2
2562 070610 000207 RETURN
2563 .....
2564 *** MASTER OR SLAVE1 OR SLAVE2 OR SLAVE3 FROM BAD ***
2565 .....
2566 .....
2567 070612 TAG86$: IF BAD EQ #0
2567 070612 005737 002056 TST BAD
2567 070616 001002 BNE L1023
2568 070620 TYPE MSG113 ;MASTER
2568 070620 104401 117103 TYPEIT ,MSG113
2568 070620 .DSABL CRF
2569 070624 END IF
2569 070624 L1023:.....
2570 070624 IF BAD EQ #1
2570 070624 023727 002056 000001 CMP BAD,#1
2570 070632 001002 BNE L1024
2571 070634 TYPE MSG114 ;SLAVE1
2571 070634 104401 117113 TYPEIT ,MSG114
2571 070634 .DSABL CRF

```

```
2572 070640          END IF
      070640
2573 070640          IF BAD EQ #2
      070640 023727 002056 000002
      070646 001002
      070650          TYPE MSG115          ;SLAVE2
      070650 104401 117123          TYPEIT ,MSG115
      .DSABL CRF
      END IF
2575 070654
      070654
2576 070654          IF BAD EQ #3
      070654 023727 002056 000003
      070662 001002
      070664          TYPE MSG116          ;SLAVE3
      070664 104401 117133          TYPEIT ,MSG116
      .DSABL CRF
      END IF
2578 070670
      070670
2579 070670 000207          RETURN
```

L1024:::~::~

CMP BAD,#2
BNE L1025

L1025:::~::~

CMP BAD,#3
BNE L1026

L1026:::~::~

```

2581 070672          WHOBOX: SUBTST <<SUBR WHO'S BOX IS THIS?>>
:*****
:*SUBTEST          SUBR   WHO'S BOX IS THIS?
:*****
2582 070672 005737 002126          TST      MKFLAG          :MK11?
2583 070676 001003          BNE      1$              :YES - SKIP
2584          :NO - MJ11 - DON'T KNOW ABOUT BOX NUMBERS
2585 070700          TYPE      MSG057          : ?
      070700 104401 114716          TYPEIT   ,MSG057
      .DSABL   CRF
2586 070704 000207          RETURN
2587
2588 070706          1$:  PUSH      CSRNO,R1,R2,R3,R4
      070706 013746 002256          MOV      CSRNO,-(SP)
      070712 010146          MOV      R1,-(SP)
      070714 010246          MOV      R2,-(SP)
      070716 010346          MOV      R3,-(SP)
      070720 010446          MOV      R4,-(SP)
2589 070722 005037 002424          CLR      SUCCESS
2590 070726 012703 000200          MOV      #200,R3
2591 070732 012701 000060          MOV      #'0,R1
2592 070736          BEGIN WHODUNIT
2593 070736          FOR R4 := #0 TO #16 BY #2
      070736 005004
2594 070740          IFB R3 SET.JN MKCSRS
      070740 130337 002334
      070744 001452
2595 070746 016437 002214 002260          MOV      CSRFB(R4),FIRSTB
2596 070754 016437 002234 002262          MOV      CSRLB(R4),LASTB
2597 070762 010437 002000          MOV      R4,BADCSR
2598 070766 006337 002000          ASL      BADCSR
2599 070772          IF FIRSTB LE BANK AND LASTB GE BANK
      070772 023737 002260 002106          CMP      FIRSTB,BANK
      071000 003034          BGT      L1030
      071002 023737 002262 002106          CMP      LASTB,BANK
      071010 002430          BLT      L1030
2600 071012 016402 002174          MOV      CSR2(R4),R2
2601 071016 072227 177764          ASH      #-12.,R2
2602 071022 042702 177770          BIC      #^C7,R2
2603 071026          CASE R2
      071026 010246          MOV      R2,-(SP)
      071030 006316          ASL      @SP
      071032 004737 071056          JSR      PC,L1031
2604 071036 071152          ADD..0
2605 071040 071162          ADD..1
2606 071042 071202          ADD..2
2607 071044 071232          ADD..3
2608 071046 071262          ADD..4
2609 071050 071264          ADD..5
2610 071052 071304          ADD..6
2611 071054 071334          ADD..7
2612 071056          END ;OF CASE R2
      071056 062616
      071060 013646
L1031:
      ADD (SP)+,@SP
      MOV @ (SP)+,-(SP)

```

```

2613 071062 004736
      071064 005737 002424
      071070 001007
2614 071072
      071072
2615 071072
      071072
2616 071072 006003
2617 071074 005201
2618 071076
      071076 062704 000002
      071102 020427 000016
      071106 003714
      071110
2619 071110
      071110
2620 071110
      071110 005737 002424
      071114 001402
2621 071116 110137 114717
2622 071122
      071122
2623 071122
      071122 104401 114716
2624 071126 112737 000077 114717
2625 071134
      071134 012604
      071136 012603
      071140 012602
      071142 012601
      071144 012637 002256
2626 071150 000207

```

```

      IF SUCCESS IS TRUE THEN LEAVE WHODUNIT
      END ;OF IF FIRSTB
      END ;OF IFB R3
      ROR R3
      INC R1
      END ;OF FOR CSRNO
      END WHODUNIT
      IF SUCCESS IS TRUE
      MOV B R1,MSG057+1
      END ;OF IF SUCCESS
      TYPE MSG057
      TYPEIT ,MSG057
      .DSABL CRF
      MOV B #'?,MSG057+1
      POP R4,R3,R2,R1,CSRNO
      RETURN

```

```

      JSR PC,@(SP)+
      TST SUCCESS
      BNE E202
      L1030:::
      L1027:::
      ADD #2,R4
      CMP R4,#16
      BLE B203
      E203:::
      E202:::
      TST SUCCESS
      BEQ L1032
      L1032:::
      MOV (SP)+,R4
      MOV (SP)+,R3
      MOV (SP)+,R2
      MOV (SP)+,R1
      MOV (SP)+,CSRNO

```

```

2629 071152          ADD..0: SET      SUCCESS
      071152 012737 177777 002424          MOV #-1,SUCCESS
2630 071160          RETURN
2631 071162          ADD..1: IF #BIT2 OFF.IN ADDRESS THEN LET SUCCESS := ONES
      071162 032737 000004 002040          BIT #BIT2,ADDRESS
      071170 001003          BNE L1033
      071172 013737 002726 002424          MOV ONES,SUCCESS
      071200          L1033:::
2632 071200          RETURN
2633 071202          ADD..2: IF #BIT3 OFF.IN ADDRESS AND #BIT2 OFF.IN ADDRESS THEN LET SUCCESS := ONES
      071202 032737 000010 002040          BIT #BIT3,ADDRESS
      071210 001007          BNE L1034
      071212 032737 000004 002040          BIT #BIT2,ADDRESS
      071220 001003          BNE L1034
      071222 013737 002726 002424          MOV ONES,SUCCESS
      071230          L1034:::
2634 071230          RETURN
2635 071232          ADD..3: IF #BIT3 SET.IN ADDRESS AND #BIT2 OFF.IN ADDRESS THEN LET SUCCESS := ONES
      071232 032737 000010 002040          BIT #BIT3,ADDRESS
      071240 001407          BEQ L1035
      071242 032737 000004 002040          BIT #BIT2,ADDRESS
      071250 001003          BNE L1035
      071252 013737 002726 002424          MOV ONES,SUCCESS
      071260          L1035:::
2636 071260          RETURN
2637 071262          ADD..4: RETURN
2638 071264          ADD..5: IF #BIT2 SET.IN ADDRESS THEN LET SUCCESS := ONES
      071264 032737 000004 002040          BIT #BIT2,ADDRESS
      071272 001403          BEQ L1036
      071274 013737 002726 002424          MOV ONES,SUCCESS
      071302          L1036:::
2639 071302          RETURN
2640 071304          ADD..6: IF #BIT3 OFF.IN ADDRESS AND #BIT2 SET.IN ADDRESS THEN LET SUCCESS := ONES
      071304 032737 000010 002040          BIT #BIT3,ADDRESS
      071312 001007          BNE L1037
      071314 032737 000004 002040          BIT #BIT2,ADDRESS
      071322 001403          BEQ L1037
      071324 013737 002726 002424          MOV ONES,SUCCESS
      071332          L1037:::
2641 071332          RETURN
2642 071334          ADD..7: IF #BIT3 SET.IN ADDRESS AND #BIT2 SET.IN ADDRESS THEN LET SUCCESS := ONES
      071334 032737 000010 002040          BIT #BIT3,ADDRESS
      071342 001407          BEQ L1040
      071344 032737 000004 002040          BIT #BIT2,ADDRESS
      071352 001403          BEQ L1040
      071354 013737 002726 002424          MOV ONES,SUCCESS
      071362          L1040:::
2643 071362          RETURN
2644
2645 071364          DETAIL: SUBTST <<SUBR DETAILED ERROR REPORT>>
      :*****
      :*SUBTEST      SUBR      DETAILED ERROR REPORT
      :*****
2646 071364 005237 002326          INC      DETFLAG
2647 071370 022737 000003 002326          CMP      #3,DETFLAG
2648 071376 101465          BLOS    4$
2649 071400 022737 000002 002326          CMP      #2,DETFLAG
    
```

```

SUBR DETAILED ERROR REPORT
2650 071406 001435 BEQ 2$
2651 071410 PUSH HEADER,MUT MOV HEADER,-(SP)
      071410 013746 002724 MOV MUT,-(SP)
      071414 013746 002116
2652 071420 SET HEADER MOV #-1,HEADER
      071420 012737 177777 002724
2653 071426 005037 002116 CLR MUT
2654 071432 010037 002306 MOV R0,DETRO
2655 071436 012700 002310 MOV #DETR1,R0
2656 071442 010120 MOV R1,(R0)+
2657 071444 010220 MOV R2,(R0)+
2658 071446 010320 MOV R3,(R0)+
2659 071450 010420 MOV R4,(R0)+
2660 071452 010520 MOV R5,(R0)+
2661 071454 013720 002030 MOV ERRSP,(R0)+
2662 071460 013720 002034 MOV ERRPSW,(R0)+
2663 071464 013700 002306 MOV DETRO,R0
2664 071470 SET NOERROR
      071470 012737 177777 002530 MOV #-1,NOERROR
2665 071476 104013 ERROR +13
2666 071500 000415 BR 1$
2667 071502 2$: PUSH HEADER,MUT MOV HEADER,-(SP)
      071502 013746 002724 MOV MUT,-(SP)
      071506 013746 002116
2668 071512 SET HEADER MOV #-1,HEADER
      071512 012737 177777 002724
2669 071520 005037 002116 CLR MUT
2670 071524 SET NOERROR MOV #-1,NOERROR
      071524 012737 177777 002530
2671 071532 104031 ERROR +31
2672 071534 1$: POP MUT,HEADER MOV (SP)+,MUT
      071534 012637 002116 MOV (SP)+,HEADER
      071540 012637 002724
2673 ;WARNING RECURSIVE
2674 071544 004737 071364 CALL DETAIL
2675 071550 000207 RETURN

```

```

2678                                     ;SIMULATE CONTROL 'T'
2679 071552 004737 073404      4$: CALL CONTT                               ;DISPLAY 'DISPLAY' INFO
2680
2681                                     ;TYPE CONTENTS OF ALL CSR'S
2682 071556                                     PUSH CSR,CSR+2,CSRNO,R1
                                MOV CSR,-(SP)
                                MOV CSR+2,-(SP)
                                MOV CSRNO,-(SP)
                                MOV R1,-(SP)
                                071556 013746 002144
                                071562 013746 002146
                                071566 013746 002256
                                071572 010146
2683 071574                                     TYPE MSG058
                                071574 104401 114722      TYPEIT ,MSG058
                                .DSABL CRF
                                TYPE $CRLF
                                TYPEIT , $CRLF
                                .DSABL CRF
2684 071600                                     MOV MKCSRS,R1
                                071600 104401 003010      BEGIN DUMPCSRLOOP
                                B204: :::::::
                                FOR CSRNO := #0 TO #34 BY #4.
                                CLR CSRNO
                                B205: :::::::
                                ASLB R1
                                ON.ERROR
                                BCC L1041
                                READCSR
                                TYPOCT CSR
                                MOV CSR,-(SP)      ;;SAVE CSR FOR TYPEOUT
                                TYPOC              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                .DSABL CRF
                                TYPE MSG018      ;2 SPACES
                                TYPEIT ,MSG018
                                .DSABL CRF
                                TYPOCT CSR+2
                                MOV CSR+2,-(SP)   ;;SAVE CSR+2 FOR TYPEOUT
                                TYPOC              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                .DSABL CRF
                                TYPE MSG018      ;2 SPACES
                                TYPEIT ,MSG018
                                .DSABL CRF
                                END ;OF ON.ERROR
                                L1041: :::::::
                                IFB R1 EQ #0 THEN LEAVE DUMPCSRLOOP
                                TSTB R1
                                BEQ E204
                                END ;OF FOR CSRNO
                                ADD #4,CSRNO
                                CMP CSRNO,#34
                                BLE B205
                                E205: :::::::
                                END DUMPCSRLOOP
                                E204: :::::::
                                POP R1,CSRNO,CSR+2,CSR
                                MOV (SP)+,R1
                                MOV (SP)+,CSRNO
                                MOV (SP)+,CSR+2
                                MOV (SP)+,CSR
2685 071604 013701 002334
2686 071610
2687 071610
                                071610 005037 002256
                                071614
2688 071614 106301
2689 071616
                                071616 103013
2690 071620 104426
2691 071622
                                071622 013746 002144
                                071626 104402
2692 071630
                                071630 104401 112161
2693 071634
                                071634 013746 002146
                                071640 104402
2694 071642
                                071642 104401 112161
2695 071646
2696 071646
                                071646 105701
                                071650 001407
2697 071652
                                071652 062737 000004 002256
                                071660 023727 002256 000034
                                071666 003752
                                071670
2698 071670
                                071670
2699 071670
                                071670 012601
                                071672 012637 002256
                                071676 012637 002146
                                071702 012637 002144

```

```

2702                                     ;TYPE STACKS
2703 071706 PUSH RO,R1
      071706 010046
      071710 010146
                                         MOV RO,-(SP)
                                         MOV R1,-(SP)

2704
2705                                     ;TYPE KERNEL STACK
2706 071712 TYPE MSG088 ;KERNEL STACK
      071712 104401 115760
      .DSABL CRF
      MOV KSTACK,R1
      SUB #2,R1
      FOR RO := SP TO R1 BY #2
                                         MOV SP,R0
                                         B206:::

2710 071730 TYPE $CRLF
      071730 104401 003010
      TYPEIT $CRLF
      .DSABL CRF
      TYPOCT RO
      MOV RO,-(SP) ;:SAVE RO FOR TYPEOUT
      TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
      TYPE MSG018 ;2 SPACES
      TYPEIT ,MSG018
      .DSABL CRF
      TYPOCT (RO)
      MOV (RO),-(SP) ;:SAVE (RO) FOR TYPEOUT
      TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
      END ;OF FOR RO
                                         ADD #2,R0
                                         CMP RO,R1
                                         BLE B206
                                         E206:::

2714 071750
      071750 062700 000002
      071754 020001
      071756 003764
      071760

2715
2716                                     ;TYPE SUPERVISOR STACK
2717 071760 BIC #BIT13!BIT12,PSW ;SET PREVIOUS MODE TO SUPERVISOR
      071766 042737 030000 177776
      071774 052737 010000 177776
      071776 006506
      071776 012600
      072000
      072000 104401 115776
      TYPE MSG089 ;SUPERVISOR STACK
      TYPEIT ,MSG089
      .DSABL CRF
      IF RO LT #SUPSTK
                                         MOV (SP)+,RO
                                         CMP RO,#SUPSTK
                                         BGE L1042
      FOR RO := RO TO #SUPSTK-2 BY #2
                                         B207:::

2723 072012 TYPE $CRLF
      072012 104401 003010
      TYPEIT $CRLF
      .DSABL CRF
      TYPOCT RO
      MOV RO,-(SP) ;:SAVE RO FOR TYPEOUT
      TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
      TYPE MSG018 ;2 SPACES
      TYPEIT ,MSG018

2725 072016
      072016 010046
      072020 104402

2726 072022
      072022 104401 112161
    
```

```

2727 072026      .DSABL CRF
      072026 011046      TYPOCT      (R0)
      072030 104402      MOV      (R0),-(SP)      ;;SAVE (R0) FOR TYPEOUT
                                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                .DSABL CRF
                                END ;OF FOR R0
2728 072032      ADD #2,R0
      072032 062700 000002      CMP R0,#SUPSTK-2
      072036 020027 000736      BLE B207
      072042 003763      E207:;:;:;:;:;:
      072044      BR L1043
2729 072044      ELSE
      072044 000402      L1042:;:;:;:;:;:
      072046      TYPE MSG091      ;IS EMPTY
      072046 104401 116034      TYPEIT ,MSG091
                                .DSABL CRF
                                END ;OF IF R0
2730 072046      L1043:;:;:;:;:;:
      072052      ;TYPE USER STACK
      072052      BIS #BIT13!BIT12,PSW      ;SET PREVIOUS MODE TO USER
2731 072052      MFPI USP
      072052      POP R0
      072052      MOV (SP)+,R0
2732 072052      TYPE MSG090      ;USER STACK
      072052 052737 030000 177776      TYPEIT ,MSG090
      072060 006506      .DSABL CRF
      072062 012600      IF R0 LT #USESTK
      072062 104401 116020      FOR R0 := R0 TO #USESTK-2 BY #2
      072064 104401 116020      TYPE $CRLF
      072064      TYPEIT ,CRLF
      072064      .DSABL CRF
      072070 020027 000700      MOV R0,R0      ;SAVE R0 FOR TYPEOUT
      072074 002016      TYPOCT R0      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      072076      .DSABL CRF
      072076 104401 003010      TYPE MSG018      ;2 SPACES
      072076      TYPEIT ,MSG018
      072102 010046      .DSABL CRF
      072102 104402      TYPOCT      (R0)
      072104      MOV (R0),-(SP)      ;;SAVE (R0) FOR TYPEOUT
      072104      TYPOCT      (R0)      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      072106 104401 112161      .DSABL CRF
      072106      END ;OF FOR R0
      072112 011046      ADD #2,R0
      072112 104402      CMP R0,#USESTK-2
      072114      BLE B210
2741 072112      E210:;:;:;:;:;:
      072112 010046      ELSE
      072114 104402      L1044:;:;:;:;:;:
      072116 062700 000002      TYPE MSG091      ;IS EMPTY
      072122 020027 000676      TYPEIT ,MSG091
      072126 003763      .DSABL CRF
      072130      END ;OF FOR R0
      072130      ADD #2,R0
      072130 000402      CMP R0,#USESTK-2
      072132      BLE B210
2745 072130      E210:;:;:;:;:;:
      072130 000402      BR L1045
      072132      L1044:;:;:;:;:;:
2746 072132      TYPE MSG091      ;IS EMPTY
      072132 104401 116034      TYPEIT ,MSG091
                                .DSABL CRF
    
```

2747 072136
 072136
 2748 072136 104401 003010
 072136
 2749 072142 005037 002326
 2750 072146
 072146 012601
 072150 012600
 2751 072152 000207

END ;OF IF R0
 TYPE \$CRLF
 TYPEIT .CRLF
 .DSABL CRF
 CLR DETFLAG
 POP R1,R0
 RETURN

L1045:~::~~::~

MOV (SP)+,R1
 MOV (SP)+,R0

```

2754 072154      QUE:  SUBTST <<QUE MESSAGES FROM SLAVE CPU'S>>
*****
*SUBTEST      QUE MESSAGES FROM SLAVE CPU'S
*****
2755 072154 104410      CKSWR
2756 072156 010046      PUSH  R0,R1,CONTRL
                                MOV R0,-(SP)
                                MOV R1,-(SP)
                                MOV CONTRL,-(SP)
                                072160 010146
                                072162 013746 177746
2757 072166 104424      CACHOFF      :TURN CACHE OFF
2758 072170 012700 000002  FOR R0 := #2 TO #6 BY #2
                                MOV #2,R0
                                B211:.....
2759 072174 005760 002626      IF PTYFLAG(R0) IS TRUE
                                TST PTYFLAG(R0)
                                BEQ L1046
2760 072202 010001      MOV  R0,R1
2761 072204 006201      ASR  R1
2762 072206 062701 000060      ADD  #0,R1
2763 072212 110137 116242      MOVB R1,MSGA98
2764 072216 104401 116234      TYPE  MSG098      ;SLAVEN>      ;WHERE N = 1,2, OR 3
                                TYPEIT ,MSG098
                                .DSABL CRF
2765 072222 126027 002636 000032  QUE1:  IFB PTYBUF(R0) NE #32      ;CONTROL Z?
                                CMPB PTYBUF(R0),#32
                                BEQ L1047
2766 072232 016046 002636      PUSH  PTYBUF(R0)
                                MOV PTYBUF(R0),-(SP)
2767 072236 005060 002626      CLR  PTYFLAG(R0)
2768 072242 004737 057076      CALL $TYPEC
2769 072246 062706 000002      ADD  #2,SP      ;FIX STACK
2770 072252 005760 002626      TST  PTYFLAG(R0)
2771 072256 100375      BPL  1$
2772 072260 000760      GOTO QUE1
                                BR QUE1
2773 072262 000415      ELSE
                                BR L1050
2774 072264 104401 116245      TYPE  MSG099
                                L1047:.....
                                ;MASTER>
                                TYPEIT ,MSG099
                                .DSABL CRF
2775 072270 005060 002626      CLR  PTYFLAG(R0)
2776 072274 005737 002134      IF RLFLAG IS TRUE
                                TST RLFLAG
                                BEQ L1051
2777 072302 012737 177777 002570      SET  SLAVERROR
                                MOV #-1,SLAVERROR
2778 072310 104000      ERROR
                                ;LOG ERROR
2779 072312 005037 002570      CLR  SLAVERROR
2780 072316 072316      END ;OF IF RLFLAG IS TRUE
                                L1051:.....
2781 072316 072316      END ;OF IF PTYBUF(R0)
                                L1050:.....
2782 072316 072316      END ;OF IF PTYFLAG(R0)
                                L1046:.....
2783 072316 062700 000002      END ;OF FOR R0
                                ADD #2,R0
  
```

072322	020027	000006		
072326	003722			
072330				
2784 072330			POP	CONTRL,R1,R0
072330	012637	177746		
072334	012601			
072336	012600			
2785 072340	162737	000001	002574	SUB #1,HUNGTIME
2786 072346	000207			RETURN

CMP R0,#6
BLE B211
E211::::::::::
MOV (SP),CONTRL
MOV (SP),R1
MOV (SP),R0
:CARRY IS ERROR FLAG ON RETURN

2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845

.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.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS   N              ;;CALL FOR TYPEOUT
*   .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;;M=1 OR 0
*                           ;;1=TYPE LEADING ZEROS
*                           ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   N              ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC   N              ;;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
  
```

000000
000001 072573
072575
000002
000406
000001 072573
000006 072575
000005 072572
072575
000006
072574
072573
000012
072574
177770

```

2846 072514 052703 000060
2847 072520 052703 000040
2848 072524 110337 072570
2849 072530
      072530 104401 072570

2850 072534 105337 072572
2851 072540 003347
2852 072542 002402
2853 072544 005204
2854 072546 000744
2855 072550 012605
2856 072552 012604
2857 072554 012603
2858 072556 016666 000002 000004
2859 072564 012616
2860 072566 000002
2861 072570      000
2862 072571      000
2863 072572      000
2864 072573      000
2865 072574 000000

      5$:  BIS #'0,R3      ::MAKE THIS DIGIT ASCII
          BIS #' ,R3      ::MAKE ASCII IF NOT ALREADY
          MOVB R3,8$      ::SAVE FOR TYPING
          TYPE 8$         ::GO TYPE THIS DIGIT
          TYPEIT ,8$
          .DSABL CRF

      6$:  DECB $OCNT      ::COUNT BY 1
          BGT 2$          ::BR IF MORE TO DO
          BLT 7$          ::BR IF DONE
          INC R4          ::INSURE LAST DIGIT ISN'T A BLANK
          BR 2$           ::GO DO THE LAST DIGIT

      7$:  MOV (SP)+,R5    ::RESTORE R5
          MOV (SP)+,R4    ::RESTORE R4
          MOV (SP)+,R3    ::RESTORE R3
          MOV 2(SP),4(SP) ::SET THE STACK FOR RETURNING
          MOV (SP)+,(SP)

      8$:  .BYTE 0        ::RETURN
          .BYTE 0        ::STORAGE FOR ASCII DIGIT
          .BYTE 0        ::TERMINATOR FOR TYPE ROUTINE
          $OCNT: .BYTE 0  ::OCTAL DIGIT COUNTER
          $OFILL: .BYTE 0 ::ZERO FILL SWITCH
          $OMODE: .WORD 0 ::NUMBER OF DIGITS TO TYPE
    
```

```

2867          .SBTTL  ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
2868          :*****
2869          :*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2870          :*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2871          :*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2872          :*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2873          :*REPLACED WITH SPACES.
2874          :*CALL:
2875          :*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
2876          :*      TYPDS   R0,R1,R2,R3,R5   ;;GO TO THE ROUTINE
2877          $TYPDS: PUSH
2878          072576 010046                                MOV R0,-(SP)
2879          072600 010146                                MOV R1,-(SP)
2880          072602 010246                                MOV R2,-(SP)
2881          072604 010346                                MOV R3,-(SP)
2882          072606 010546                                MOV R5,-(SP)
2883          072610 012746 020200                        MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
2884          072614 016605 000020                        MOV 20(SP),R5    ;;GET THE INPUT NUMBER
2885          072620 100004                                BPL 1$          ;;BR IF INPUT IS POS.
2886          072622 005405                                NEG R5          ;;MAKE THE BINARY NUMBER POS.
2887          072624 112766 000055 000001                MOVB #'-,1(SP)  ;;MAKE THE ASCII NUMBER NEG.
2888          072632 005000                                CLR R0          ;;ZERO THE CONSTANTS INDEX
2889          072634 012703 073012                        MOV #SDBLK,R3   ;;SETUP THE OUTPUT POINTER
2890          072640 112723 000040                        MOVB #' ,(R3)+  ;;SET THE FIRST CHARACTER TO A BLANK
2891          072644 005002                                CLR R2          ;;CLEAR THE BCD NUMBER
2892          072646 016001 073002                        MOV $DTBL(R0),R1 ;;GET THE CONSTANT
2893          072652 160105                                SUB R1,R5       ;;FORM THIS BCD DIGIT
2894          072654 002402                                BLT 4$          ;;BR IF DONE
2895          072656 005202                                INC R2          ;;INCREASE THE BCD DIGIT BY 1
2896          072660 000774                                BR 3$
2897          072662 060105                                ADD R1,R5       ;;ADD BACK THE CONSTANT
2898          072664 005702                                TST R2          ;;CHECK IF BCD DIGIT=0
2899          072666 001002                                BNE 5$          ;;FALL THROUGH IF 0
2900          072670 105716                                TSTB (SP)      ;;STILL DOING LEADING 0'S?
2901          072672 100407                                BMI 7$          ;;BR IF YES
2902          072674 106316                                ASLB (SP)      ;;MSD?
2903          072676 103003                                BCC 6$          ;;BR IF NO
2904          072700 116663 000001 177777                MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
2905          072706 052702 000060                        BIS #'0,R2     ;;MAKE THE BCD DIGIT ASCII
2906          072712 052702 000040                        BIS #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2907          072716 110223                                MOVB R2,(R3)+  ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2908          072720 005720                                TST (R0)+     ;;JUST INCREMENTING
2909          072722 020027 000010                        CMP R0,#10     ;;CHECK THE TABLE INDEX
2910          072726 002746                                BLT 2$          ;;GO DO THE NEXT DIGIT
2911          072730 003002                                BGT 8$          ;;GO TO EXIT
2912          072732 010502                                MOV R5,R2     ;;GET THE LSD
2913          072734 000764                                BR 6$          ;;GO CHANGE TO ASCII
2914          072736 105726                                TSTB (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
2915          072740 100003                                BPL 9$          ;;BR IF NO
2916          072742 116663 177777 177776                MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
2917          072750 105013                                CLRB (R3)     ;;SET THE TERMINATOR
2918          072752                                POP R5,R3,R2,R1,R0
2919          072752 012605                                MOV (SP)+,R5
2920          072754 012603                                MOV (SP)+,R3
2921          072756 012602                                MOV (SP)+,R2
2922          072760 012601                                MOV (SP)+,R1
2923          072762 012600                                MOV (SP)+,R0
    
```



```

2924 .SBTTL ROUTINE TTY INPUT
2925 :*****
2926 :*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2927 :*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2928 :*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2929 :*WHEN OPERATING IN TTY FLAG MODE.
2930 .ENABLE LSB
2931 073022 $CKSWR: IF CPUBIT NE #MASTER ;SLAVES DON'T GET KEYBOARDS!
      073022 023727 002114 000020 ;CMP CPUBIT,#MASTER
      073030 001401 ;BEQ L1052
2932 073032 000002 RTI
2933 073034 END ;OF IF CPUBIT
      073034 L1052:::
2934 073034 105777 107732 TSTB @STKS ;:CHAR THERE?
2935 073040 100125 BPL 12$ ;:IF NO, DON'T WAIT AROUND
2936 073042 117746 107726 MOVB @STKB,-(SP) ;:SAVE THE CHAR
2937 073046 042716 177600 BIC #^C177,(SP) ;:STRIP-OFF THE ASCII
2938 073052 022716 000006 CONTS1: CMP #6,(SP) ;:IS IT CONTROL F?
2939 073056 001002 BNE 1$ ;:NO SKIP
2940 073060 004737 043774 CALL FIELDSERVICE
2941 073064 022716 000024 1$: CMP #24,(SP) ;:IS IT CONTROL T?
2942 073070 001002 BNE 16$ ;:NO - SKIP
2943 073072 004737 073404 CALL CONTT ;:YES - CALL CONTROL T ROUTINE
2944 073076 022716 000003 16$: CMP #3,(SP) ;:IS IT CONTROL C?
2945 073102 001451 BEQ 5$ ;:YES EXIT *****NOTE***** STACK IS SCREWED UP!
2946 073104 022716 000004 CMP #4,(SP) ;:IS IT CONTROL D?
2947 073110 001002 BNE 2$ ;:NO - SKIP
2948 073112 000137 075774 CONTP: JMP @#0.ODT ;:YES - GO TO ODT
2949 073116 022716 000023 2$: CMP #23,(SP) ;:IS IT CONTROL S?
2950 073122 001002 BNE 6$ ;:NO - SKIP
2951 073124 004737 073544 CALL CONTS ;:YES - CALL CONTROL S ROUTINE
2952 073130 022737 000176 002766 6$: CMP #SWREG,SWR ;:IS THE SOFT-SWR SELECTED?
2953 073136 001067 BNE CKEND ;:BRANCH IF NO
2954 073140 022726 000007 CMP #7,(SP)+ ;:IS IT A CONTROL G?
2955 073144 001064 BNE CKEND ;:NO, RETURN TO USER
2956 073146 005737 002066 TST $AUTO ;:ARE WE RUNNING IN AUTO-MODE?
2957 073152 001061 BNE CKEND ;:BRANCH IF YES
2958 073154 TYPE $CNTLG ;:ECHO THE CONTROL-G (^G)
      073154 104401 074236 TYPEIT , $CNTLG
2959 073160 $GTSWR: TYPE $MSWR ;:TYPE CURRENT CONTENTS
      073160 104401 074243 TYPEIT , $MSWR
      .DSABL CRF
2960 073164 TYPOCT @SWR ;:OF THE SWR
      073164 017746 107576 MOV @SWR,-(SP) ;:SAVE @SWR FOR TYPEOUT
      073170 104402 TYPOC @SWR ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
2961 073172 TYPE $MNEW ;:PROMPT FOR NEW SWR
      073172 104401 074254 TYPEIT , $MNEW
      .DSABL CRF
2962 073176 005046 3$: CLR -(SP) ;:CLEAR COUNTER
2963 073200 005046 CLR -(SP) ;:THE NEW SWR
2964 073202 105777 107564 4$: TSTB @STKS ;:CHAR THERE?
2965 073206 100375 BPL 4$ ;:IF NOT TRY AGAIN
2966 073210 117746 107560 MOVB @STKB,-(SP) ;:PICK UP CHAR
2967 073214 042716 177600 BIC #^C177,(SP) ;:MAKE IT 7-BIT ASCII
2968 073220 021627 000003 CMP (SP),#3 ;:IS IT A CONTROL-C?
  
```

2969	073224	001006			BNE	7\$::BRANCH IF NOT
2970	073226			5\$:	TYPE	\$CNTLC	::YES, ECHO CONTROL-C (^C)
	073226	104401	074224		TYPEIT	,\$CNTLC	
					.DSABL	CRF	
2971	073232	062706	000006		ADD	#6,SP	::CLEAN UP STACK
2972	073236	000137	043416		JMP	BOOT	::CONTROL-C RESTART
2973	073242	021627	000025	7\$:	CMP	(SP),#25	::IS IT A CONTROL-U?
2974	073246	001005			BNE	9\$::BRANCH IF NOT
2975	073250				TYPE	\$CNTLU	::YES, ECHO CONTROL-U (^U)
	073250	104401	074231		TYPEIT	,\$CNTLU	
					.DSABL	CRF	
2976	073254	062706	000006	8\$:	ADD	#6,SP	::IGNORE PREVIOUS INPUT
2977	073260	000746			BR	3\$::LET'S TRY IT AGAIN
2978	073262	021627	000015	9\$:	CMP	(SP),#15	::IS IT A <CR>?
2979	073266	001016			BNE	13\$::BRANCH IF NO
2980	073270	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
2981	073274	001403			BEQ	10\$::BRANCH IF YES
2982	073276	016677	000002	107462	MOV	2(SP),@SWR	::SAVE NEW SWR
2983	073304	062706	000006	10\$:	ADD	#6,SP	::CLEAR UP STACK
2984	073310				TYPE	\$CRLF	::ECHO <CR> AND <LF>
	073310	104401	003010		TYPEIT	,\$CRLF	
					.DSABL	CRF	
2985	073314	000002		12\$:	RTI		::RETURN
2986	073316	062706	000002	CKEND:	ADD	#2,SP	::FIX STACK
2987	073322	000002			RTI		::RETURN
2988	073324	004737	057076	13\$:	CALL	\$TYPEC	::ECHO CHAR
2989	073330	021627	000060		CMP	(SP),#60	::CHAR < 0?
2990	073334	002420			BLT	15\$::BRANCH IF YES
2991	073336	021627	000067		CMP	(SP),#67	::CHAR > 7?
2992	073342	003015			BGT	15\$::BRANCH IF YES
2993	073344	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
2994	073350	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
2995	073354	001403			BEQ	14\$::BRANCH IF YES
2996	073356	006316			ASL	(SP)	::NO, SHIFT PRESENT
2997	073360	006316			ASL	(SP)	:: CHAR OVER TO MAKE
2998	073362	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
2999	073364	005266	000002	14\$:	INC	2(SP)	::KEEP COUNT OF CHAR
3000	073370	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
3001	073374	000702			BR	4\$::GET THE NEXT ONE
3002	073376			15\$:	TYPE	\$QUES	::TYPE ?<CR><LF>
	073376	104401	003007		TYPEIT	,\$QUES	
					.DSABL	CRF	
3003	073402	000724			BR	8\$::SIMULATE CONTROL-U
3004					.DSABL	LSB	

```

3007 073404          CONTT: SUBTST <<CONTROL T>>
:*****
:*SUBTEST          CONTROL T
:*****
3008 073404          PUSH      R0
073404 010046                                MOV R0,-(SP)
3009 073406          TYPE      $CRLF
073406 104401 003010 TYPEIT   , $CRLF
                                .DSABL  CRF
3010 073412          IF MDISPLAY LT #0 AND MDISPLAY GT #-256.
073412 005737 002676                                TST MDISPLAY
073416 002015                                BGE L1053
073420 023727 002676 177400                        CMP MDISPLAY,#-256.
073426 003411                                BLE L1053
3011                                     ;WE ARE IN MULTIPROCESSOR MODE
3012 073430 013700 002676 MOV      MDISPLAY,R0
3013 073434 005400 NEG      R0
3014 073436          TYPE      MSG100                ;MULTIPROCESSOR TEST #
073436 104401 116256 TYPEIT   ,MSG100
                                .DSABL  CRF
3015 073442          TYPOCS   R0,,3                ;TYPE 3 DIGITS
073442 010046 MOV      R0,-(SP)                ;;SAVE R0 FOR TYPEOUT
073444 104403 TYPOS                                ;;GO TYPE--OCTAL ASCII
073446 003                                .BYTE 3                ;;TYPE 3 DIGIT(S)
073447 000                                .BYTE 0                ;;SUPPRESS LEADING ZEROS
                                .DSABL  CRF
3016 073450          ELSE
073450 000433                                BR L1054
073452                                L1053:;;;;;
3017 073452          IF RLFLAG IS TRUE
073452 005737 002134                                TST RLFLAG
073456 001402                                BEQ L1055
3018 073460          TYPE      MSG092                ;RELOCATED
073460 104401 116046 TYPEIT   ,MSG092
                                .DSABL  CRF
3019 073464          END ;OF IF RLFLAG
073464                                L1055:;;;;;
3020 073464          TYPE      MSG093                ;BANK=
073464 104401 116062 TYPEIT   ,MSG093
                                .DSABL  CRF
3021 073470          TYPOCS   BANK,,3                ;TYPE 3 DIGITS
073470 013746 002106 MOV      BANK,-(SP)                ;;SAVE BANK FOR TYPEOUT
073474 104403 TYPOS                                ;;GO TYPE--OCTAL ASCII
073476 003                                .BYTE 3                ;;TYPE 3 DIGIT(S)
073477 000                                .BYTE 0                ;;SUPPRESS LEADING ZEROS
                                .DSABL  CRF
3022 073500          TYPE      MSG094                ;MAR=
073500 104401 116070 TYPEIT   ,MSG094
                                .DSABL  CRF
3023 073504 013700 177750 MOV      MAINT,R0
3024 073510 006200 ASR      R0
3025 073512 042700 177770 BIC      #^C7,R0
3026 073516          TYPOCS   R0,,1                ;TYPE 1 DIGIT
073516 010046 MOV      R0,-(SP)                ;;SAVE R0 FOR TYPEOUT
073520 104403 TYPOS                                ;;GO TYPE--OCTAL ASCII
073522 001                                .BYTE 1                ;;TYPE 1 DIGIT(S)
073523 000                                .BYTE 0                ;;SUPPRESS LEADING ZEROS
    
```

```
3027 073524      .DSABL CRF
073524 104401 116077  TYPE MSG095      ;PAT=
                                TYPEIT ,MSG095
                                .DSABL CRF
3028 073530      TYPOCS REALPAT,,2      ;TYPE 2 DIGITS
073530 013746 002366  MOV REALPAT,-(SP)  ;;SAVE REALPAT FOR TYPEOUT
073534 104403      TYPOS      ;;GO TYPE--OCTAL ASCII
073536 002        .BYTE 2      ;;TYPE 2 DIGIT(S)
073537 000        .BYTE 0      ;;SUPPRESS LEADING ZEROS
                                .DSABL CRF
3029 073540      END ;OF IF @DISPLAY
073540
3030 073540      POP R0      L1054:;:::::
073540 012600      RETURN      MOV (SP)+,R0
3031 073542 000207
3032
3033 073544      CONTS: SUBTST <<CONTROL S & CONTROL Q>>
;*****
;*SUBTEST CONTROL S & CONTROL Q
;*****
3034 073544      POP R0      ;GET RID OF RETURN ADDRESS FROM STACK
073544 012600      MOV (SP)+,R0
3035 073546 105777 107220      CONTS2: TSTB @STKS      ;WAIT FOR CHARACTER
3036 073552 100375      BPL CONTS2
3037 073554 117716 107214      MOVB @STKB,(SP)      ;REPLACE OVER OLD CHARACTER ON STACK
3038 073560 042716 177600      BIC #^C177,(SP)      ;STRIP ALL BUT ASCII
3039 073564      IF (SP) EQ #21      ;IF IT IS A CONTROL Q
073564 021627 000021      CMP (SP),#21
073570 001003      BNE L1056
3040 073572 000137 073052      JMP CONTS1
3041 073576      ELSE
073576 000401      BR L1057
073600      L1056:;:::::
3042 073600 000762      BR CONTS2
3043 073602      END ;OF IF (SP)
073602      L1057:;:::::
```

```

3045
3046
3047
3048
3049
3050
3051
3052
3053 073602 011646
3054 073604 016666 000004 000002
3055 073612 105777 107154
3056 073616 100375
3057 073620 117766 107150 000004
3058 073626 042766 177600 000004
3059 073634 026627 000004 000023
3060 073642 001013
3061 073644 105777 107122
3062 073650 100375
3063 073652 117746 107116
3064 073656 042716 177600
3065 073662 022627 000021
3066 073666 001366
3067 073670 000750
3068 073672 026627 000004 000140
3069 073700 002407
3070 073702 026627 000004 000175
3071 073710 003003
3072 073712 042766 000040 000004
3073 073720 000002
3074
3075
3076
3077
3078
3079
3080 073722 010346
3081 073724 005046
3082 073726 012703 074200
3083 073732 022703 074224
3084 073736 101467
3085 073740 104411
3086 073742 112613
3087 073744 122713 000003
3088 073750 001006
3089 073752 104401 074224
3090 073756 005726
3091 073760 012603
3092 073762 000137 043416
3093 073766 122713 000177
3094 073772 001022
3095 073774 005716
3096 073776 001007
3097 074000 112737 000134 074176
3098 074006 074006 104401 074176

*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
* RDCHR
* RETURN HERE
*
* INPUT A SINGLE CHARACTER FROM THE TTY
* 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 @STKS ;;WAIT FOR
BPL 1$ ;;A CHARACTER
MOVB @STKB,4(SP) ;;READ THE TTY
BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
CMP 4(SP),#23 ;;IS IT A CONTROL-S?
BNE 3$ ;;BRANCH IF NO
2$: TSTB @STKS ;;WAIT FOR A CHARACTER
BPL 2$ ;;LOOP UNTIL ITS THERE
MOVB @STKB,-(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 #STTYIN,R3 ;;GET ADDRESS
2$: CMP #STTYIN+20.,R3 ;;BUFFER FULL?
BLOS 8$ ;;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)
TYPEIT $CNTLC
.DSABL CRF
TST (SP)+ ;;CLEAN RUBOUT KEY OFF OF THE STACK
MOV (SP)+,R3 ;;RESTORE R3
JMP BOOT ;;GOTO CONTROL-C RESTART
3$: 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
TYPE 10$
TYPEIT ,10$
    
```

```

3099 074012 012716 177777          .DSABL CRF
3100 074016 005303          4$: MOV    #-1,(SP)      ;;SET THE RUBOUT KEY
3101 074020 020327 074200      DEC    R3           ;;BACKUP BY ONE
3102 074024 103434          CMP    R3,#$TTYIN  ;;STACK EMPTY?
3103 074026 111337 074176      BLO   8$           ;;BR IF YES
3104 074032 104401 074176      MOVB  (R3),10$     ;;SETUP TO TYPEOUT THE DELETED CHAR.
                                TYPE  10$
                                TYPEIT ,10$
                                .DSABL CRF
3105 074036 000735          BR    2$           ;;GO READ ANOTHER CHAR.
3106 074040 005716          5$: TST   (SP)      ;;RUBOUT KEY SET?
3107 074042 001406          BEQ   6$           ;;BR IF NO
3108 074044 112737 000134 074176 MOVB  #'\',10$     ;;TYPE A BACK SLASH
3109 074052 104401 074176      TYPE  10$
                                TYPEIT ,10$
                                .DSABL CRF
3110 074056 005016          CLR   (SP)        ;;CLEAR THE RUBOUT KEY
3111 074060 122713 000025      6$: CMPB  #25,(R3)   ;;IS CHARACTER A CTRL U?
3112 074064 001003          BNE   7$           ;;BR IF NO
3113 074066 104401 074231      TYPE  $CNTLU
                                TYPEIT , $CNTLU
                                .DSABL CRF
3114 074072 000715          BR    1$           ;;GO START OVER
3115 074074 122713 000022      7$: CMPB  #22,(R3)   ;;IS CHARACTER A '^R'?
3116 074100 001011          BNE   9$           ;;BRANCH IF NO
3117 074102 105013          CLRB  (R3)        ;;CLEAR THE CHARACTER
3118 074104 104401 003010      TYPE  $CRLF
                                TYPEIT , $CRLF
                                .DSABL CRF
3119 074110 104401 074200      TYPE  $TTYIN
                                TYPEIT , $TTYIN
                                .DSABL CRF
3120 074114 000706          BR    2$           ;;GO PICKUP ANOTHER CHACTER
3121 074116 104401 003007      8$: TYPE  $QUES
                                TYPEIT , $QUES
                                .DSABL CRF
3122 074122 000701          BR    1$           ;;CLEAR THE BUFFER AND LOOP
3123 074124 111337 074176      9$: MOVB  (R3),10$   ;;ECHO THE CHARACTER
3124 074130 104401 074176      TYPE  10$
                                TYPEIT ,10$
                                .DSABL CRF
3125 074134 122723 000015      CMPB  #15,(R3)+   ;;CHECK FOR RETURN
3126 074140 001274          BNE   2$           ;;LOOP IF NOT RETURN
3127 074142 105063 177777      CLRB  -1(R3)      ;;CLEAR RETURN (THE 15)
3128 074146 104401 003011      TYPE  $LF
                                TYPEIT , $LF
                                .DSABL CRF
3129 074152 005726          TST   (SP)+
3130 074154 012603          MOV   (SP)+,R3    ;;CLEAN RUBOUT KEY FROM THE STACK
3131 074156 011646          MOV   (SP),-(SP)  ;;RESTORE R3
3132 074160 016666 000004 000002 MOV   4(SP),2(SP)  ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3133 074166 012766 074200 000004 MOV   #$TTYIN,4(SP)
3134 074174 000002          RTI
3135 074176 000          10$: .BYTE 0
3136 074177 000          .BYTE 0
3137 074200 000024          $TTYIN: .REPT 20
3140 074224 136 103 015 $CNTLC: .ASCII / ^C / <15> <12>
                                ;;RESERVE SIZE BYTES FOR TTY INPUT
                                ;;CONTROL 'C'
    
```

3141	074227	012	000				
	074231	136	125	015	\$CNTLU: .ASCIZ	/^U/<15><12>	::CONTROL 'U'
	074234	012	000				
3142	074236	136	107	015	\$CNTLG: .ASCIZ	/^G/<15><12>	::CONTROL 'G'
	074241	012	000				
3143	074243	015	012	123	\$MSWR: .ASCIZ	<15><12>/SWR = /	
	074246	127	122	040			
	074251	075	040	000			
3144	074254	040	040	116	\$MNEW: .ASCIZ	/ NEW = /	
	074257	105	127	040			
	074262	075	040	000			
3145						.EVEN	

```

3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159 074266 011646
3160 074270 016666 000004 000002
3161 074276
      074276 010046
      074300 010146
      074302 010246
3162 074304 104412
3163 074306 012600
3164 074310 010037 074414
3165 074314 005001
3166 074316 005002
3167 074320 112046
3168 074322 001420
3169 074324 122716 000060
3170 074330 003026
3171 074332 122716 000067
3172 074336 002423
3173 074340 006301
3174 074342 006102
3175 074344 006301
3176 074346 006102
3177 074350 006301
3178 074352 006102
3179 074354 042716 177770
3180 074360 062601
3181 074362 000756
3182 074364 005726
3183 074366 010166 000012
3184 074372 010237 074434
3185 074376
      074376 012602
      074400 012601
      074402 012600
3186 074404 000002
3187 074406 005726
3188 074410 105010
3189 074412
      074412 104401
3190 074414 000000
3191 074416
      074416 104401 115067
3192 074422
      074422 104401 115107

```

```

.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
*                   ::PROVIDE SPACE FOR THE
*                   ::INPUT NUMBER
$RDOCT: MOV      (SP),-(SP)
MOV      4(SP),2(SP)
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

MOV      (SP)+,R2
MOV      (SP)+,R1
MOV      (SP)+,R0

RTI
4$:      TST          (SP)+
CLRB     (R0)
TYPE
TYPEIT
.DSABL   CRF
.WORD   0
5$:      TYPE         MSG062      ;INPUT MUST BE A
TYPEIT  .MSG062
.DSABL   CRF
TYPE         MSG063      ;N OCTAL
TYPEIT  .MSG063

```

3193	074426				.DSABL CRF		
	074426	104401	115120		TYPE MSG064	:NUMBER	
					TYPEIT .MSG064		
					.DSABL CRF		
3194	074432	000724			BR 1\$:: TRY AGAIN	
3195	074434	000000			\$HI OCT: .WORD 0	:: HIGH ORDER BITS GO HERE	
3196					.SBTTL	ROUTINE READ A DECIMAL NUMBER FROM THE TTY	
3197							
3198							
3199							
3200							
3201							
3202							
3203							
3204							
3205							
3206							
3207							
3208							
3209							
3210	074436	011646			\$RDDEC: MOV (SP),-(SP)	:: PROVIDE SPACE FOR	
3211	074440	016666	000004	000002	MOV 4(SP),2(SP)	:: THE INPUT NUMBER	
3212	074446				PUSH R0,R1,R2		
	074446	010046					MOV R0,-(SP)
	074450	010146					MOV R1,-(SP)
	074452	010246					MOV R2,-(SP)
3213	074454	104412			1\$: RDLIN	:: READ AN ASCII LINE	
3214	074456	012600			MOV (SP)+,R0	:: ADDRESS OF 1ST CHAR.	
3215	074460	010037	074604		MOV R0,6\$:: SAVE IN CASE OF BAD INPUT	
3216	074464	005046			CLR -(SP)	:: CLEAR DATA WORD	
3217	074466	005002			CLR R2	:: SIGN SET POSITIVE	
3218	074470	122710	000055		CMPB #'-(R0)	:: SEE IF A MINUS SIGN WAS TYPED	
3219	074474	001001			BNE 2\$:: BR IF NO MINUS SIGN	
3220	074476	112002			MOVB (R0)+,R2	:: SAVE FOR LATER USE	
3221	074500	112001			2\$: MOVB (R0)+,R1	:: PICKUP THIS CHARACTER	
3222	074502	001424			BEQ 3\$:: GET OUT IF ZERO	
3223	074504	122701	000060		CMPB #'0,R1	:: MAKE SURE THIS CHARACTER	
3224	074510	003032			BGT 5\$:: IS A DIGIT BETWEEN 0 & 9	
3225	074512	122701	000071		CMPB #'9,R1		
3226	074516	002427			BLT 5\$		
3227	074520	032716	170000		BIT #'(7777),(SP)	:: DON'T LET NUMBER GET TO BIG	
3228	074524	001024			BNE 5\$:: BR IF NUMBER WOULD OVERFLOW	
3229	074526	006316			ASL (SP)	:: *2	
3230	074530	011646			MOV (SP),-(SP)	:: SAVE FOR LATER	
3231	074532	006316			ASL (SP)	:: *4	
3232	074534	006316			ASL (SP)	:: *8	
3233	074536	062616			ADD (SP)+,(SP)	:: *10	
3234	074540	102416			BVS 5\$:: OVERFLOW ISN'T ALLOWED	
3235	074542	162701	000060		SUB #'0,R1	:: STRIP AWAY THE ASCII JUNK	
3236	074546	060116			ADD R1,(SP)	:: ADD IN THIS DIGIT	
3237	074550	102412			BVS 5\$:: OVERFLOW ISN'T ALLOWED	
3238	074552	000752			BR 2\$:: LOOP	
3239	074554	005702			3\$: TST R2	:: CHECK IF NUMBER IS NEG	
3240	074556	001401			BEQ 4\$:: BR IF NO	
3241	074560	005416			NEG (SP)	:: YES--NEGATE THE NUMBER	
3242	074562	012666	000012		4\$: MOV (SP)+,12(SP)	:: SAVE THE RESULT	
3243	074566				POP R2,R1,R0		

```

074566 012602
074570 012601
074572 012600
3244 074574 000002
3245
3246 074576 005726
3247 074600 105010
3248 074602
    074602 104401
3249 074604 000000
3250 074606
    074606 104401 115067
3251 074612
    074612 104401 115130
3252 074616
    074616 104401 115120
3253 074622 000714

RTI
5$: TST (SP)+
    (LRB (RO)
    TYPE
    TYPEIT
    .DSABL CRF
    .WORD 0
    TYPE MSG062
    TYPEIT ,MSG062
    .DSABL CRF
    TYPE MSG065
    TYPEIT ,MSG065
    .DSABL CRF
    TYPE MSG064
    TYPEIT ,MSG064
    .DSABL CRF
    BR 1$

::RETURN
::CLEAN PARTIAL NUMBER FROM STACK
::SET A TERMINATOR
::TYPE THE INPUT UP TO BAD CHAR.
::POINTER GOES HERE
:INPUT MUSST BE A
:DECIMAL
:NUMBER
::TRY AGAIN

MOV (SP),R2
MOV (SP),R1
MOV (SP),R0
  
```

```

3255 .SBTTL ROUTINE SAVE AND RESTORE R0-R5
3256
3257 .....
3258 ;*SAVE R0-R5
3259 ;*CALL:
3260 ;* SAVREG
3261 ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
3262 ;*
3263 ;*TOP---(+16)
3264 ;* +2---(+18)
3265 ;* +4---R5
3266 ;* +6---R4
3267 ;* +8---R3
3268 ;*+10---R2
3269 ;*+12---R1
3270 ;*+14---R0
3271
3272 074624 $SAVREG:
3273 074624 PUSH R0,R1,R2,R3,R4,R5
3274 074624 010046 MOV R0,-(SP)
3275 074626 010146 MOV R1,-(SP)
3276 074630 010246 MOV R2,-(SP)
3277 074632 010346 MOV R3,-(SP)
3278 074634 010446 MOV R4,-(SP)
3279 074636 010546 MOV R5,-(SP)
3280
3281 ;*RESTORE R0-R5
3282 ;*CALL:
3283 ;* RESREG
3284 074662 012666 000022 $RESREG:
3285 074666 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
3286 074672 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
3287 074676 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
3288 074702 012605 POP R5,R4,R3,R2,R1,R0
3289 074716 000002 RTI

```

```

MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)
MOV R3,-(SP)
MOV R4,-(SP)
MOV R5,-(SP)

```

```

MOV (SP)+,R5
MOV (SP)+,R4
MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0

```

3291
 3292
 3293
 3294
 3295
 3296
 3297
 3298
 3299
 3300
 3301

.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
    
```

3302 074720
 074720 010046
 074722 010146
 074724 010246
 3303 074726 013700 002716
 3304 074732 013701 002714
 3305 074736 012702 000007
 3306 074742 006300
 3307 074744 006101
 3308 074746 077203
 3309 074750 063700 002716
 3310 074754 005501
 3311 074756 063701 002714
 3312 074762 062700 001057
 3313 074766 005501
 3314 074770 062701 047401
 3315 074774 010037 002716
 3316 075000 010137 002714
 3317 075004
 075004 012602
 075006 012601
 075010 012600
 3318 075012 000207

```

$RAND: PUSH   R0,R1,R2                                MOV R0,-(SP)
                                                    MOV R1,-(SP)
                                                    MOV R2,-(SP)

    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

    MOV (SP)+,R2
    MOV (SP)+,R1
    MOV (SP)+,R0

    RETURN
    
```

```

3321          .SBTTL  ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
3322          :*****
3323          :*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
3324          :*UNSIGNED OCTAL ASCII NUMBER.
3325          :*CALL
3326          :*      MOV      #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
3327          :*      CALL    $DB20           ;; CALL THE ROUTINE
3328          :*      RETURN                    ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
3329
3330
3331 075014 104415          $DB20: SAVREG      ;; SAVE ALL REGISTERS
3332 075016 016601 000002  MOV      2(SP),R1      ;; PICKUP THE POINTER TO LOW WORD
3333 075022 012705 075133  MOV      #SOCTVL+13.,R5  ;; POINTER TO DATA TABLE
3334 075026 012704 000014  MOV      #12.,R4        ;; DO ELEVEN CHARACTERS
3335 075032 012703 177770  MOV      #^C7,R3        ;; MASK
3336 075036 012100          MOV      (R1)+,R0       ;; LOWER WORD
3337 075040 012101          MOV      (R1)+,R1       ;; HIGH WORD
3338 075042 005002          CLR      R2            ;; TERMINATOR
3339 075044 110245          1$:  MOVVB  R2,-(R5)     ;; PUT CHARACTER IN DATA TABLE
3340 075046 010002          MOV      R0,R2        ;; GET THIS DIGIT
3341 075050 005304          DEC      R4           ;; COUNT THIS CHARACTER
3342 075052 003007          BGT     3$           ;; BR IF NOT THE LAST DIGIT
3343 075054 001405          BEQ     2$           ;; BR IF IT IS THE LAST DIGIT
3344 075056 005205          INC     R5           ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
3345 075060 010566 000002  MOV      R5,2(SP)      ;; ASCII CHAR. & PUT IT ON THE STACK
3346 075064 104416          RESREG                    ;; RESTORE ALL REGISTERS
3347 075066 000207          RETURN                   ;; RETURN TO USER
3348 075070 006203          2$:  ASR     R3           ;; POSITION THE MASK FOR THE LAST DIGIT
3349 075072 006001          3$:  ROR     R1           ;; POSITION THE BINARY NUMBER FOR
3350 075074 006000          ROR     R0           ;; THE NEXT OCTAL DIGIT
3351 075076 006001          ROR     R1
3352 075100 006000          ROR     R0
3353 075102 006001          ROR     R1
3354 075104 006000          ROR     R0
3355 075106 040302          BIC     R3,R2        ;; MASK OUT ALL JUNK
3356 075110 062702 000060  ADD     #'0,R2        ;; MAKE THIS CHAR. ASCII
3357 075114 000753          BR      1$           ;; GO PUT IT IN THE DATA TABLE
3358 075116 000016          $OCTVL: .REPT 14.    ;; RESERVE DATA TABLE
3361          $OCT8=$OCTVL+4  ;; POINTER TO 11 DIGIT NUMBER

```

TABLES

```

3363          .SBTTL  TABLES
3364
3365          .SBTTL  APT MAILBOX-ETABLE
3366 075134    $MAIL:
3367 075134    $MSGTY: .WORD 0      ;;MESSAGE TYPE CODE
3368 075136    $FATAL: .WORD 0      ;;FATAL ERROR NUMBER (ERROR PC)
3369 075140    $TESTN: .WORD 0      ;;TEST NUMBER (DISPLAY REGISTER)
3370 075142    $PASS:  .WORD 0      ;;PASS COUNT
3371 075144    $DEVCT: .WORD 0      ;;DEVICE COUNT
3372 075146    $UNIT:  .WORD 0      ;;I/O UNIT NUMBER
3373 075150    $MSGAD: .WORD 0      ;;MESSAGE ADDRESS
3374 075152    $MSGLG: .WORD 0      ;;MESSAGE LENGTH
3375 075154    $ETABLE:
3376 075154    $ENV:   .BYTE 0      ;;ENVIRONMENT BYTE ;SET TO A 1 FOR APT AUTO MODE
3377 075155    $ENVM:  .BYTE 0      ;ENVIRONMENT MODE
3378
3379 075156    $SWREG: .WORD 1      ;;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
3380 075160    $USWR:  .WORD 0      ;;APT SWITCH REGISTER
3381 075162    $CPUOP: .WORD 30400  ;;USED TO LIMIT THE NUMBER OF PASSES
3382          ;;CPU TYPE,OPTIONS
3383          ;;BITS 15-11=CPU TYPE
3384          ;;11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
3385          ;;11/70=06,PDQ=07,Q=10
3386          ;;BIT 10=REAL TIME CLOCK
3387          ;;BIT 9=FLOATING POINT PROCESSOR
3388          ;;BIT 8=MEMORY MANAGEMENT
3388 075164    $MAMS1: .BYTE 1      ;;HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
3389 075165    $MTYP1: .BYTE 4      ;;MEM. TYPE,BLK#1
3390          ;;MEM.TYPE BYTE -- (HIGH BYTE)
3391          ;;900 NSEC CORE=001
3392          ;;300 NSEC BIPOLAR=002
3393          ;;500 NSEC MOS=003
3394          ;;MK11 ERROR CORRECTING MOS=004
3395 075166    $MADR1: .WORD 177776 ;;HIGH ADDRESS,BLK#1
3396          ;;MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
3397 075170    $MAMS2: .BYTE 0      ;;HIGH ADDRESS,M.S. BYTE
3398 075171    $MTYP2: .BYTE 0      ;;MEM.TYPE,BLK#2
3399 075172    $MADR2: .WORD 0      ;;MEM.LAST ADDRESS,BLK#2
3400 075174    $MAMS3: .BYTE 0      ;;HIGH ADDRESS,M.S.BYTE
3401 075175    $MTYP3: .BYTE 0      ;;MEM.TYPE,BLK#3
3402 075176    $MADR3: .WORD 0      ;;MEM.LAST ADDRESS,BLK#3
3403 075200    $MAMS4: .BYTE 0      ;;HIGH ADDRESS,M.S.BYTE
3404 075201    $MTYP4: .BYTE 0      ;;MEM.TYPE,BLK#4
3405 075202    $MADR4: .WORD 0      ;;MEM.LAST ADDRESS,BLK#4
3406 075204    $VECT1: .WORD 0      ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
3407 075206    $VECT2: .WORD 0      ;;INTERRUPT VECTOR#2BUS PRIORITY#2
3408 075210    $BASE:  .WORD 0      ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
3409 075212    $DEVMT: .WORD 0      ;;DEVICE MAP
3410
3411          ;IF EITHER OF THESE 2 WORDS ARE NON ZERO
3412          ;THEY WILL BE LOADED INTO ALL CSR'S & THE PROGRAM WILL STOP
3413 075214    $CDW1:  .WORD 0      ;CSR
3414 075216    $CDW2:  .WORD 0      ;CSR+2
3415
3416          ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
3417          ;ARE TO BE RUN FOR PARTICULAR MEMORIES
3418
3419          ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.

```

```
3420 ;BITO SET WILL RUN THE FIRST ENTRY IN THE TABLE, BITO SET
3421 ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
3422 ;
3423 ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
3424 ;
3425 075220 177777 $DDW0: .WORD 177777 ;MK11 CSR TESTS TABLE = MKCSRT:
3426 075222 177777 $DDW1: .WORD 177777 ;MK11 CSR TESTS TABLE = MKCSRT:
3427 075224 102777 $DDW2: .WORD 102777 ;MK11 PATTERNS TABLE = MKPAT:
3428 075226 177777 $DDW3: .WORD 177777 ;MK11 PATTERNS TABLE = MKPAT:
3429 075230 103777 $DDW4: .WORD 103777 ;MJ11 PATTERNS TABLE = MJPAT:
3430 075232 177777 $DDW5: .WORD 177777 ;MJ11 PATTERNS TABLE = MJPAT:
3431 075234 000001 $DDW6: .WORD 1 ;NUMBER OF CPU'S
3432 075236 $ETEND:
3433 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
3434 ;INTERFACE SPEC.
3435 ;
3436 075236 $APTHD:
3437 075236 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
3438 075240 075134 $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
3439 075242 000010 $TSTM: .WORD 8. ;;RUN TIM OF LONGEST TEST
3440 075244 000144 $PASTM: .WORD 100. ;;RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
3441 075246 000310 $UNITM: .WORD 200. ;;ADDITIONAL RUN TIME OF A PASS FOR EACH ADDITIONAL 128K
3442 075250 000041 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```

3444                                     .SBTTL ROUTINE TRAP DECODER
3445
3446                                     ;*****
3447                                     ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3448                                     ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3449                                     ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3450                                     ;*GO TO THAT ROUTINE.
3451
3452 075252 010046 $TRAP: MOV R0,-(SP) ;;SAVE R0
3453 075254 016600 000002 MOV 2(SP),R0 ;;GET TRAP ADDRESS
3454 075260 005740 TST -(R0) ;;BACKUP BY 2
3455 075262 111000 MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
3456 075264 006300 ASL R0 ;;POSITION FOR INDEXING
3457 075266 016000 075314 MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
3458 075272 000200 RTS R0 ;;GO TO ROUTINE
3459
3460
3461                                     ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3462
3463 075274 011646 $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
3464 075276 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
3465 075304 000002 RTI ;;RESTORE THE PSW
3466
3467 075306 $NOTRAP: TYPE MSG006 ;UNDEFINED TRAP INSTRUCTION
    075306 104401 111640 TYPEIT ,MSG006
    .DSABL CRF
3468 075312 000000 $HALT2: HALT
    
```

TRAP TABLE

.SBTTL TRAP TABLE

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

Address	Word	Trap	Routine
3471			
3472			
3473			
3474			
3475			
3476			
3477			
3478	075314	075274	
3479	075316	056752	
3480	075320	072374	
3481	075322	072350	
3482	075324	075306	
3483	075326	072576	
3484	075330	075306	
3485			
3486	075332	073160	
3487	075334	073022	
3488			
3489	075336	073602	
3490	075340	073722	
3491	075342	074266	
3492	075344	074436	
3493			
3494	075346	074624	
3495	075350	074662	
3496			
3497	075352	037020	
3498	075354	037030	
3499	075356	037040	
3500			
3501	075360	041016	
3502			
3503	075362	037050	
3504	075364	037066	
3505			
3506	075366	037076	
3507	075370	037220	
3508			
3509	075372	064574	
3510	075374	064622	
3511	075376	064650	
3512	075400	064700	
3513	075402	064762	
3514	075404	065004	
3515	075406	065034	
3516	075410	065054	
3517	075412	065076	
3518	075414	065116	
3519	075416	065140	
3520	075420	065162	
3521	075422	065202	
3522	075424	065220	
3523	075426	065236	
3524	075430	065256	
3525	075432	065274	
3526	075434	065312	
3527	075436	054570	

```

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)
$NOTRAP:$TYPON :CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS :CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
$NOTRAP:$TYPBN :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

```

TRAP TABLE

3528	075440	065502	\$PER26	:CALL=PERR26	TRAP+52(104452)	PROGRAM DETECTED ERROR
3529	075442	065522	\$PER27	:CALL=PERR27	TRAP+53(104453)	PROGRAM DETECTED ERROR
3530	075444	054762	\$PER30	:CALL=PERR30	TRAP+54(104454)	PROGRAM DETECTED ERROR
3531	075446	065720	\$PER31	:CALL=PERR31	TRAP+55(104455)	PROGRAM DETECTED ERROR
3532	075450	066016	\$PER32	:CALL=PERR32	TRAP+56(104456)	PROGRAM DETECTED ERROR
3533	075452	066064	\$PER33	:CALL=PERR33	TRAP+57(104457)	PROGRAM DETECTED ERROR
3534	075454	066144	\$PER34	:CALL=PERR34	TRAP+60(104460)	PROGRAM DETECTED ERROR
3535	075456	066176	\$PER35	:CALL=PERR35	TRAP+61(104461)	PROGRAM DETECTED ERROR
3536	075460	075306	\$NOTRAP	:CALL=PERR36	TRAP+62(104462)	PROGRAM DETECTED ERROR
3537	075462	075306	\$NOTRAP	:CALL=PERR37	TRAP+63(104463)	PROGRAM DETECTED ERROR
3538	075464	075306	\$NOTRAP	:CALL=PERR40	TRAP+64(104464)	PROGRAM DETECTED ERROR
3539	075466	075306	\$NOTRAP	:CALL=PERR41	TRAP+65(104465)	PROGRAM DETECTED ERROR
3540	075470	075306	\$NOTRAP	:CALL=PERR42	TRAP+66(104466)	PROGRAM DETECTED ERROR
3541	075472	075306	\$NOTRAP	:CALL=PERR43	TRAP+67(104467)	PROGRAM DETECTED ERROR
3542						
3543	075474	037466	\$ECCDIS	:CALL=ECCDIS	TRAP+70(104470)	DISABLE ECC ON ALL CSR'S
3544	075476	037522	\$ECC1DIS	:CALL=ECC1DIS	TRAP+71(104471)	DISABLE ECC ON 1 SELECTED CSR
3545	075500	037554	\$ECCINIT	:CALL=ECCINIT	TRAP+72(104472)	INITIALIZE ALL MK11 CSR'S
3546	075502	037574	\$ECC1INIT	:CALL=ECC1INIT	TRAP+73(104473)	INITIALIZE 1 SELECTED MK11 CSR
3547	075504	037700	\$CBCSR	:CALL=CBCSR	TRAP+74(104474)	WRITE GENERATED CHECKBITS IN ALL CSR'S
3548	075506	037742	\$CB1CSR	:CALL=CB1CSR	TRAP+75(104475)	WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
3549	075510	040002	\$WASSBE	:CALL=WASSBE	TRAP+76(104476)	WAS THERE A SBE ON ANY CSR?
3550	075512	040116	\$WAS1SBE	:CALL=WAS1SBE	TRAP+77(104477)	WAS THERE A SBE ON 1 SELECTED CSR?
3551	075514	040146	\$WASDBE	:CALL=WASDBE	TRAP+100(104500)	WAS THERE A DBE ON ANY CSR?
3552	075516	040262	\$WAS1DBE	:CALL=WAS1DBE	TRAP+101(104501)	WAS THERE A DBE ON 1 SELECTED CSR?
3553	075520	040322	\$CLRCSR	:CALL=CLRCSR	TRAP+102(104502)	CLEAR ALL CSR'S
3554	075522	040340	\$CLR1CSR	:CALL=CLR1CSR	TRAP+103(104503)	CLEAR 1 SELECTED CSR
3555	075524	040354	\$CHKDIS	:CALL=CHKDIS	TRAP+104(104504)	DISABLE ECC & WRITE CKBITS FROM ALL CSR'S
3556	075526	040410	\$CHK1DIS	:CALL=CHK1DIS	TRAP+105(104505)	DISABLE ECC & WRITE CKBITS FROM 1 CSR
3557	075530	037612	\$ENASBE	:CALL=ENASBE	TRAP+106(104506)	ENABLE TRAPS ON SBE'S FROM ALL CSR'S
3558	075532	037646	\$ENA1SBE	:CALL=ENA1SBE	TRAP+107(104507)	ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
3559	075534	037310	\$TSTRD	:CALL=TSTREAD	TRAP+110(104510)	TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES
3560	075536	040510	\$INVALID	:CALL=INVALID	TRAP+111(104511)	INVALIDATE BACKGROUND PATTERN ON BANK
3561	075540	075306	\$NOTRAP			
3562	075542	075306	\$NOTRAP			
3563	075544	075306	\$NOTRAP			
3564	075546	075306	\$NOTRAP			
3565	075550	075306	\$NOTRAP			
3566	075552	075306	\$NOTRAP			
3567	075554	075306	\$NOTRAP			
3568	075556	075306	\$NOTRAP			

		.SBTTL	ODT (CEMKA)	DATA AREA	
3571		ST	=	177776	:STATUS REGISTER
3572	177776	O.BKP	=	16	:NUMBER OF BREAKPOINTS-1 MULT. BY 2
3573	000016	O.TVEC	=	14	:TRT VECTOR LOCATION
3574	000014	O.STM	=	340	:PRIORITY MASK - STATUS REGISTER
3575	000340	O.TBT	=	20	:T-BIT MASK - STATUS REGISTER
3576	000020	TRT	=	000003	:TRT INSTRUCTION
3577	000003	O.RDB	=	177562	:R DATA BUFFER
3578	177562	O.RCSR	=	177560	:R C/SR
3579	177560	O.TDB	=	177566	:T DATA BUFFER
3580	177566	O.TCSR	=	177564	:T C/SR
3581	177564				

O.T: .BYTE 0

ODT (CEMKA) DATA AREA
 3598 075573 000
 3599
 3600
 3601 075574 000
 3602 075575 000
 3603 075576 000000
 3604 075600 000
 3605
 3606 075601 000
 3607
 3608
 3609 075602 000
 3610
 3611 075603 000
 3612
 3613 075604 000000

O.P: .BYTE 0
 O.CSR1: .BYTE 0
 O.CSR2: .BYTE 0
 O.FIL: 0
 O.CMFD: .BYTE 0
 O.SMFD: .BYTE 0
 O.SCRN: .BYTE 0
 O.MINS: .BYTE 0
 O.BIAS: .WORD 0

:PROCEED FLAG = -2 IF MANUAL ENTRY
 : -1 IF NO PROCEED ALLOWED
 : 0-7 IF PCEED ALLOWED
 :SAVE CELL - R C/SR
 :SAVE CELL - T C/SR
 :FILL COUNTER
 :COMMA FOUND SWITCH, =0 NO COMMA FOUND
 : =1 COMMA FOUND
 :SEMICOLON FOUND SWITCH
 : =0 NO SEMICOLON FOUND
 : =1 SEMICOLON FOUND
 :FLAG; 1=PASS SPACES ON FROM TTY
 :ALSO, IF =1, <LF> IS ECHOED
 :MINUS SIGN TYPED (SWITCH)
 :0=NO MINUS TYPED; 1=MINUS SIGN TYPED
 :CURRENT RELOCATION BIAS


```

3642      ; INITIALIZE ODT
3643      ; USE O.ODT FOR A NORMAL ENTRY
3644      ; USE O.ODT+2 TO RESTART ODT - WIPING OUT ALL BREAKPOINTS
3645      ; USE O.ODT+4 TO RE-ENTER (I.E. - FAKE A BREAKPOINT)
3646
3647 075774 000421      0.ODT: BR      2$      ;NORMAL ENTRY
3648 075776 000440      BR      3$      ;RESTART
3649 076000 004737 101172 1$: JSR      PC,O.RRST  ;RE-ENTER -- SAVE STATUS
3650 076004 013746 000016 MOV      O.TVEC+2,-(SP) ;SET UP LOCAL STATUS
3651 076010 004737 101202 JSR      PC,O.WST
3652 076014 010746      MOV      PC,-(SP)      ;FAKE THE PC
3653 076016 012637 075700 MOV      (SP)+,O.UPC
3654 076022 112737 177777 075573 MOVB     #-1,O.P      ;DISALLOW PROCEED
3655 076030 105037 075571 CLRB     O.S
3656 076034 000137 100060 JMP      O.BK1
3657
3658      ;RUN AT CURRENT STATUS
3659 076040 004537 100500 2$: JSR      5,O.SVTT  ;SAVE TTY STATUS
3660 076044 012737 076000 073114 MOV      #1$,@#CONTP+2
3661 076052 004037 100360 JSR      0,O.SVR      ;SAVE REGISTERS (MAINLY SP)
3662 076056 012706 075662 MOV      #O.UR0,SP    ;SET UP STACK
3663 076062 012704 101300 MOV      #O.ID,R4     ;TYPE ID
3664 076066 012703 101315 MOV      #O.IDND,R3
3665 076072 004537 101126 JSR      5,O.TYPE
3666 076076 000414      BR      4$
3667 076100 004037 100360 3$: JSR      0,O.SVR  ;SAVE REGISTERS
3668 076104 004537 100602 JSR      5,O.REM      ;REMOVE ALL BREAKPOINTS
3669 076110 113704 075704 MOVB     O.PRI,R4     ;GET ODT PRIORITY
3670 076114 106004      RORB     R4          ;SHIFT
3671 076116 106004      RORB     R4          ; INTO
3672 076120 106004      RORB     R4          ; POSITION
3673 076122 010446      MOV      R4,-(SP)    ;STORE IN STATUS
3674 076124 004737 101202 JSR      PC,O.WST
3675 076130 105037 075571 4$: CLRB     O.S      ;DISABLE SINGLE INSTRUCTION FOR NOW
3676 076134 112737 177777 075573 MOVB     #-1,O.P      ;DISALLOW PROCEED
3677 076142 012737 000340 000016 MOV      #O.STM,O.TVEC+2 ;STATUS WORD TO TRT VECTOR+2
3678 076150 012737 100042 000014 MOV      #O.BRK,O.TVEC ;PC TO TRT VECTOR
3679 076156 000137 077430 JMP      O.RALL      ;CLEAR BRK PT TABLES
3680
3681      ;O.CTLC
3682      ; ^C PROCESSING. RETURN TO KEYBOARD MONITOR
3683
3684
3685 076162 013706 075676 0.OTLC: MOV      O.USP,SP  ;RESTORE USER STACK
3686 076166 012704 076206 MOV      #1$,R4
3687 076172 012703 076207 MOV      #2$,R3
3688 076176 004537 101126 JSR      R5,O.TYPE    ;ECHO '^C'
3689 076202 000137 043416 JMP      BOOT
3690 076206      136
3691 076207      103
    
```

```

3694
3695 076210 013706 075676
3696 076214 012704 076240
3697 076220 012703 076241
3698 076224 004537 101126
3699 076230 004737 043774
3700 076234 000137 075774
3701 076240      136
3702 076241      106
3703
3704
3705 076242 012704 076266
3706 076246 012703 076267
3707 076252 004537 101126
3708 076256 012705 073116
3709 076262 000137 077624
3710 076266      136
3711 076267      105
3712
3713
3714
3715
3716 076270 004537 101006
3717 076274 012704 101345
3718 076300 120024
3719 076302 001414
3720 076304 022704 101365
3721 076310 101373
3722 076312 042700 177770
3723 076316 010004
3724 076320 006304
3725 076322 062704 075662
3726 076326 005202
3727 076330 000137 076554
3728 076334 162704 101336
3729 076340 000767
    
```

```

;CONTROL F PROCESSING
O.CTLF: MOV      O.USP,SP      ;RESTORE USER STACK
        MOV      #1$,R4
        MOV      #2$,R3
        JSR      R5,O.TYPE    ;ECHO ^F
        CALL     FIELDSERVICE
        JMP      O.ODT
1$:     .BYTE    '^'
2$:     .BYTE    '^F'

;CONTROL E PROCESSING
O.CTLE: MOV      #1$,R4
        MOV      #2$,R3
        JSR      R5,O.TYPE    ;ECHO ^E
        MOV      #CONTP+4,R5
        JMP      O.GOGO
1$:     .BYTE    '^'
2$:     .BYTE    '^E'

; SPECIAL NAME HANDLER
; DEPENDS UPON THE EXPLICIT ORDER OF THE TWO TABLES O.TL AND O.URO
O.REGT: JSR      5,O.GET      ;SPECIAL NAME, GET ONE MORE CHARACTER
        MOV      #O.TL,R4    ;TABLE START ADDRESS
1$:     CMPB     R0,(R4)+     ;IS THIS THE CORRECT CHARACTER?
        BEQ      3$         ;JUMP IF YES
        CMP     #O.TL+O.LG,R4 ;IS THE SEARCH DONE?
        BHI     1$         ;BRANCH IF NOT
        BIC     #177770,R0   ;MASK OFF OCTAL
        MOV     R0,R4
2$:     ASL     R4
        ADD     #O.URO,R4    ;GENERATE ADDRESS
        INC     R2          ;SET FOUND FLAG
        JMP     O.SCAN      ;GO FIND NEXT CHARACTER
3$:     SUB     #O.TL-7,R4   ;GO FIND NEXT CHARACTER
        BR     2$
    
```

```

3732      ; 'BACKARROW' HANDLER - OPEN INDEXED ON THE PC (BACK ARROW)
3733      .ENABL  LSB
3734 076342 004537 076416 0.ORPC: JSR 5,2$      ;TEST WORD MODE AND CLOSE
3735 076346 061202      ADD @R2,R2      ;COMPUTE
3736 076350 005202      INC R2
3737 076352 005202      INC R2      ; NEW ADDRESS
3738 076354 010237 075560 1$: MOV R2,0.CAD ;UPDATE CAD
3739 076360 000137 077226 JMP 0.OP2A      ;GO FINISH UP
3740 076364 004537 076416 0.ORAB: JSR 5,2$      ;TEST WORD MODE AND CLOSE
3741 076370 011202      MOV @R2,R2      ;GET ABSOLUTE ADDRESS
3742 076372 000770      BR 1$
3743 076374 004537 076416 0.ORRB: JSR 5,2$      ;TEST AND CLOSE
3744 076400 011201      MOV @R2,R1      ;COMPUTE NEW ADDRESS
3745 076402 110101      MOVB R1,R1      ;EXTEND THE SIGN
3746 076404 006301      ASL R1          ;R2=2(@R2)
3747 076406 005201      INC R1          ; +2
3748 076410 005201      INC R1
3749 076412 060102      ADD R1,R2      ; +PC
3750 076414 000757      BR 1$
3751 076416 004737 101214 2$: JSR PC,0.CLSE ;CLOSE CURRENT CELL
3752 076422 022737 000002 075566 CMP #2,0.BW ;ONLY WORD MODE ALLOWED
3753 076430 001003      BNE 3$         ;BRANCH IF ERROR
3754 076432 013702 075560 MOV 0.CAD,R2    ;CURRENT ADDRESS IN R2
3755 076436 000205      RTS R5
3756 076440 005726      3$: TST (SP)+
3757 076442 000137 076510 0.TCL2: JMP 0.ERR ;POP A WORD AND SHOW THE ERROR
3758      .DSABL  LSB
3759
3760      ; PROCESS S - SINGLE INSTRUCTION MODE
3761 076446 105737 075601 0.SNGL: TSTB 0.SMFD ;DONT REACT IF ; NOT TYPED
3762 076452 001773      BEQ 0.TCL2
3763 076454 005702      TST R2          ;SEE IF TURN ON OR TURN OFF
3764 076456 001004      BNE 1$         ;BRANCH IF TURNING IT ON
3765 076460 105037 075571 CLRB 0.S        ;CLEAR THE FLAG
3766 076464 000137 076520 JMP 0.DCD       ;CONTINUE THE SCAN
3767 076470 112737 177777 075571 1$: MOVB #-1,0.S ;SET THE FLAG
3768 076476 000137 076520 JMP 0.DCD
    
```

```

3771
3772
3773
3774 076502 105237 075603
3775 076506 000420
3776
3777
3778
3779
3780
3781 076510 012700 000077
3782 076514 004537 100730
3783 076520 005037 075566
3784 076524 004537 101254
3785 076530 105037 075601
3786 076534 105037 075600
3787 076540 105037 075603
3788 076544 005003
3789 076546 005005
3790 076550 005004
3791 076552 005002
3792 076554 004537 101006
3793 076560 022700 000060
3794 076564 101013
3795 076566 022700 000067
3796 076572 103410
3797 076574 042700 177770
3798 076600 006304
3799 076602 006304
3800 076604 006304
3801 076606 060004
3802 076610 005202
3803 076612 000760
3804 076614 005001
3805 076616 120061 101321
3806 076622 001405
3807 076624 005201
3808 076626 020127 000024
3809 076632 103326
3810 076634 000770
3811 076636 105737 075603
3812 076642 001401
3813 076644 005404
3814 076646 105737 075600
3815 076652 001402
3816 076654 063704 075604
3817 076660 105037 075603
3818 076664 006301
3819 076666 000171 076672

```

```

:MINUS PROCESSING
O.MIN: INCB O.MINS ;SET MINUS FOUND SWITCH ON
BR O.DCD1
; COMMAND DECODER - ODT11X
; ALL REGISTERS MAY BE USED (R0-R5),
O.ERR: MOV #'?,R0 ; ? TO BE TYPED
JSR 5,O.FTYP ; OUTPUT ?
O.DCD: CLR O.BW ;CLOSE ALL
JSR 5,O.CRLS ;TYPE <CR><LF>*
O.DCD3: CLRB O.SMFD ;SEMICOLON FOUND FLAG
CLRB O.CMFD ;COMMA FOUND FLAG
CLRB O.MINS ;MINUS SIGN FOUND FLAG
CLR R3 ;R3 IS A SAVE REGISTER FOR R2
CLR R5 ;R5 IS A SAVE REGISTER FOR R4
O.DCD1: CLR R4 ; R4 CONTAINS THE CONVERTED OCTAL
CLR R2 ; R2 IS THE NUMBER FOUND FLAG
O.SCAN: JSR 5,O.GET ;GET A CHAR, RETURN IN R0
CMP #'0,R0 ;COMPARE WITH ASCII 0
BHI 5$ ;CHECK LEGALITY IF NON-NUMERIC
CMP #'7,R0 ;COMPARE WITH ASCII 7
BLO 5$ ;CHECK LEGALITY IF NOT OCTAL
BIC #177770,R0 ;CONVERT TO BCD
ASL R4 ; MAKE ROOM
ASL R4 ; IN
ASL R4 ; R4
ADD R0,R4 ;PACK THREE BITS IN R4
INC R2 ;R2 HAS NUMERIC FLAG
BR O.SCAN ; AND TRY AGAIN
5$: CLR R1 ;CLEAR INDEX
1$: CMPB R0,O.LGCH(R1) ;DO THE CODES MATCH?
BEQ 2$ ;JUMP IF YES
INC R1 ; SET INDEX FOR NEXT SEARCH
CMP R1,#O.CLGT ;IS THE SEARCH DONE?
BHS O.ERR ; OOPS!
BR 1$ ;RE-LOOP
2$: TSTB O.MINS ;IF MINUS WAS NOT TYPED
BEQ 4$ ;DO NOT NEGATE K
NEG R4 ;OTHERWISE, TAKE 2'S COMPLEMENT.
4$: TSTB O.CMFD ;IF A COMMA NOT TYPED,
BEQ 3$ ;SKIP NEXT INSTRUCTION.
ADD O.BIAS,R4 ;OTHERWISE, ADD RELOC. BIAS TO (R4)
3$: CLRB O.MINS ;REINITIALIZE MINUS-TYPED SWITCH FOR NXT SCAN
ASL R1 ;MULTIPLY BY TWO
JMP @6$(R1) ;GO TO PROPER ROUTINE

```

3822 076672 076742
3823 076674 076760
3824 076676 076772
3825 076700 077140
3826 076702 076270
3827 076704 077576
3828 076706 077160
3829 076710 076342
3830 076712 077154
3831 076714 077302
3832 076716 077466
3833 076720 077324
3834 076722 077732
3835 076724 076364
3836 076726 076374
3837 076730 076446
3838 076732 076502
3839 076734 076162
3840 076736 076210
3841 076740 076242

6\$: O.SEMI : : SEMICOLON
O.WRD : / OPEN WORD
O.BYT : \ OPEN BYTE -BACK SLASH
O.CRET : CARRIAGE RETURN CLOSE
O.REGT : \$ REGISTER OPS
O.GO : G GO TO ADDRESS K
O.OP1 : <LF> MODIFY, CLOSE, OPEN NEXT
O.ORPC : 'BACKARROW' OPEN RELATED, INDEX - PC (BACK ARROW)
O.OLD : < RETURN TO OLD SEQUENCE AND OPEN
O.BACK : ^ OPEN PREVIOUS (UP ARROW)
O.OFST : O OFFSET
O.BKPT : B BREAKPOINTS
O.PROC : P PROCEED
O.ORAB : @ OPEN RELATED, ABSOLUTE
O.ORRB : > OPEN RELATED, REL. BRANCH
O.SNGL : S SINGLE INSTRUCTION MODE
O.MIN : - MINUS, NEGATES NUMBER TYPED IN
O.CTLC : ^C EXIT TO MONITOR
O.CTLF : ^F EXIT TO FIELD SERVICE MODE
O.CTLE : ^E PROCEED FROM ^D

```

3844
3845
3846
3847 076742 010203
3848 076744 010405
3849 076746 105237 075601
3850 076752 105037 075600
3851 076756 000674
3852
3853
3854
3855
3856
3857
3858 076760 012737 000002 075566 0.WRD:
3859 076766 000404
3860 076770 006104
3861 076772 012737 000001 075566 0.BYT:
3862 077000 005702
3863 077002 001011
3864 077004 105737 075600
3865 077010 001402
3866 077012 000137 076510
3867 077016 105737 075601
3868 077022 001373
3869 077024 000404
3870 077026 010437 075562
3871 077032 010437 075560
3872 077036 022737 000001 075566 0.WRD1:
3873 077044 001407
3874 077046 013704 075560
3875 077052 006204
3876 077054 103745
3877 077056 017700 176476
3878 077062 000402
3879 077064 117700 176470
3880 077070 004537 100634
3881 077074 022737 000001 075566
3882 077102 001212
3883 077104 112700 000075
3884 077110 004537 100730
3885 077114 117700 176440
3886 077120 004537 100730
3887 077124 112700 000040
3888 077130 004537 100730
3889 077134 000137 076530
3890

; SEMI-COLON PROCESSOR
0.SEMI: MOV R2,R3 ;A SEMI-COLON HAS BEEN RECEIVED
        MOV R4,R5 ;NUMERIC FLAG TO R3, CONTENTS TO R5
        INCB 0.SMFD ;SET SEMICOLON FOUND FLAG
        CLRB 0.CMFD ;RESET COMMA FOUND FLAG
        BR 0.DCD1 ;GO BACK FOR MORE

; PROCESS / AND \ - OPEN WORD OR BYTE
;INPUT - IF R2 IS NON-ZERO A NEW REXP HAS BEEN TYPED IN
;INPUT - -ADDRESS OF WORD TO BE OPENED IS IN R4
        .ENABL LSB
0.WRD: MOV #2,0.BW ;OPEN WORD
        BR 11$
1$: ROL R4 ;GET THE ADDRESS BACK
0.BYT: MOV #1,0.BW ;OPEN BYTE
11$: TST R2 ;GET VALUE IF R2 IS NON-ZERO
        BNE 14$ ;BRANCH IF NUMBER INPUT
        TSTB 0.CMFD ;TEST FOR ',' 'AND';
        BEQ 12$
13$: JMP 0.ERR ;ERROR IF PRESENT WITHOUT NUMBER.
12$: TSTB 0.SMFD
        BNE 13$
        BR 0.WRD1 ;NO NUMBER - REOPEN PREVIOUS LOCATION
14$: MOV R4,0.DOT ;PUT VALUE IN DOT
        MOV R4,0.CAD ;ALSO IN CAD
0.WRD1: CMP #1,0.BW ;CHECK BYTE MODE
        BEQ 22$ ;JUMP IF BYTE
        MOV 0.CAD,R4
        ASR R4 ;MOVE ONE BIT TO CARRY
        BCS 1$ ;JUMP IF ODD ADDRESS
        MOV @0.CAD,R0 ;GET CONTENTS OF WORD
        BR 23$
22$: MOVB @0.CAD,R0 ;GET CONTENTS OF BYTE
23$: JSR 5,0.CADV ;GO GET AND TYPE OUT @CAD
        CMP #1,0.BW ;CHECK IF BYTE MODE.
        BNE 0.DCD3 ;IF NOT WE'RE DONE. ELSE:
        MOVB #'=,R0 ;TYP '=' AND THEN THE ASCII BYTE
        JSR 5,0.FTYP
        MOVB @0.CAD,R0
        JSR 5,0.FTYP
        MOVB #' ,R0
        JSR 5,0.FTYP
        JMP 0.DCD3 ;GO BACK TO DECODER
        .DSABL LSB
    
```

```

3893      ; PROCESS CARRIAGE RETURN
3894 077140 004737 101214  O.CRET: JSR   PC,O.CLSE  ;CLOSE LOCATION
3895 077144 000137 076520  O.DCDA: JMP   O.DCD      ;RETURN TO DECODER
3896
3897 077150 000137 076510  O.ERR3: JMP   O.ERR      ; INTERMEDIATE HELP
3898
3899      ; PROCESS <LF>, OPEN NEXT WORD
3900
3901 077154 105237 075570  O.OLD:  INCB  O.SEQ      ;SET FLAG TO LATER RESTORE CAD
3902 077160 005737 075566  O.OP1:  TST   O.BW      ;<LF> RECEIVED
3903 077164 001771          O.ERR2: BEQ   O.ERR3    ;ERROR IF NOTHING IS OPEN
3904 077166 004737 101214          JSR   PC,O.CLSE    ;CLOSE PRESENT CELL
3905 077172 105737 075570          TSTB  O.SEQ      ;SHOULD CAD BE RESTORED?
3906 077176 001405          BEQ   1$         ;BRANCH IF NOT
3907 077200 013737 075562 075560  MOV   O.DOT,O.CAD    ;RESTORE PREVIOUS SEQUENCE
3908 077206 105037 075570          CLRB  O.SEQ      ;RESET FLAG; NO LONGER NEEDED
3909 077212 063737 075566 075560  1$:   ADD   O.BW,O.CAD ;GENERATE NEW ADDRESS
3910 077220 013737 075560 075562  O.OP2: MOV   O.CAD,O.DOT ;INITIALIZE DOT
3911 077226 004537 101246  O.OP2A: JSR   5,O.CRLF  ;<CR><LF>
3912 077232 013746 075566          MOV   O.BW,-(SP)    ;SAVE BW
3913 077236 012737 000002 075566  MOV   #2,O.BW      ;SET TO TYPE FULL WORD ADDRESS
3914 077244 013700 075560          MOV   O.CAD,R0    ;NUMBER TO TYPE
3915 077250 004537 101272          JSR   5,O.RORA     ; CHECK FORMAT
3916 077254 011637 075566          MOV   @SP,O.BW    ;RESTORE BW
3917 077260 012700 027534          MOV   #27534,R0  ;PUT '/' IN R0
3918 077264 022726 000001          CMP   #1,(SP)+   ;IS IT BYTE MODE?
3919 077270 001401          BEQ   1$         ;JUMP IF YES
3920 077272 000300          SWAB  R0         ;TYPE A /
3921 077274 004537 100730  1$:   JSR   5,O.FTYP   ;TYPE THE LOW BYTE OF R0
3922 077300 000656          BR   O.WRD1      ;GO PROCESS IT
3923

```

```

3926
3927 ; PROCESS ^, OPEN PREVIOUS WORD
3928
3929 077302 005737 075566 O.BACK: TST O.BW ; ^ RECEIVED
3930 077306 001726 BEQ O.ERR2 ;ERROR IF NOTHING OPEN
3931 077310 004737 101214 JSR PC,O.CLSE
3932 077314 163737 075566 075560 SUB O.BW,O.CAD ;GENERATE NEW ADDRESS
3933 077322 000736 BR O.OP2 ;GO DO THE REST
3934
3935 ; B HANDLER - SET AND REMOVE BREAKPOINTS
3936
3937 077324 012700 101370 O.BKPT: MOV #O.TRTC,R0
3938 077330 006304 ASL R4 ;MULTIPLY NUMBER BY TWO
3939 077332 005703 TST R3
3940 077334 001423 BEQ 3$ ;IF R3 IS ZERO GO REMOVE BREAKPOINT
3941 077336 006205 ASR R5 ;GET ONE BIT TO CARRY
3942 077340 103514 BCS O.ERR1 ;BADNESS IF ODD ADDRESS
3943 077342 006305 ASL R5 ;RESTORE ONE BIT
3944 077344 062704 075706 ADD #O.ADR1,R4
3945 077350 005702 TST R2
3946 077352 001007 BNE 1$ ;JUMP IF SPECIFIC CELL
3947 077354 020014 2$: CMP R0,@R4 ;IS THIS CELL FREE?
3948 077356 001405 BEQ 1$ ;JUMP IF YES
3949 077360 020427 075724 CMP R4,#O.BKP+O.ADR1 ;ARE WE AT THE END OF OUR ROPE
3950 077364 103102 BHIS O.ERR1 ;YES, THERE IS NOTHING FREE
3951 077366 005724 TST (R4)+ ;INCREMENT BY TWO
3952 077370 000771 BR 2$
3953 077372 020427 075724 1$: CMP R4,#O.BKP+O.ADR1
3954 077376 101075 BHI O.ERR1 ;ERROR IF TOO LARGE
3955 077400 010514 MOV R5,@R4 ;SET BREAKPOINT
3956 077402 000660 BR O.DCDA ;RETURN
3957 077404 005702 3$: TST R2
3958 077406 001410 BEQ O.RALL ;GO REMOVE ALL
3959 077410 020427 000016 CMP R4,#O.BKP
3960 077414 101066 BHI O.ERR1 ;JUMP IF NUMBER TOO LARGE
3961 077416 010064 075706 MOV R0,O.ADR1(R4) ;CLEAR BREAKPOINT
3962 077422 005064 075730 CLR O.CT(R4) ;CLEAR COUNT ALSO
3963 077426 000646 BR O.DCDA
3964 077430 005004 O.RALL: CLR R4
3965 077432 012700 101370 MOV #O.TRTC,R0
3966 077436 020427 000020 1$: CMP R4,#O.BKP+2 ;ALL DONE?
3967 077442 101240 BHI O.DCDA ;JUMP IF YES
3968 077444 010064 075706 MOV R0,O.ADR1(R4) ;RESET BKPT
3969 077450 012764 000003 075752 MOV #TRT,O.UIN(R4) ;RESET CONTENTS OF TABLE
3970 077456 005064 075730 CLR O.CT(R4) ;CLEAR COUNT
3971 077462 005724 TST (R4)+ ;INCREMENT BY TWO
3972 077464 000764 BR 1$
3973
    
```

```

3976
3977
3978
3979 077466 022737 000002 075566 0.OFST: CMP #2,O.BW :CHECK WORD MODE
3980 077474 001036 BNE O.ERR1 :ERROR IF NOT CORRECT MODE
3981 077476 012700 000040 MOV #' ,R0 :TYPE ONE BLANK
3982 077502 004537 100730 JSR 5,O.FTYP : AS A SEPARATOR
3983 077506 005703 TST R3 :WAS SEMI-COLON TYPED?
3984 077510 001430 BEQ O.ERR1 :NO, CALL IT AN ERROR
3985 077512 163705 075560 SUB O.CAD,R5 :COMPUTE
3986 077516 005305 DEC R5
3987 077520 005305 DEC R5 : 16 BIT OFFSET
3988 077522 010500 MOV R5,R0
3989 077524 004537 100634 JSR 5,O.CADV :NUMBER IN R0 - WORD MODE
3990 077530 010500 MOV R5,R0
3991 077532 006200 ASR R0 :DIVIDE BY TWO
3992 077534 103414 BCS 1$ :ERROR IF ODD
3993 077536 022700 177600 CMP #-200,R0 :COMPARE WITH -200
3994 077542 003011 BGT 1$ :DO NOT TYPE IF OUT OF RANGE
3995 077544 022700 000177 CMP #177,R0 :COMPARE WITH +177
3996 077550 002406 BLT 1$ :DO NOT TYPE IF OUT OF RANGE
3997 077552 005337 075566 DEC O.BW :SET TEMPORARY BYTE MODE
3998 077556 004537 100634 JSR 5,O.CADV :NUMBER IN R0 - BYTE MODE
3999 077562 005237 075566 INC O.BW :RESTORE WORD MODE
4000 077566 000137 076530 1$: JMP O.DCD3 :ALL DONE
4001
4002 077572 000137 076510 0.ERR1: JMP O.ERR :INTERMEDIATE HELP
    
```

```

4005          : PROCESS G - GO
4006
4007 077576   105737   075601   0.G0:  TSTB   0.SMFD   :WAS "" TYPED?
4008 077602   001773           BEQ    0.ERR1   :BR IF NOT TYPED
4009 077604   005703           TST    R3       :WAS K; TYPED?
4010 077606   001771           BEQ    0.ERR1   : TYPE ?<CR,LF> IF NOT
4011 077610   112737   000021   075573  MOVB   #0.BKP+3,0.P :CLEAR PROCEED
4012 077616   006205           ASR    R5       :CHECK LOW ORDER BIT
4013 077620   103764           BCS   0.ERR1   :ERROR IF ODD NUMBER
4014 077622   006305           ASL    R5       :RESTORE WORD
4015 077624   010537   075700   0.G0G0: MOV   R5,0.UPC  :SET UP NEW PC
4016 077630   012746   000340   MOV   #0.STM,-(SP) :SET HIGH PRIORITY
4017 077634   004737   101202   JSR   PC,0.WST
4018 077640   004537   100534   JSR   5,0.RSTT   :RESTORE TELETYPE
4019 077644   105037   075572   0.TBIT: CLRB  0.T    :CLEAR
4020 077650   052737   000020   075702  BIS   #0.TBT,0.UST : BOTH T-BIT FLAGS
4021 077656   105737   075571   TSTB  0.S       :SEE IF WE NEED A T BIT
4022 077662   001005           BNE   0.G02    :IF NOT GO NOW
4023 077664   042737   000020   075702  BIC   #0.TBT,0.UST :CLEAR THE T BIT
4024 077672   004537   100450   JSR   5,0.RSB   :RESTORE BREAKPOINTS
4025 077676   004037   100416   0.G02: JSR   0,0.RSR   :RESTORE REGISTERS
4026 077702   013746   075702   MOV   0.UST,-(SP) : AND STATUS
4027 077706   013746   075700   MOV   0.UPC,-(SP) : AND PC
4028 077712   000240           NOP
4029 077714   013746   075702   MOV   0.UST,-(SP) : CHANGE TO HALT FOR DEBUGGING
4030 077720   042716   177420   BIC   #0.TBT!177400,(SP) : MOV IN STATUS FIRST W/O T BIT
4031 077724   004737   101202   JSR   PC,0.WST  : SO INTERRUPTS CAN HAPPEN BEFORE
4032 077730   000006           RTT          : RTT TURNS ON THE T BIT.
    
```

```

4035 ; PROCESS P - PROCEED
4036 ; ONLY ALLOWED AFTER A BREAKPOINT
4037
4038 077732 105737 075601 0.PROC: TSTB 0.SMFD ;WAS ':' TYPED?
4039 077736 001715 BEQ 0.ERR1 ;BR IF NOT TYPED
4040 077740 113700 075573 MOV 0.P,R0
4041 077744 105700 TSTB R0 ;CHECK LEGALITY OF PROCEED
4042 077746 002711 BLT 0.ERR1 ;NOT LEGAL
4043 077750 005702 TST R2 ;CHECK FOR ILLEGAL COUNT
4044 077752 001307 BNE 0.ERR1 ;JUMP IF ILLEGAL
4045 077754 005703 TST R3 ;WAS COUNT SPECIFIED?
4046 077756 001402 BEQ 0.PR1 ;NO
4047 077760 010560 075730 MOV R5,0.CT(R0) ;YES, PUT AWAY COUNT
4048 077764 012746 000340 0.PR1: MOV #0.STM,-(SP) ;FORCE HIGH PRIORITY
4049 077770 004737 101202 JSR PC,0.WST
4050 077774 004537 100534 JSR 5,0.RSTT ;RESTORE TTY
4051 100000 123727 075573 000016 0.C1: CMPB 0.P,#0.BKP ;SEE IF A REAL ONE OR A FAKE
4052 100006 003316 BGT 0.TBIT ;BRANCH IF FAKE
4053 100010 105737 075571 TSTB 0.S ;SEE IF SINGLE INSTRUCTION MODE
4054 100014 001313 BNE 0.TBIT ;IF SO EXIT NOW
4055 100016 012746 000340 MOV #0.STM,-(SP) ;SET HIGH PRIORITY
4056 100022 004737 101202 JSR PC,0.WST
4057 100026 105237 075572 INCB 0.T ;SET T-BIT FLAG
4058 100032 052737 000020 075702 BIS #0.TBT,0.UST ;SET T-BIT
4059 100040 000716 BR 0.G02

```

```

4062          ; BREAKPOINT HANDLER
4063 100042 012637 075700 0.BRK: MOV (SP)+,0.UPC ;PRIORITY IS 7 UPON ENTRY
4064 100046 012637 075702          MOV (SP)+,0.UST ;SAVE STATUS AND PC
4065 100052 112737 000021 075573 MOVB #0.BKP+3,0.P ;TELL ;P THAT WE CAN CONTINUE
4066 100060 004037 100360 0.BK1: JSR 0,0.SVR ;SAVE VARIOUS REGISTERS
4067 100064 105737 075572          TSTB 0,T ;CHECK FOR T-BIT SET
4068 100070 001265          BNE 0,TBIT ;JUMP IF SET
4069 100072 004537 100602          JSR 5,0.REM ;REMOVE BREAKPOINTS
4070 100076 105737 075704          TSTB 0,PRI ;CHECK IF PRIORITY
4071 100102 100003          BPL 2$ ; IS AS SAME AS USER PGM
4072 100104 113705 075702          MOVB 0,UST,R5 ;PICK UP USER UST IF SO
4073 100110 000407          BR 3$
4074 100112 113705 075704 2$: MOVB 0,PRI,R5 ;OTHERWISE PICK UP ACTUAL PRIORITY
4075 100116 000257          CCC ;CLEAR CARRY
4076 100120 106005          RORB R5 ;SHIFT LOW ORDER BITS
4077 100122 106005          RORB R5 ; INTO
4078 100124 106005          RORB R5 ; HIGH ORDER
4079 100126 106005          RORB R5 ; POSITION
4080 100130 042705 177420 3$: BIC #0.TBT!177400,R5 ;CLEAR POSSIBLE T BIT (S/I MODE)
4081 100134 010546          MOV R5,-(SP) ;PUT THE STATUS AWAY WHERE IT BELONGS
4082 100136 004737 101202          JSR PC,0.WST
4083 100142 013705 075700          MOV 0,UPC,R5 ;GET PC, IT POINTS TO THE TRT
4084 100146 105737 075571          TSTB 0,S ;SEE IF IT WAS SINGLE INSTRUCTION FUN
4085 100152 100432          BMI 14$ ;IF SO HANDLE THERE
4086 100154 005745          TST -(R5)
4087 100156 010537 075700          MOV R5,0.UPC
4088 100162 012704 000016          MOV #0.BKP,R4 ;GET A COUNTER
4089 100166 020564 075706 11$: CMP R5,0.ADR1(R4) ;COMPARE WITH LIST
4090 100172 001426          BEQ 12$ ;JUMP IF FOUND
4091 100174 005304          DEC R4
4092 100176 005304          DEC R4
4093 100200 002372          BGE 11$ ;RE-LOOP UNTIL FOUND
4094 100202 004537 100500          JSR 5,0.SVTT ;SAVE TELETYPE STATUS
4095 100206 004537 101246          JSR 5,0.CRLF
4096 100212 012704 101366          MOV #0.BD,R4 ;ERROR, NOTHING FOUND
4097 100216 012703 101367          MOV #0.BD+1,R3
4098 100222 004537 101126          JSR 5,0.TYPE ;OUTPUT 'BE' FOR BAD ENTRY
4099 100226 010500          MOV R5,R0
4100 100230 062737 000002 075700 ADD #2,0.UPC ;POP OVER THE ADJUSTMENT ABOVE
4101 100236 000444          BR 13$ ; OR CONTINUE

```

4104	100240	112704	000020		14\$:	MOVB	#0,BKP+2,R4	:SET BREAK POINT HIGH + 1
4105	100244	010564	075706			MOV	R5,O.ADR1(R4)	:STORE NEXT PC VALUE FOR TYPE OUT
4106	100250	110437	075573		12\$:	MOVB	R4,O.P	:ALLOW PROCEED
4107	100254	005364	075730			DEC	O.CT(R4)	
4108	100260	003247				BGT	O.C1	:JUMP IF REPEAT
4109	100262	012764	000001	075730		MOV	#1,O.CT(R4)	:RESET COUNT TO 1
4110	100270	004537	100500			JSR	5,O.SVTT	:SAVE TELETYPE STATUS, R4 IS SAFE
4111	100274	012700	000102			MOV	#'B',R0	
4112	100300	004537	100730			JSR	5,O.FTYP	:TYPE 'B'
4113	100304	113700	075573			MOVB	O.P,R0	:CONVERT BREAKPOINT NUMBER TO ASCII
4114	100310	062700	000140			ADD	#140,R0	
4115	100314	006200				ASR	R0	
4116	100316	004537	100730			JSR	5,O.FTYP	
4117	100322	012700	000073			MOV	#',R0	
4118	100326	004537	100730			JSR	5,O.FTYP	: TYPE
4119	100332	012737	000002	075566		MOV	#2,O.BW	: SET WORD MODE
4120	100340	113704	075573			MOVB	O.P,R4	
4121	100344	016400	075706			MOV	O.ADR1(R4),R0	:GET ADDRESS OF BREAK
4122	100350	004537	101272		13\$:	JSR	5,O.RORA	: CHECK FORMAT
4123	100354	000137	076520			JMP	O.DCD	:GO TO DECODER

```

4126           ; SAVE REGISTERS R0-R6
4127           ;   INTERNAL STACK
4128
4129 100360 012637 075564 0.SVR: MOV (SP)+,0.XXX :PICK REGISTER FROM STACK AND SAVE
4130 100364 010637 075676      MOV SP,0.USP :SAVE USER STACK ADDRESS
4131 100370 012706 075676      MOV #0.USP,SP :SET TO INTERNAL STACK
4132 100374 010546      MOV R5,-(SP) :SAVE
4133 100376 010446      MOV R4,-(SP) :REGISTERS
4134 100400 010346      MOV R3,-(SP) :1
4135 100402 010246      MOV R2,-(SP) :THRU
4136 100404 010146      MOV R1,-(SP) :5
4137 100406 013746 075564      MOV 0.XXX,-(SP) :PUT SAVED REGISTER ON STACK
4138 100412 005746      TST -(SP)
4139 100414 000200      RTS R0
4140
4141           ; RESTORE REGISTERS R0-R6
4142
4143 100416 005726 0.SVR: TST (SP)+ :POP THE EXTRA CELL
4144 100420 012637 075564      MOV (SP)+,0.XXX :GET R0 FROM STACK
4145 100424 012601      MOV (SP)+,R1 :RESTORE
4146 100426 012602      MOV (SP)+,R2 :REGISTERS
4147 100430 012603      MOV (SP)+,R3 :1
4148 100432 012604      MOV (SP)+,R4 :THRU
4149 100434 012605      MOV (SP)+,R5 :5
4150 100436 013706 075676      MOV 0.USP,SP :RESTORE USER STACK
4151 100442 013746 075564      MOV 0.XXX,-(SP) :PUT R0 ON USER STACK
4152 100446 000200      RTS R0
    
```

```

4155          ; RESTORE BREAKPOINTS 0-7
4156
4157 100450 012704 000016      0.RSB: MOV    #0.BKP,R4          ;RESTORE ALL BREAKPOINTS
4158 100454 017464 075706 075752 1$:  MOV    @0.ADR1(R4),0.UIN(R4);SAVE CONTENTS
4159 100462 013774 101370 075706      MOV    0.TRTC,@0.ADR1(R4);REPLACE WITH TRAP
4160 100470 005304              DEC    R4
4161 100472 005304              DEC    R4
4162 100474 002367              BGE   1$          ;RE-LOOP UNTIL DONE
4163 100476 000205              RTS    R5          ; THEN QUIT
4164
4165          ; SAVE TELETYPE STATUS
4166
4167 100500 113737 177560 075574 0.SVTT: MOVB   0.RCSR,0.CSR1      ;SAVE R C/SR
4168 100506 113737 177564 075575      MOVB   0.TCSR,0.CSR2      ;SAVE T C/SR
4169 100514 105037 177560              CLRB   0.RCSR            ;CLEAR ENABLE AND MAINTENANCE
4170 100520 105037 177564              CLRB   0.TCSR            ; BITS IN BOTH C/SR
4171 100524 105737 177564      1$:  TSTB   0.TCSR            ;LOOP UNTIL READY BIT COMES ON
4172 100530 100375              BPL   1$              ;BR IF BIT NOT ON
4173 100532 000205              RTS    R5
4174
4175          ; RESTORE TELETYPE STATUS
4176 100534 004537 101246      0.RSTT: JSR    5,0.CRLF
4177 100540 105737 177564              TSTB   0.TCSR            ;WAIT READY
4178 100544 100375              BPL   -4              ; ON PRINTER
4179 100546 032737 004000 177560      BIT    #4000,0.RCSR      ;CHECK BUSY FLAG
4180 100554 001403              BEQ    1$              ;SKIP READY LOOP IF NOT BUSY
4181 100556 105737 177560              TSTB   0.RCSR            ;WAIT READY
4182 100562 100375              BPL   -4              ; ON READER
4183 100564 113737 075574 177560 1$:  MOVB   0.CSR1,0.RCSR      ;RESTORE
4184 100572 113737 075575 177564      MOVB   0.CSR2,0.TCSR      ; THE STATUS REGISTERS
4185 100600 000205              RTS    R5
4186
4187          ; REMOVE BREAKPOINTS 0-7
4188          ; IN THE OPPOSITE ORDER OF SETTING
4189
4190 100602 105737 075571      0.REM: TSTB   0.S          ;SEE IF SINGLE INSTRUCTION IS GOING
4191 100606 001011              BNE   2$              ;EXIT IF SO
4192 100610 005004              CLR    R4              ;REMOVE ALL BREAKPOINTS
4193 100612 016474 075752 075706 1$:  MOV    0.UIN(R4),@0.ADR1(R4) ;CLEAR BREAKPOINT
4194 100620 005204              INC    R4
4195 100622 005204              INC    R4
4196 100624 020427 000016      CMP    R4,#0.BKP
4197 100630 003770              BLE   1$              ;RE-LOOP UNTIL DONE
4198 100632 000205      2$:  RTS    R5          ;THEN QUIT
    
```

```

4201          : TYPE OUT CONTENTS OF WORD OR BYTE WITH ONE TRAILING SPACE
4202          : WORD IS IN R0
4203
4204 100634 012703 000006      0.CADV: MOV      #6,R3          :# OF DIGITS
4205 100640 012704 177776      MOV      #-2,R4         :# OF BITS FIRST-3
4206 100644 022737 000001 075566  CMP      #1,0.BW       :SEE IF WORD MODE
4207 100652 001004              BNE      3$            :BRANCH IF SO
4208 100654 162703 000003      SUB      #3,R3         :ONLY DO 3 DIGITS
4209 100660 005204              INC      R4            :DO 2 BITS FIRST
4210 100662 000300              SWAB    R0             :AND TURN R0 AROUND
4211 100664 010046              3$: MOV     R0,-(SP)    :SAVE R0
4212 100666 062704 000003      2$: ADD     #3,R4       :COMPUTE THE NUMBER OF BITS TO DO
4213 100672 005000              CLR     R0
4214 100674 006116              1$: ROL    (SP)        :GET A BIT
4215 100676 006100              ROL    R0              :STORE IT AWAY
4216 100700 005304              DEC    R4              :DECREMENT COUNTER
4217 100702 003374              BGT    1$              :LOOP IF MORE BITS NEEDED
4218 100704 062700 000060      ADD     #'0,R0         :CONVERT TO ASCII
4219 100710 004537 100730      JSR    R5,0.FTYP      :TYPE IT
4220 100714 005303              DEC    R3              :SEE IF MORE DIGITS TO DO
4221 100716 003363              BGT    2$              :LOOP IF SO
4222 100720 112700 000040      MOVB   #' ,R0         :SET UP FOR TRAILING SPACE
4223 100724 005726              TST   (SP)+           :GET RID OF JUNK
4224 100726 000400              BR    0.FTYP
    
```

```

4225
4226
4227          : TYPE ONLY ONE CHARACTER (CONTAINED IN R0)
4228
4229 100730 105737 177564      0.FTYP: TSTB   0.TCSR
4230 100734 100375              BPL    0.FTYP
4231 100736 042700 177400      BIC    #177400,R0     :CLEAR HIGH BYTE,SHOULD NOT
4232                                     :CONTAIN INPORTANT INFO.
4233 100742 001416              BEQ    3$
4234 100744 022700 000176      CMP    #176,R0        :PRINT ? FOR 177:16-37
4235 100750 103411              BLO   2$              : 1-10 AND 200-377.
4236 100752 022700 000037      CMP    #37,R0
4237 100756 103410              BLO   3$
4238 100760 022700 000015      CMP    #15,R0
4239 100764 103403              BLO   2$
4240 100766 022700 000010      CMP    #10,R0
4241 100772 103402              BLO   3$
4242 100774 012700 000077      2$: MOV     #'?,R0
4243 101000 110037 177566      3$: MOVB   R0,0.TDB
4244 101004 000205      0.TYP1: RTS    R5
    
```

```

4247      ; GENERAL CHARACTER INPUT ROUTINE -- ODT11X
4248      ; CHARACTER INPUT GOES TO R0
4249
4250 101006 105737 177560      0.GET:  TSTB  O.RCSR      ;WAIT FOR
4251 101012 100375              BPL  O.GET          ; INPUT FROM KBD
4252 101014 113700 177562      MOVB  O.RDB,R0      ;GET CHARACTER - STRIP OFF PARITY
4253 101020 042700 177600      BIC   #177600,R0   ;STRIP OFF PARITY FROM CHARACTER
4254 101024 122700 000003      CMPB  #3,R0        ;IS IT ^C?
4255 101030 001435              BEQ   3$            ;IF SO, DO NOT ECHO
4256 101032 122700 000006      CMPB  #6,R0        ;IS IT ^F?
4257 101036 001432              BEQ   3$            ;IF SO, DO NOT ECHO
4258 101040 122700 000005      CMPB  #5,R0        ;IS IT ^E?
4259 101044 001427              BEQ   3$            ;IF SO, DO NOT ECHO
4260 101046 105737 075602      TSTB  O.SCRN       ;SHOULD WE ECHO <LF>?
4261 101052 001003              BNE   2$            ;BR IF YES
4262 101054 120027 000012      CMPB  R0,#012     ;SEE IF A <LF>
4263 101060 001421              BEQ   3$            ;IF SO SAVE THE PAPER
4264 101062 120027 000173      2$:  CMPB  R0,#173
4265 101066 002005              BGE   1$
4266 101070 122700 000140      CMPB  #140,R0     ;TEST FOR LOWER CASE
4267 101074 002002              BGE   1$
4268 101076 162700 000040      SUB   #40,R0      ;CHAR IS LC-CONVERT TO UPPER CASE
4269 101102 004537 100730      1$:  JSR   5.O.FTYP  ;ECHO CHARACTER
4270 101106 001737              BEQ   0.GET        ;IGNORE NULLS
4271 101110 105737 075602      TSTB  O.SCRN       ;SHOULD WE PASS ON SPACES?
4272 101114 001003              BNE   3$            ;BR IF YES
4273 101116 122700 000040      CMPB  #40,R0      ;CHECK FOR SPACES
4274 101122 001731              BEQ   0.GET        ;IGNORE SPACES
4275 101124 000205      3$:  RTS   R5
    
```

```

4278 ; GENERAL CHARACTER OUTPUT ROUTINE - ODT11X
4279 ; ADDRESS OF FIRST BYTE IN R4,
4280 ; ADDRESS OF LAST BYTE IN R3, (R3)>(R4)
4281 ; EXPECTS LOCS 56,57 TO BE INITIALIZED BY MONITOR FOR FILL
4282 ; CHARACTERISTICS OF TERMINAL
4283 ; 56=CHAR TO BE FILLED AFTER
4284 ; 57=# OF NULLS TO FILL WITH
4285
4286 101126 020304 O.TYPE: CMP R3,R4 ;CHECK FOR COMPLETION
4287 101130 103725 BLO O.TYP1 ; EXIT WHEN DONE
4288 101132 112400 MOVB (R4)+,R0 ;GET A CHARACTER
4289 101134 004537 100730 JSR 5,O.FTYP ;TYPE ONE CHARACTER
4290 101140 120037 003002 CMPB R0,@#$FILLC ;COMPARE CHAR AGAINST FILL REQUIREMENT
4291 101144 001370 BNE O.TYPE ;NO FILL NEEDED
4292 101146 113737 002451 075576 MOVB @#$FILLS,O.FIL ;FILL COUNT INTO TEMP
4293 101154 005000 CLR R0 ;FILL WITH NULLS
4294 101156 004537 100730 2$: JSR R5,O.FTYP ;TYPE NULLS
4295 101162 105337 075576 DECB O.FIL ;DECREASE COUNT
4296 101166 003373 BGT 2$ ;BRANCH IF NOT DONE
4297 101170 000756 BR O.TYPE ;LOOP UNTIL DONE
    
```

```

4300                :SUBROUTINE TO READ THE PS INDEPENDENT OF MACHINE
4301
4302 101172 013737 177776 075702 O.RRST: MOV     ST,O.UST      :STORE THE STATUS IN USER
4303                                     :STATUS AREA.FOR AN LSI
4304                                     :THIS IS CHANGED TO
4305                                     :MFPS O.UST
4306 101200 000207                RTS     PC
4307
4308                :SUBROUTINE TO WRITE PS INDEPENDENT OF MACHINE
4309                :CALL ROUTINE WITH PS VALUE
4310                :ON THE STACK
4311
4312 101202 016637 000002 177776 O.WST:  MOV     2(SP),ST      :STORE NEW PS VALUE
4313                                     :THIS INSTRUCTION IS CHANGED FOR LSI
4314                                     :TO MTPS 2(SP)
4315 101210 012616                MOV     (SP)+,(SP)    :PUT RETURN PC OVER SUB. ARGUMENT
4316 101212 000207                RTS     PC                  :TO RETURN WITHOUT IT ON THE STACK
4317                : CLOSE WORD OR BYTE AND EXIT,
4318                : UPON ENTERING, R2 HAS NUMERIC FLAG, R4 HAS CONTENTS
4319
4320                .ENABL  LSB
4321 101214 005702 O.CLSE: TST     R2                          :IF NO NUMBER WAS TYPED THERE IS
4322 101216 001412                BEQ     1$                          :NO CHANGE TO THE OPEN CELL
4323 101220 022737 000001 075566    CMP     #1,O.BW
4324 101226 001404                BEQ     2$                          :JUMP IF BYTE MODE
4325 101230 101005                BHI     1$                          :JUMP IF ALREADY CLOSED
4326 101232 010477 174322                MOV     R4,@O.CAD      :STORE WORD
4327 101236 000402                BR      1$
4328 101240 110477 174314 2$:      MOVB    R4,@O.CAD      :STORE BYTE
4329 101244 000207 1$:      RTS     PC
4330 101246 012703 101317 O.CRLF: MOV     #O.CR+1,R3      :LWA <CR,LF>
4331 101252 000402                BR      3$
4332 101254 012703 101320 O.CRLS: MOV     #O.CR+2,R3      :LWA <CR,LF>*
4333 101260 012704 101316 3$:      MOV     #O.CR,R4        :FWA
4334 101264 004537 101126                JSR     5,O.TYPE      :TYPE SOMETHING
4335 101270 000205                RTS     R5
4336                .DSABL  LSB
    
```

```

4339      ;SUBROUTINE O.RORA
4340      ;FUNCTION:
4341      ;ADDRESS WILL BE PRINTED IN ABSOLUTE FORM
4342      ;INPUT: THE ADDRESS TO BE PRINTED IS IN R0.
4343      ;DATA SAVED: R0 CONTAINING THE ADDRESS TO BE
4344      ;PRINTED, AND LOCATION O.CAD CONTAINING
4345      ;THE CURRENT ADDRESS WERE SAVED AND RESTORED.
4346      ;CALLED: JSR 5,O.RORA
4347
4348 101272 004537 100634 O.RORA: JSR 5,O.CADV ;PRINT ABSOLUTE ADDRESS
4349 101276 000205      RTS R5
4350 101300      012 O.ID: .BYTE 012
4351 101301      015      .BYTE 015
4352 101302      040      .BYTE 40
4353 101303      117      .ASCII /ODT (CEMKA)/
4354      101306      040      104      124
4355      101311      105      050      103
4356      101314      101      115      113
4357      101315      051
4358
4359      O.IDND=-1
4360 O.CR: .BYTE 015 ; <CR>
4361      .BYTE 012 ; <LF>
4362      .BYTE '*' ; *
4363
4364 O.LGCH: .BYTE ':' ;
4365      .BYTE '/' ; /
4366      .BYTE '\ ' ; \ (BACK SLASH)
4367      .BYTE 015 ; CARRIAGE RETURN
4368      .BYTE '$' ; $
4369      .BYTE 'G' ; G
4370      .BYTE 012 ; <LF>
4371      .BYTE '~' ; ~ (BACK ARROW)
4372      .BYTE '^' ; ^ (UP ARROW)
4373      .BYTE 'O' ; O
4374      .BYTE 'B' ; B
4375      .BYTE 'P' ; P
4376      .BYTE '@' ; @
4377      .BYTE '>' ; >
4378      .BYTE 'S' ; S
4379      .BYTE '-' ; -
4380      .BYTE 003 ; CTRL C
4381      .BYTE 006 ; CTRL F
4382      .BYTE 005 ; CTRL E
4383 O.CLGT = .-O.LGCH ;TABLE LENGTH
4384
4385 O.TL: .BYTE 'S' ;DO 1
4386      .BYTE 'P' ;NOT 2
4387      .BYTE 'M' ;CHANGE 3
4388      .BYTE 0 ;THE 4
4389      .BYTE 0 ;ORDER 5
4390      .BYTE 'C' ; 6
4391      .BYTE 'F' ; 7
4392      .BYTE 'R' ; 10
4393      .BYTE 0 ; 11
4394      .BYTE 0 ; 12
4395      .BYTE 0 ; 13
4396      .BYTE 0 ; 14
    
```

4393 101361 000
4394 101362 000
4395 101363 000
4396 101364 102
4397 000020
4398
4399 101366 042502
4400 101370 000003

.BYTE 0 : 15
.BYTE 0 : 16
.BYTE 0 : 17
.BYTE 'B : 20
O.LG = -.O.TL
O.BD: .EVEN
O.TRFC: TRT .WORD 'BE

;TRACE TRAP PROTOTYPE

4403
 4404
 4405
 4406
 4407
 4408
 4409
 4410
 4411
 4412
 4413
 4414
 4415
 4416
 4417 101372
 4418 101372 104530
 4419 101374 110307
 4420 101376 102504
 4421 101400 103103
 4422
 4423 101402 103220
 4424 101404 107366
 4425 101406 102252
 4426 101410 102743
 4427
 4428 101412 103256
 4429 101414 107446
 4430 101416 102302
 4431 101420 103071
 4432
 4433 101422 103311
 4434 101424 107446
 4435 101426 102312
 4436 101430 103071
 4437
 4438 101432 103357
 4439 101434 107507
 4440 101436 102322
 4441 101440 102743
 4442
 4443 101442 103434
 4444 101444 107507
 4445 101446 102322
 4446 101450 102743
 4447
 4448 101452 103461
 4449 101454 107507
 4450 101456 102322
 4451 101460 102743
 4452
 4453 101462 103633
 4454 101464 107366
 4455 101466 102252
 4456 101470 102743

.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

\$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
 EM10
 DH1
 DT1
 DF2

TABLE ERROR POINTER		
4459		:ERROR 11
4460	101472	EM11
4461	101474	DH7
4462	101476	DT7
4463	101500	DF3
4464		:ERROR 12
4465	101502	EM11
4466	101504	DH7
4467	101506	DT7
4468	101510	DF4
4469		:ERROR 13
4470	101512	EM12
4471	101514	DH10
4472	101516	DT10
4473	101520	DF2
4474		:ERROR 14
4475	101522	EM11
4476	101524	DH7
4477	101526	DT7
4478	101530	DF5
4479		:ERROR 15
4480	101532	EM11
4481	101534	DH7
4482	101536	DT7
4483	101540	DF6
4484		:ERROR 16
4485	101542	EM13
4486	101544	DH13
4487	101546	DT13
4488	101550	DF11
4489		:ERROR 17
4490	101552	EM14
4491	101554	DH13
4492	101556	DT13
4493	101560	DF11
4494		:ERROR 20
4495	101562	EM15
4496	101564	DH13
4497	101566	DT13
4498	101570	DF11
4499		:ERROR 21
4500	101572	EM16
4501	101574	DH11
4502	101576	DT11
4503	101600	DF2
4504		:ERROR 22
4505	101602	EM17
4506	101604	DH7
4507	101606	DT7
4508	101610	DF5
4509		:ERROR 23
4510	101612	EM18
4511	101614	DH8
4512	101616	DT8
4513	101620	DF10

4516			:ERROR	24	
4517	101622	104256	EM19		
4518	101624	110307	DH13		
4519	101626	102504	DT13		
4520	101630	103103	DF11		
4521			:ERROR	25	
4522	101632	104333	EM20		
4523	101634	110307	DH13		
4524	101636	102504	DT13		
4525	101640	103103	DF11		
4526			:ERROR	26	
4527	101642	000000	0		:NO MESSAGE
4528	101644	110272	DH12		
4529	101646	102476	DT12		
4530	101650	102743	DF2		
4531			:ERROR	27	
4532	101652	104415	EM21		
4533	101654	110243	DH11		
4534	101656	102466	DT11		
4535	101660	102743	DF2		
4536			:ERROR	30	
4537	101662	104454	EM22		
4538	101664	110307	DH13		
4539	101666	102504	DT13		
4540	101670	103103	DF11		
4541			:ERROR	31	
4542	101672	000000	0		:NO MESSAGE
4543	101674	110415	DH14		
4544	101676	102530	DT14		
4545	101700	102743	DF2		
4546			:ERROR	32	
4547	101702	104501	EM23		
4548	101704	107507	DH5		
4549	101706	102322	DT5		
4550	101710	102743	DF2		
4551			:ERROR	33	
4552	101712	106025	EM38		
4553	101714	110672	DH17		
4554	101716	102264	DT2		
4555	101720	103116	DF13		
4556			:ERROR	34	
4557	101722	104607	EM25		
4558	101724	110533	DH15		
4559	101726	102556	DT16		
4560	101730	103053	DF7		
4561			:ERROR	35	
4562	101732	104634	EM26		
4563	101734	110655	DH16		
4564	101736	102610	DT17		
4565	101740	102767	DF3		
4566			:ERROR	36	
4567	101742	103521	EM8		
4568	101744	110672	DH17		
4569	101746	102616	DT18		
4570	101750	103116	DF13		

4573			:ERROR	37
4574	101752	103556	EM9	
4575	101754	110106	DH9	
4576	101756	102432	DT9	
4577	101760	102743	DF2	
4578			:ERROR	40
4579	101762	104705	EM27	
4580	101764	110655	DH16	
4581	101766	102610	DT17	
4582	101770	103067	DF8	
4583			:ERROR	41
4584	101772	104774	EM28	
4585	101774	110672	DH17	
4586	101776	102634	DT19	
4587	102000	103116	DF13	
4588			:ERROR	42
4589	102002	105624	EM35	
4590	102004	107664	DH7	
4591	102006	102362	DT7	
4592	102010	102767	DF3	
4593			:ERROR	43
4594	102012	105027	EM29	
4595	102014	107664	DH7	
4596	102016	102362	DT7	
4597	102020	102767	DF3	
4598			:ERROR	44
4599	102022	105111	EM30	
4600	102024	107664	DH7	
4601	102026	102362	DT7	
4602	102030	103021	DF5	
4603			:ERROR	45
4604	102032	105230	EM31	
4605	102034	107664	DH7	
4606	102036	102362	DT7	
4607	102040	102767	DF3	
4608			:ERROR	46
4609	102042	105330	EM32	
4610	102044	107664	DH7	
4611	102046	102362	DT7	
4612	102050	102767	DF3	
4613			:ERROR	47
4614	102052	105435	EM33	
4615	102054	107664	DH7	
4616	102056	102362	DT7	
4617	102060	102767	DF3	
4618			:ERROR	50
4619	102062	105543	EM34	
4620	102064	111023	DH19	
4621	102066	102700	DT22	
4622	102070	103114	DF12	
4623			:ERROR	51
4624	102072	105711	EM36	
4625	102074	107615	DH6	
4626	102076	102346	DT6	
4627	102100	102743	DF2	

TABLE ERROR POINTER

4630			:ERROR	52
4631	102102	105760	EM37	
4632	102104	110672	DH17	
4633	102106	102616	DT18	
4634	102110	103116	DF13	
4635			:ERROR	53
4636	102112	106133	EM39	
4637	102114	110672	DH17	
4638	102116	102616	DT18	
4639	102120	103116	DF13	
4640			:ERROR	54
4641	102122	106220	EM40	
4642	102124	107423	DH2	
4643	102126	102652	DT20	
4644	102130	102743	DF2	
4645			:ERROR	55
4646	102132	106272	EM41	
4647	102134	110746	DH18	
4648	102136	102662	DT21	
4649	102140	103103	DF11	
4650			:ERROR	56
4651	102142	106374	EM42	
4652	102144	110746	DH18	
4653	102146	102662	DT21	
4654	102150	103103	DF11	
4655			:ERROR	57
4656	102152	103134	EM1	
4657	102154	000000	0	
4658	102156	000000	0	
4659	102160	000000	0	
4660			:ERROR	60
4661	102162	106477	EM43	
4662	102164	111037	DH20	
4663	102166	102706	DT23	
4664	102170	102743	DF2	
4665			:ERROR	61
4666	102172	106575	EM44	
4667	102174	111056	DH21	
4668	102176	102714	DT24	
4669	102200	102743	DF2	
4670			:ERROR	62
4671	102202	106672	EM45	
4672	102204	111037	DH20	
4673	102206	102706	DT23	
4674	102210	102743	DF2	
4675			:ERROR	63
4676	102212	106773	EM46	
4677	102214	110672	DH17	
4678	102216	102264	DT2	
4679	102220	103116	DF13	
4680			:ERROR	64
4681	102222	107062	EM47	
4682	102224	110655	DH16	
4683	102226	102610	DT17	
4684	102230	102767	DF3	

4687			:ERROR 65
4688	102232	107170	EM48
4689	102234	111063	DH22
4690	102236	102652	DT20
4691	102240	102743	DF2
4692			:ERROR 66
4693	102242	107267	EM49
4694	102244	111111	DH23
4695	102246	102720	DT25
4696	102250	103125	DF14

Address	DT	SBTTL	ERRPC	ADDRESS	GOOD	BAD	0
4699		.SBTTL	ERROR DATA TAGS (DT)				
4700	DT1:	.WORD	ERRPC	ADDRESS	GOOD	BAD	0
102252							
102260							
4701	DT2:	.WORD	ERRPC	DUMMY	SLFLAG	SLFLAG+2	SLFLAG+4
102264							
102272							
102300							
4702	DT3:	.WORD	ERRPC	PADDRESS	PARCNT		0
102302							
102310							
4703	DT4:	.WORD	ERRPC	ADDRESS	NEMCNT		0
102312							
102320							
4704	DT5:	.WORD	ERRPC	CPUERR	MEMERR	HIADRS	LOADRS
102322							
102330							
102336							
102344							
4705	DT6:	.WORD	ERRPC	APTCORE	MJSIZE	APTMS	MKSIZE
102346							
102354							
4706	DT7:	.WORD	ERRPC	DUMMY	ADDRESS	DUMMY	GOOD
102362							
102370							
102376							
4707		.WORD	DUMMY	DUMMY	DUMMY	DUMMY	DUMMY
102400							
102406							
102414							
4708	DT8:	.WORD	ADDRESS	GOOD	BAD	GDSWITCH	BDSWITCH
102416							
102424							
4709	DT9:	.WORD	ERRPC	CPUERR	MEMERR	CSRNO	0
102432							
102440							
4710	DT10:	.WORD	DETRO	DETR1	DETR2	DETR3	DETR4
102444							
102452							
102460							
4711	DT11:	.WORD	ERRPC	CSR	CSR+2	0	
102466							
102474							
4712	DT12:	.WORD	CSR	CSR+2	0		
102476							
4713	DT13:	.WORD	ERRPC	DUMMY	ADDRESS	DUMMY	TSTDAT
102504							
102512							
102520							
102526							
4714	DT14:	.WORD	MEMERR	MAINT	CONTRL	HIADRS	LOADRS
102530							
102536							
102544							
102552							
4715	DT16:	.WORD	ERRPC	DUMMY	DUMMY	GOOD	GOOD2
102556							
102564							
4716		.WORD	BAD	BAD2	BAD3	DUMMY	DUMMY
102572							
102600							
102606							
4717	DT17:	.WORD	ERRPC	DUMMY	0		
102610							
4718	DT18:	.WORD	ERRPC	DUMMY	PORTCOUNT	PORTCOUNT+2	PORTCOUNT+4
102616							
102624							
102632							
4719	DT19:	.WORD	ERRPC	DUMMY	PORTDIR	PORTDIR+2	PORTDIR+4
102634							
102642							
102650							
4720	DT20:	.WORD	ERRPC	GOOD	BAD	0	
102652							
102660							
4721	DT21:	.WORD	ERRPC	DUMMY	ADDRESS	DUMMY	GOOD
102662							
102670							
102676							

4722	102700	002024	002304	000000	DT22:	.WORD	ERRPC,DUMMY,0
4723	102706	002024	002056	000000	DT23:	.WORD	ERRPC,BAD,0
4724	102714	002024	000000		DT24:	.WORD	ERRPC,0
4725	102720	002024	002304	002646	DT25:	.WORD	ERRPC,DUMMY,PORTCOUNT,PORTCOUNT+2,PORTCOUNT+4,PORTCOUNT+6,BAD,0
	102726	002650	002652	002654			
	102734	002056	000000				

4746					
4752	103134	125	116	105	EM1: .SBITL ERROR MESSAGES (EM)
	103137	130	120	105	.ASCIZ /UNEXPECTED TRAP TO 170 (IIST) - PROGRAM CONTINUING!/ .ASCIZ
	103142	103	124	105	
	103145	104	040	124	
	103150	122	101	120	
	103153	040	124	117	
	103156	040	061	067	
	103161	060	040	050	
	103164	111	111	123	
	103167	124	051	040	
	103172	055	040	120	
	103175	122	117	107	
	103200	122	101	115	
	103203	040	103	117	
	103206	116	124	111	
	103211	116	125	111	
	103214	116	107	041	
	103217	000			
4753	103220	103	101	116	EM2: .ASCIZ /CAN'T SET 22 BIT MODE IN MMR3/
	103223	047	124	040	
	103226	123	105	124	
	103231	040	062	062	
	103234	040	102	111	
	103237	124	040	115	
	103242	117	104	105	
	103245	040	111	116	
	103250	040	115	115	
	103253	122	063	000	
4754	103256	120	101	122	EM3: .ASCIZ /PARITY ERROR(S) IN BANK 0/
	103261	111	124	131	
	103264	040	105	122	
	103267	122	117	122	
	103272	050	123	051	
	103275	040	111	116	
	103300	040	102	101	
	103303	116	113	040	
	103306	060	051	000	
4755	103311	116	117	116	EM4: .ASCIZ /NON-EXISTANT MEMORY (HOLES) IN BANK 0/
	103314	055	105	130	
	103317	111	123	124	
	103322	101	116	124	
	103325	040	115	105	
	103330	115	117	122	
	103333	131	040	050	
	103336	110	117	114	
	103341	105	123	051	
	103344	040	111	116	
	103347	040	102	101	
	103352	116	113	040	
	103355	060	000		
4756	103357	111	114	114	EM5: .ASCIZ /ILLEGAL OR RESERVED INSTRUCTION (TRAP TO 10)/
	103362	105	107	101	
	103365	114	040	117	
	103370	122	040	122	
	103373	105	123	105	
	103376	122	126	105	

	103401	104	040	111	
	103404	116	123	124	
	103407	122	125	103	
	103412	124	111	117	
	103415	116	040	050	
	103420	124	122	101	
	103423	120	040	124	
	103426	117	040	061	
	103431	060	051	000	
4757	103434	125	116	105	EM6: .ASCIZ /UNEXPECTED TRAP TO 4/
	103437	130	120	105	
	103442	103	124	105	
	103445	104	040	124	
	103450	122	101	120	
	103453	040	124	117	
	103456	040	064	000	
4758	103461	115	105	115	EM7: .ASCIZ /MEMORY MANAGEMENT (TRAP TO 250)/
	103464	117	122	131	
	103467	040	115	101	
	103472	116	101	107	
	103475	105	115	105	
	103500	116	124	040	
	103503	050	124	122	
	103506	101	120	040	
	103511	124	117	040	
	103514	062	065	060	
	103517	051	000		
4759	103521	120	117	122	EM8: .ASCIZ /PORT CONTENTION NOT BALANCED/
	103524	124	040	103	
	103527	117	116	124	
	103532	105	116	124	
	103535	111	117	116	
	103540	040	116	117	
	103543	124	040	102	
	103546	101	114	101	
	103551	116	103	105	
	103554	104	000		
4760					
4761	103556	116	117	116	EM9: .ASCIZ /NON-EXISTANT CSR CAUSED INCORRECT ERROR TRAP/
	103561	055	105	130	
	103564	111	123	124	
	103567	101	116	124	
	103572	040	103	123	
	103575	122	040	103	
	103600	101	125	123	
	103603	105	104	040	
	103606	111	116	103	
	103611	117	122	122	
	103614	105	103	124	
	103617	040	105	122	
	103622	122	117	122	
	103625	040	124	122	
	103630	101	120	000	
4762	103633	111	116	103	EM10: .ASCIZ /INCORRECT DATA FROM DEVICE REGISTER/
	103636	117	122	122	
	103641	105	103	124	
	103644	040	104	101	

	103647	124	101	040	
	103652	106	122	117	
	103655	115	040	104	
	103660	105	126	111	
	103663	103	105	040	
	103666	122	105	107	
	103671	111	123	124	
	103674	105	122	000	
4763	103677	115	105	115	EM11: .ASCIZ /MEMORY DATA ERROR/
	103702	117	122	131	
	103705	040	104	101	
	103710	124	101	040	
	103713	105	122	122	
	103716	117	122	000	
4764	103721	104	105	124	EM12: .ASCIZ /DETAILED ERROR DUMP/
	103724	101	111	114	
	103727	105	104	040	
	103732	105	122	122	
	103735	117	122	040	
	103740	104	125	115	
	103743	120	000		
4765	103745	115	111	123	EM13: .ASCIZ /MISSING EXPECTED SBE FLAG/
	103750	123	111	116	
	103753	107	040	105	
	103756	130	120	105	
	103761	103	124	105	
	103764	104	040	123	
	103767	102	105	040	
	103772	106	114	101	
	103775	107	000		
4766	103777	127	122	111	EM14: .ASCIZ /WRITE BYTE FAILED TO CLEAR SBE FLAG/
	104002	124	105	040	
	104005	102	131	124	
	104010	105	040	106	
	104013	101	111	114	
	104016	105	104	040	
	104021	124	117	040	
	104024	103	114	105	
	104027	101	122	040	
	104032	123	102	105	
	104035	040	106	114	
	104040	101	107	000	
4767	104043	106	101	111	EM15: .ASCIZ /FAILED TO GET INTERRUPT WITH DBE FLAG/
	104046	114	105	104	
	104051	040	124	117	
	104054	040	107	105	
	104057	124	040	111	
	104062	116	124	105	
	104065	122	122	125	
	104070	120	124	040	
	104073	127	111	124	
	104076	110	040	104	
	104101	102	105	040	
	104104	106	114	101	
	104107	107	000		
4768	104111	117	116	114	EM16: .ASCIZ /ONLY ONE DBE FLAG SET/
	104114	131	040	117	

	104117	116	105	040	
	104122	104	102	105	
	104125	040	106	114	
	104130	101	107	040	
	104133	123	105	124	
	104136	000			
4769	104137	115	105	115	EM17: .ASCIZ /MEMORY DATA ERROR IN CHECK BITS/
	104142	117	122	131	
	104145	040	104	101	
	104150	124	101	040	
	104153	105	122	122	
	104156	117	122	040	
	104161	111	116	040	
	104164	103	110	105	
	104167	103	113	040	
	104172	102	111	124	
	104175	123	000		
4770	104177	123	131	123	EM18: .ASCIZ /SYSTEM SIZE REGISTER LOW DOES NOT MATCH MEMORY/
	104202	124	105	115	
	104205	040	123	111	
	104210	132	105	040	
	104213	122	105	107	
	104216	111	123	124	
	104221	105	122	040	
	104224	114	117	127	
	104227	040	104	117	
	104232	105	123	040	
	104235	116	117	124	
	104240	040	115	101	
	104243	124	103	110	
	104246	040	115	105	
	104251	115	117	122	
	104254	131	000		
4771	104256	123	102	105	EM19: .ASCIZ /SBE OR DBE CAUSED PARITY TRAP WHEN INHIBITED/
	104261	040	117	122	
	104264	040	104	102	
	104267	105	040	103	
	104272	101	125	123	
	104275	105	104	040	
	104300	120	101	122	
	104303	111	124	131	
	104306	040	124	122	
	104311	101	120	040	
	104314	127	110	105	
	104317	116	040	111	
	104322	116	110	111	
	104325	102	111	124	
	104330	105	104	000	
4772	104333	123	102	105	EM20: .ASCIZ /SBE OR DBE DID NOT CAUSE PARITY TRAP WHEN ENABLED/
	104336	040	117	122	
	104341	040	104	102	
	104344	105	040	104	
	104347	111	104	040	
	104352	116	117	124	
	104355	040	103	101	
	104360	125	123	105	
	104363	040	120	101	

	104366	122	111	124	
	104371	131	040	124	
	104374	122	101	120	
	104377	040	127	110	
	104402	105	116	040	
	104405	105	116	101	
	104410	102	114	105	
	104413	104	000		
4773	104415	123	102	105	EM21: .ASCIZ /SBE OR DBE ON MASTER TEST WORD/
	104420	040	117	122	
	104423	040	104	102	
	104426	105	040	117	
	104431	116	040	115	
	104434	101	123	124	
	104437	105	122	040	
	104442	124	105	123	
	104445	124	040	127	
	104450	117	122	104	
	104453	000			
4774	104454	115	111	123	EM22: .ASCIZ /MISSING EXPECTED DBE/
	104457	123	111	116	
	104462	107	040	105	
	104465	130	120	105	
	104470	103	124	105	
	104473	104	040	104	
	104476	102	105	000	
4775	104501	125	116	105	EM23: .ASCIZ /UNEXPECTED PARITY TRAP/
	104504	130	120	105	
	104507	103	124	105	
	104512	104	040	120	
	104515	101	122	111	
	104520	124	131	040	
	104523	124	122	101	
	104526	120	000		
4776	104530	122	105	103	EM24: .ASCIZ /RECEIVED DBE FLAG WHEN EXPECTING ONLY SBE FLAG/
	104533	105	111	126	
	104536	105	104	040	
	104541	104	102	105	
	104544	040	106	114	
	104547	101	107	040	
	104552	127	110	105	
	104555	116	040	105	
	104560	130	120	105	
	104563	103	124	111	
	104566	116	107	040	
	104571	117	116	114	
	104574	131	040	123	
	104577	102	105	040	
	104602	106	114	101	
	104605	107	000		
4777	104607	103	110	105	EM25: .ASCIZ /CHECK BIT DATA ERROR/
	104612	103	113	040	
	104615	102	111	124	
	104620	040	104	101	
	104623	124	101	040	
	104626	105	122	122	
	104631	117	122	000	

4778 104634 101 104 104 EM26: .ASCIZ /ADDRESS PARITY ERROR DID NOT CAUSE ABORT/
104637 122 105 123
104642 123 040 120
104645 101 122 111
104650 124 131 040
104653 105 122 122
104656 117 122 040
104661 104 111 104
104664 040 116 117
104667 124 040 103
104672 101 125 123
104675 105 040 101
104700 102 117 122

4779 104703 124 000
104705 115 113 061 EM27: .ASCIZ /MK11 PROTECTION POINTER FAILURE - DID NOT PROTECT BANK/
104710 061 040 120
104713 122 117 124
104716 105 103 124
104721 111 117 116
104724 040 120 117
104727 111 116 124
104732 105 122 040
104735 106 101 111
104740 114 125 122
104743 105 040 055
104746 040 104 111
104751 104 040 116
104754 117 124 040
104757 120 122 117
104762 124 105 103
104765 124 040 102
104770 101 116 113

4780 104773 000
104774 103 120 125 EM28: .ASCIZ /CPU IDENTIFICATION FAILURE/
104777 040 111 104
105002 105 116 124
105005 111 106 111
105010 103 101 124
105013 111 117 116
105016 040 106 101
105021 111 114 125

4781 105024 122 105 000
105027 103 117 122 EM29: .ASCIZ /CORRECTION FAILURE WITH ECC ENABLED ON FORCED SBE/
105032 122 105 103
105035 124 111 117
105040 116 040 106
105043 101 111 114
105046 125 122 105
105051 040 127 111
105054 124 110 040
105057 105 103 103
105062 040 105 116
105065 101 102 114
105070 105 104 040
105073 117 116 040
105076 106 117 122
105101 103 105 104

	105104	040	123	102	
	105107	105	000		
4782	105111	127	122	111	EM30: .ASCII /WRITE BYTE (MOVB) WITH ECC ENABLED FAILED TO CLEAR DATA AT/<CRLF>
	105114	124	105	040	
	105117	102	131	124	
	105122	105	040	050	
	105125	115	117	126	
	105130	102	051	040	
	105133	127	111	124	
	105136	110	040	105	
	105141	103	103	040	
	105144	105	116	101	
	105147	102	114	105	
	105152	104	040	106	
	105155	101	111	114	
	105160	105	104	040	
	105163	124	117	040	
	105166	103	114	105	
	105171	101	122	040	
	105174	104	101	124	
	105177	101	040	101	
	105202	124	200		
4783	105204	106	117	122	.ASCIZ /FORCED SBE LOCATION/
	105207	103	105	104	
	105212	040	123	102	
	105215	105	040	114	
	105220	117	103	101	
	105223	124	111	117	
	105226	116	000		
4784	105230	101	123	122	EM31: .ASCIZ /ASRB (R3)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	105233	102	040	050	
	105236	122	063	051	
	105241	053	040	127	
	105244	111	124	110	
	105247	040	105	103	
	105252	103	040	105	
	105255	116	101	102	
	105260	114	105	104	
	105263	040	103	110	
	105266	101	116	107	
	105271	105	104	040	
	105274	104	101	124	
	105277	101	040	101	
	105302	124	040	106	
	105305	117	122	103	
	105310	105	104	040	
	105313	104	102	105	
	105316	040	114	117	
	105321	103	101	124	
	105324	111	117	116	
	105327	000			
4785	105330	115	117	126	EM32: .ASCIZ /MOVB #360,(R2)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	105333	102	040	043	
	105336	063	066	060	
	105341	054	050	122	
	105344	062	051	053	
	105347	040	127	111	

	105352	124	110	040
	105355	105	103	103
	105360	040	105	116
	105363	101	102	114
	105366	105	104	040
	105371	103	110	101
	105374	116	107	105
	105377	104	040	104
	105402	101	124	101
	105405	040	101	124
	105410	040	106	117
	105413	122	103	105
	105416	104	040	104
	105421	102	105	040
	105424	114	117	103
	105427	101	124	111
	105432	117	116	000
4786	105435	115	117	126
	105440	040	043	061
	105443	067	067	064
	105446	060	060	054
	105451	050	122	061
	105454	051	040	127
	105457	111	124	110
	105462	040	105	103
	105465	103	040	105
	105470	116	101	102
	105473	114	105	104
	105476	040	103	110
	105501	101	116	107
	105504	105	104	040
	105507	104	101	124
	105512	101	040	101
	105515	124	040	106
	105520	117	122	103
	105523	105	104	040
	105526	104	102	105
	105531	040	114	117
	105534	103	101	124
	105537	111	117	116
	105542	000		
4787	105543	123	102	105
	105546	040	104	105
	105551	124	105	103
	105554	124	105	104
	105557	040	102	131
	105562	040	115	113
	105565	061	061	040
	105570	102	125	124
	105573	040	116	117
	105576	124	040	111
	105601	123	117	114
	105604	101	124	105
	105607	104	040	102
	105612	131	040	120
	105615	122	117	107
	105620	122	101	115

EM33: .ASCIZ /MOV #177400,(R1) WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/

EM34: .ASCIZ /SBE DETECTED BY MK11 BUT NOT ISOLATED BY PROGRAM/



4788	105623	000				
	105624	125	116	105	EM35:	.ASCIZ /UNEXPECTED CORRECTION WITH ECC DISABLE ON FORCED SBE/
	105627	130	120	105		
	105632	103	124	105		
	105635	104	040	103		
	105640	117	122	122		
	105643	105	103	124		
	105646	111	117	116		
	105651	040	127	111		
	105654	124	110	040		
	105657	105	103	103		
	105662	040	104	111		
	105665	123	101	102		
	105670	114	105	040		
	105673	117	116	040		
	105676	106	117	122		
	105701	103	105	104		
	105704	040	123	102		
	105707	105	000			
4789	105711	101	120	124	EM36:	.ASCIZ /APT SIZE DISAGREES WITH PROGRAM SIZING/
	105714	040	123	111		
	105717	132	105	040		
	105722	104	111	123		
	105725	101	107	122		
	105730	105	105	123		
	105733	040	127	111		
	105736	124	110	040		
	105741	120	122	117		
	105744	107	122	101		
	105747	115	040	123		
	105752	111	132	111		
	105755	116	107	000		
4790	105760	122	125	116	EM37:	.ASCIZ /RUNNING CONTENTION INCREMENT FAILURE/
	105763	116	111	116		
	105766	107	040	103		
	105771	117	116	124		
	105774	105	116	124		
	105777	111	117	116		
	106002	040	111	116		
	106005	103	122	105		
	106010	115	105	115		
	106013	124	040	106		
	106016	101	111	114		
	106021	125	122	105		
	106024	000				
4791	106025	101	123	122	EM38:	.ASCIZ /ASRB LOCK FAILURE (1 & ONLY 1 CPU SHOULD HAVE LOCKED) (LOCKED = ONES)/
	106030	102	040	114		
	106033	117	103	113		
	106036	040	106	101		
	106041	111	114	125		
	106044	122	105	040		
	106047	050	061	040		
	106052	046	040	117		
	106055	116	114	131		
	106060	040	061	040		
	106063	103	120	125		
	106066	040	123	110		

	106071	117	125	114	
	106074	104	040	110	
	106077	101	126	105	
	106102	040	114	117	
	106105	103	113	105	
	106110	104	051	040	
	106113	050	114	117	
	106116	103	113	105	
	106121	104	040	075	
	106124	040	117	116	
	106127	105	123	051	
	106132	000			
4792	106133	101	123	122	EM39: .ASCIZ /ASRB LOCK BALANCE FAILURE (# OF LOCKS PER CPU BELOW)/
	106136	102	040	114	
	106141	117	103	113	
	106144	040	102	101	
	106147	114	101	116	
	106152	103	105	040	
	106155	106	101	111	
	106160	114	125	122	
	106163	105	040	050	
	106166	043	040	117	
	106171	106	040	114	
	106174	117	103	113	
	106177	123	040	120	
	106202	105	122	040	
	106205	103	120	125	
	106210	040	102	105	
	106213	114	117	127	
	106216	051	000		
4793	106220	102	122	101	EM40: .ASCIZ /BRANCH GOBBLE FAILED CONDITION CODES TEST/
	106223	116	103	110	
	106226	040	107	117	
	106231	102	102	114	
	106234	105	040	106	
	106237	101	111	114	
	106242	105	104	040	
	106245	103	117	116	
	106250	104	111	124	
	106253	111	117	116	
	106256	040	103	117	
	106261	104	105	123	
	106264	040	124	105	
	106267	123	124	000	
4794	106272	123	124	101	EM41: .ASCIZ /STALE DATA COULD NOT BE CREATED - CACHE FAILURE NOT MAIN MEMORY!!!/
	106275	114	105	040	
	106300	104	101	124	
	106303	101	040	103	
	106306	117	125	114	
	106311	104	040	116	
	106314	117	124	040	
	106317	102	105	040	
	106322	103	122	105	
	106325	101	124	105	
	106330	104	040	055	
	106333	040	103	101	
	106336	103	110	105	

	106341	040	106	101	
	106344	111	114	125	
	106347	122	105	040	
	106352	116	117	124	
	106355	040	115	101	
	106360	111	116	040	
	106363	115	105	115	
	106366	117	122	131	
	106371	041	041	000	
4795	106374	103	101	103	EM42: .ASCIZ /CACHE COULD NOT FLUSH STALE DATA - CACHE FAILURE NOT MAIN MEMORY!!!/
	106377	110	105	040	
	106402	103	117	125	
	106405	114	104	040	
	106410	116	117	124	
	106413	040	106	114	
	106416	125	123	110	
	106421	040	123	124	
	106424	101	114	105	
	106427	040	104	101	
	106432	124	101	040	
	106435	055	040	103	
	106440	101	103	110	
	106443	105	040	106	
	106446	101	111	114	
	106451	125	122	105	
	106454	040	116	117	
	106457	124	040	115	
	106462	101	111	116	
	106465	040	115	105	
	106470	115	117	122	
	106473	131	041	041	
	106476	000			
4796	106477	123	114	101	EM43: .ASCIZ /SLAVE DID NOT RESPOND IN TIME - ABORTING MULTIPROCESSOR TESTS/
	106502	126	105	040	
	106505	104	111	104	
	106510	040	116	117	
	106513	124	040	122	
	106516	105	123	120	
	106521	117	116	104	
	106524	040	111	116	
	106527	040	124	111	
	106532	115	105	040	
	106535	055	040	101	
	106540	102	117	122	
	106543	124	111	116	
	106546	107	040	115	
	106551	125	114	124	
	106554	111	120	122	
	106557	117	103	105	
	106562	123	123	117	
	106565	122	040	124	
	106570	105	123	124	
	106573	123	000		
4797	106575	111	111	123	EM44: .ASCIZ /IIST DID NOT RESPOND IN TIME - ABORTING MULTIPROCESSOR TESTS/
	106600	124	040	104	
	106603	111	104	040	
	106606	116	117	124	

	106611	040	122	105	
	106614	123	120	117	
	106617	116	104	040	
	106622	111	116	040	
	106625	124	111	115	
	106630	105	040	055	
	106633	040	101	102	
	106636	117	122	124	
	106641	111	116	107	
	106644	040	115	125	
	106647	114	124	111	
	106652	120	122	117	
	106655	103	105	123	
	106660	123	117	122	
	106663	040	124	105	
	106666	123	124	123	
	106671	000			
4798	106672	116	117	124	EM45: .ASCIZ /NOT ENOUGH GOOD MEMORY FOR SLAVE - ABORTING MULTIPROCESSOR TESTS/
	106675	040	105	116	
	106700	117	125	107	
	106703	110	040	107	
	106706	117	117	104	
	106711	040	115	105	
	106714	115	117	122	
	106717	131	040	106	
	106722	117	122	040	
	106725	123	114	101	
	106730	126	105	040	
	106733	055	040	101	
	106736	102	117	122	
	106741	124	111	116	
	106744	107	040	115	
	106747	125	114	124	
	106752	111	120	122	
	106755	117	103	105	
	106760	123	123	117	
	106763	122	040	124	
	106766	105	123	124	
	106771	123	000		
4799	106773	101	114	114	EM46: .ASCIZ /ALL CPUS NOT SHARING A MEMORY BANK (ACCESSABLE = ONES)/
	106776	040	103	120	
	107001	125	123	040	
	107004	116	117	124	
	107007	040	123	110	
	107012	101	122	111	
	107015	116	107	040	
	107020	101	040	115	
	107023	105	115	117	
	107026	122	131	040	
	107031	102	101	116	
	107034	113	040	050	
	107037	101	103	103	
	107042	105	123	123	
	107045	101	102	114	
	107050	105	040	075	
	107053	040	117	116	
	107056	105	123	051	

ERROR MESSAGES (EM)

4800	107061	000			
	107062	103	123	122	EM47: .ASCIZ /CSR CONFIGURATION ERROR - PROBABLY CSR SWITCHES IN BOX ARE SET WRONG!//
	107065	040	103	117	
	107070	116	106	111	
	107073	107	125	122	
	107076	101	124	111	
	107101	117	116	040	
	107104	105	122	122	
	107107	117	122	040	
	107112	055	040	120	
	107115	122	117	102	
	107120	101	102	114	
	107123	131	040	103	
	107126	123	122	040	
	107131	123	127	111	
	107134	124	103	110	
	107137	105	123	040	
	107142	111	116	040	
	107145	102	117	130	
	107150	040	101	122	
	107153	105	040	123	
	107156	105	124	040	
	107161	127	122	117	
	107164	116	107	041	
	107167	000			
4801	107170	101	120	124	EM48: .ASCIZ /APT EXPECTS DIFFERENT NUMBER OF CPU'S THAN DIAGNOSTIC CAN BOOT//
	107173	040	105	130	
	107176	120	105	103	
	107201	124	123	040	
	107204	104	111	106	
	107207	106	105	122	
	107212	105	116	124	
	107215	040	116	125	
	107220	115	102	105	
	107223	122	040	117	
	107226	106	040	103	
	107231	120	125	047	
	107234	123	040	124	
	107237	110	101	116	
	107242	040	104	111	
	107245	101	107	116	
	107250	117	123	124	
	107253	111	103	040	
	107256	103	101	116	
	107261	040	102	117	
	107264	117	124	000	
4802	107267	120	117	122	EM49: .ASCIZ /PORT CONTENTION BALANCED WITH 3 CPU'S RAISED TO HIGH PRIORITY//
	107272	124	040	103	
	107275	117	116	124	
	107300	105	116	124	
	107303	111	117	116	
	107306	040	102	101	
	107311	114	101	116	
	107314	103	105	104	
	107317	040	127	111	
	107322	124	110	040	
	107325	063	040	103	

107330	120	125	047
107333	123	040	122
107336	101	111	123
107341	105	104	040
107344	124	117	040
107347	110	111	107
107352	110	040	120
107355	122	111	117
107360	122	111	124
107363	131	000	

4803

.EVEN

Address	PC	DEV	ADD	GOOD	BAD/
4806					
4807	107366	040	040	120	DH1: .SBTTL ERROR DATA HEADERS (DH)
	107371	103	040	040	.ASCIZ / PC DEV ADD GOOD BAD/
	107374	040	104	105	
	107377	126	040	101	
	107402	104	104	040	
	107405	040	040	107	
	107410	117	117	104	
	107413	040	040	040	
	107416	040	102	101	
	107421	104	000		
4808	107423	040	040	120	DH2: .ASCIZ / PC GD-CC BD-CC/
	107426	103	040	040	
	107431	040	107	104	
	107434	055	103	103	
	107437	040	102	104	
	107442	055	103	103	
	107445	000			
4809	107446	040	040	120	DH3: .ASCIZ / PC 1ST ADD # OF ERRORS/
	107451	103	040	040	
	107454	040	061	123	
	107457	124	040	101	
	107462	104	104	040	
	107465	040	043	040	
	107470	117	106	040	
	107473	105	122	122	
	107476	117	122	123	
	107501	000			
4810	107502	040	040	120	DH4: .ASCIZ / PC/
	107505	103	000		
4811	107507	040	040	120	DH5: .ASCII / PC CPUERR MEMERR HIADRS LGADRS/
	107512	103	040	040	
	107515	040	040	103	
	107520	120	125	105	
	107523	122	122	040	
	107526	040	115	105	
	107531	115	105	122	
	107534	122	040	040	
	107537	110	111	101	
	107542	104	122	123	
	107545	040	040	114	
	107550	117	101	104	
	107553	122	123		
4812	107555	040	040	040	.ASCIZ / MMR0 MMR1 MMR2 MMR3/
	107560	115	115	122	
	107563	060	040	040	
	107566	040	040	115	
	107571	115	122	061	
	107574	040	040	040	
	107577	040	115	115	
	107602	122	062	040	
	107605	040	040	040	
	107610	115	115	122	
	107613	063	000		
4813	107615	040	040	120	DH6: .ASCIZ / PC APTCORE MJSIZE APTMOS MKSIZE/
	107620	103	040	040	
	107623	040	101	120	

4838						
4839	111176	200	103	105	MSG000: .SBTTL	MESSAGES
	111201	115	113	101	.ASCIZ	<CRLF>/CEMKABO 1170 MAIN MEMORY DIAGNOSTIC <V14-00>/
	111204	102	060	040		
	111207	061	061	067		
	111212	060	040	115		
	111215	101	111	116		
	111220	040	115	105		
	111223	115	117	122		
	111226	131	040	104		
	111231	111	101	107		
	111234	116	117	123		
	111237	124	111	103		
	111242	040	074	126		
	111245	061	064	055		
	111250	060	060	076		
	111253	000				
4840	111254	200	040	040	MSG001: .ASCIZ	<CRLF>/
	111257	040	040	040		MEMORY CONFIGURATION MAP/
	111262	040	040	040		
	111265	040	040	040		
	111270	040	040	040		
	111273	040	040	040		
	111276	040	040	040		
	111301	040	040	040		
	111304	040	115	105		
	111307	115	117	122		
	111312	131	040	103		
	111315	117	116	106		
	111320	111	107	125		
	111323	122	101	124		
	111326	111	117	116		
	111331	040	115	101		
	111334	120	000			
4841	111336	200	040	040	MSG002: .ASCIZ	<CRLF>/
	111341	040	040	040		16K WORD BANKS/
	111344	040	040	040		
	111347	040	040	040		
	111352	040	040	040		
	111355	040	040	040		
	111360	040	040	040		
	111363	040	040	040		
	111366	040	040	040		
	111371	040	040	040		
	111374	061	066	113		
	111377	040	127	117		
	111402	122	104	040		
	111405	102	101	116		
	111410	113	123	000		
4842	111413	200	040	040	MSG003: .ASCII	<CRLF>/
	111416	040	040	040		1 2 3/
	111421	040	040	040		
	111424	040	040	040		
	111427	040	040	040		
	111432	040	040	061		
	111435	040	040	040		
	111440	040	040	040		

	111443	040	062	040					
	111446	040	040	040					
	111451	040	040	040					
	111454	063							
4843	111455	040	040	040	.ASCIZ /	4	5	6	7 /
	111460	040	040	040					
	111463	040	064	040					
	111466	040	040	040					
	111471	040	040	040					
	111474	065	040	040					
	111477	040	040	040					
	111502	040	040	066					
	111505	040	040	040					
	111510	040	040	040					
	111513	040	067	040					
	111516	040	000						
4844	111520	200	040	040	MSG004: .ASCII <CRLF>/				012345670123456701234567/
	111523	040	040	040					
	111526	040	040	040					
	111531	060	061	062					
	111534	063	064	065					
	111537	066	067	060					
	111542	061	062	063					
	111545	064	065	066					
	111550	067	060	061					
	111553	062	063	064					
	111556	065	066	067					
4845	111561	060	061	062	.ASCIZ /012345670123456701234567012345670123/				
	111564	063	064	065					
	111567	066	067	060					
	111572	061	062	063					
	111575	064	065	066					
	111600	067	060	061					
	111603	062	063	064					
	111606	065	066	067					
	111611	060	061	062					
	111614	063	064	065					
	111617	066	067	060					
	111622	061	062	063					
	111625	000							
4846	111626	200	105	122	MSG005: .ASCIZ <CRLF>/ERRORS /				
	111631	122	117	122					
	111634	123	040	040					
	111637	000							
4847	111640	200	125	116	MSG006: .ASCIZ <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>				
	111643	104	105	106					
	111646	111	116	105					
	111651	104	040	124					
	111654	122	101	120					
	111657	040	111	116					
	111662	123	124	122					
	111665	125	103	124					
	111670	111	117	116					
	111673	032	000						
4848	111675	200	111	116	MSG007: .ASCIZ <CRLF>/INTRLV /				
	111700	124	122	114					
	111703	126	040	040					

	112144	040	040	040	
	112147	000			
4858	112150	040	040	040	MSG017: .ASCIZ / / ;8 SPACES
	112153	040	040	040	
	112156	040	040	000	
4859	112161	040	040	000	MSG018: .ASCIZ / / ;2 SPACES
4860	112164	040	040	040	MSG019: .ASCIZ / / ;3 SPACES
	112167	000			
4861	112170	200	106	111	MSG020: .ASCIZ <CRLF>/FIELD SERVICE COMMAND MODE/
	112173	105	114	104	
	112176	040	123	105	
	112201	122	126	111	
	112204	103	105	040	
	112207	103	117	115	
	112212	115	101	116	
	112215	104	040	115	
	112220	117	104	105	
	112223	000			
4862	112224	200	103	117	MSG021: .ASCII <CRLF>/COMMANDS AVAILABLE ARE:/
	112227	115	115	101	
	112232	116	104	123	
	112235	040	101	126	
	112240	101	111	114	
	112243	101	102	114	
	112246	105	040	101	
	112251	122	105	072	
4863	112254	200	060	040	.ASCII <CRLF>/0 = EXIT FIELD SERVICE COMMANDS/
	112257	075	040	105	
	112262	130	111	124	
	112265	040	106	111	
	112270	105	114	104	
	112273	040	123	105	
	112276	122	126	111	
	112301	103	105	040	
	112304	103	117	115	
	112307	115	101	116	
	112312	104	123		
4864	112314	200	061	040	.ASCII <CRLF>/1 = READ CSR/
	112317	075	040	122	
	112322	105	101	104	
	112325	040	103	123	
	112330	122			
4865	112331	200	062	040	.ASCII <CRLF>/2 = LOAD CSR/
	112334	075	040	114	
	112337	117	101	104	
	112342	040	103	123	
	112345	122			
4866	112346	200	063	040	.ASCII <CRLF>/3 = EXAMINE MEMORY/
	112351	075	040	105	
	112354	130	101	115	
	112357	111	116	105	
	112362	040	115	105	
	112365	115	117	122	
	112370	131			
4867	112371	200	064	040	.ASCII <CRLF>/4 = MODIFY MEMORY/
	112374	075	040	115	
	112377	117	104	111	

MESSAGES

	112402	106	131	040	
	112405	115	105	115	
	112410	117	122	131	
4868	112413	200	065	040	.ASCII <CRLF>/5 = SELECT BANK, MARGIN, & PATTERN/
	112416	075	040	123	
	112421	105	114	105	
	112424	103	124	040	
	112427	102	101	116	
	112432	113	054	115	
	112435	101	122	107	
	112440	111	116	054	
	112443	040	046	040	
	112446	120	101	124	
	112451	124	105	122	
	112454	116			
4869	112455	200	066	040	.ASCII <CRLF>/6 = TYPE CONFIGURATION MAP/
	112460	075	040	124	
	112463	131	120	105	
	112466	040	103	117	
	112471	116	106	111	
	112474	107	125	122	
	112477	101	124	111	
	112502	117	116	040	
	112505	115	101	120	
4870	112510	200	067	040	.ASCII <CRLF>'7 = BATTERY BACKUP - P/F TEST''
	112513	075	040	102	
	112516	101	124	124	
	112521	105	122	131	
	112524	040	102	101	
	112527	103	113	125	
	112532	120	040	055	
	112535	040	120	057	
	112540	106	040	124	
	112543	105	123	124	
4871	112546	200	070	040	.ASCII <CRLF>/8 = SOB-A-LONG TEST/
	112551	075	040	123	
	112554	117	102	055	
	112557	101	055	114	
	112562	117	116	107	
	112565	040	124	105	
	112570	123	124		
4872	112572	200	071	040	.ASCII <CRLF>/9 = SUPER TIGHT SCOPE LOOP/
	112575	075	040	123	
	112600	125	120	105	
	112603	122	040	124	
	112606	111	107	110	
	112611	124	040	123	
	112614	103	117	120	
	112617	105	040	114	
	112622	117	117	120	
4873	112625	200	061	060	.ASCII <CRLF>/10= ERROR SUMMARY/
	112630	075	040	105	
	112633	122	122	117	
	112636	122	040	123	
	112641	125	115	115	
	112644	101	122	131	
4874	112647	200	061	061	.ASCII <CRLF>/11= REFRESH TEST/

Line	Address	Hex	Hex	Hex	Description
	112652	075	040	122	
	112655	105	106	122	
	112660	105	123	110	
	112663	040	124	105	
	112666	123	124		
4875	112670	200	061	062	.ASCII <CRLF>/12= SET FILL COUNT/
	112673	075	040	123	
	112676	105	124	040	
	112701	106	111	114	
	112704	114	040	103	
	112707	117	125	116	
	112712	124			
4876	112713	200	061	063	.ASCII <CRLF>/13= ENTER KAMIKAZE MODE/
	112716	075	040	105	
	112721	116	124	105	
	112724	122	040	113	
	112727	101	115	111	
	112732	113	101	132	
	112735	105	040	115	
	112740	117	104	105	
4877	112743	200	061	064	.ASCII <CRLF>/14= EXIT KAMIKAZE MODE/
	112746	075	040	105	
	112751	130	111	124	
	112754	040	113	101	
	112757	115	111	113	
	112762	101	132	105	
	112765	040	115	117	
	112770	104	105		
4878	112772	200	061	065	.ASCII <CRLF>/15= TURN CACHE OFF/
	112775	075	040	124	
	113000	125	122	116	
	113003	040	103	101	
	113006	103	110	105	
	113011	040	117	106	
	113014	106			
4879	113015	200	061	066	.ASCII <CRLF>/16= TURN CACHE ON/
	113020	075	040	124	
	113023	125	122	116	
	113026	040	103	101	
	113031	103	110	105	
	113034	040	117	116	
4880	113037	200	061	067	.ASCII <CRLF>/17= RUN ONLY MULTIPOINT TESTS/
	113042	075	040	122	
	113045	125	116	040	
	113050	117	116	114	
	113053	131	040	115	
	113056	125	114	124	
	113061	111	120	117	
	113064	122	124	040	
	113067	124	105	123	
	113072	124	123		
4881	113074	200	061	070	.ASCII <CRLF>/18= RESUME RUNNING BOTH SINGLE & MULTIPOINT TESTS/
	113077	075	040	122	
	113102	105	123	125	
	113105	115	105	040	
	113110	122	125	116	
	113113	116	111	116	

MESSAGES

	113116	107	040	102	
	113121	117	124	110	
	113124	040	123	111	
	113127	116	107	114	
	113132	105	040	046	
	113135	040	115	125	
	113140	114	124	111	
	113143	120	117	122	
	113146	124	040	124	
	113151	105	123	124	
	113154	123			
4882	113155	200	061	071	.ASCII <CRLF>/19= TEST ONLY SELECTED BANKS/
	113160	075	040	124	
	113163	105	123	124	
	113166	040	117	116	
	113171	114	131	040	
	113174	123	105	114	
	113177	105	103	124	
	113202	105	104	040	
	113205	102	101	116	
	113210	113	123		
4883	113212	200	062	060	.ASCII <CRLF>/20= RESUME TESTING ALL BANKS/
	113215	075	040	122	
	113220	105	123	125	
	113223	115	105	040	
	113226	124	105	123	
	113231	124	111	116	
	113234	107	040	101	
	113237	114	114	040	
	113242	102	101	116	
	113245	113	123		
4884	113247	015	012	000	.BYTE 15,12,0
4885	113252	200	127	110	MSG022: .ASCIZ <CRLF>/WHICH CSR(0-7)? /
	113255	111	103	110	
	113260	040	103	123	
	113263	122	050	060	
	113266	055	067	051	
	113271	077	040	000	
4886	113274	200	103	123	MSG023: .ASCIZ <CRLF>/CSR - 1ST WORD? /
	113277	122	040	055	
	113302	040	061	123	
	113305	124	040	127	
	113310	117	122	104	
	113313	077	040	000	
4887	113316	200	103	123	MSG024: .ASCIZ <CRLF>/CSR - 2ND WORD? /
	113321	122	040	055	
	113324	040	062	116	
	113327	104	040	127	
	113332	117	122	104	
	113335	077	040	000	
4888	113340	200	124	110	MSG025: .ASCIZ <CRLF>/THIS CSR DOES NOT EXIST/
	113343	111	123	040	
	113346	103	123	122	
	113351	040	104	117	
	113354	105	123	040	
	113357	116	117	124	
	113362	040	105	130	

	113365	111	123	124	
	113370	000			
4889	113371	200	103	117	MSG026: .ASCIZ <CRLF>/COMMAND:/
	113374	115	115	101	
	113377	116	104	072	
	113402	000			
4890	113403	200	117	114	MSG027: .ASCIZ <CRLF>/OLD CSR WAS/
	113406	104	040	103	
	113411	123	122	040	
	113414	127	101	123	
	113417	000			
4891	113420	200	103	123	MSG028: .ASCIZ <CRLF>/CSR IS NOW/
	113423	122	040	111	
	113426	123	040	116	
	113431	117	127	000	
4892	113434	200	105	130	MSG029: .ASCIZ <CRLF>/EXAMINE MEMORY/
	113437	101	115	111	
	113442	116	105	040	
	113445	115	105	115	
	113450	117	122	131	
	113453	000			
4893	113454	200	102	101	MSG030: .ASCIZ <CRLF>/BANK(0-167)? /
	113457	116	113	050	
	113462	060	055	061	
	113465	066	067	051	
	113470	077	040	000	
4894	113473	200	120	110	MSG031: .ASCIZ <CRLF>/PHYSICAL ADDRESS(0-16777776)? /
	113476	131	123	111	
	113501	103	101	114	
	113504	040	101	104	
	113507	104	122	105	
	113512	123	123	050	
	113515	060	055	061	
	113520	066	067	067	
	113523	067	067	067	
	113526	066	051	077	
	113531	040	000		
4895	113533	200	120	101	MSG032: .ASCIZ <CRLF>/PARITY ABORT/<32>
	113536	122	111	124	
	113541	131	040	101	
	113544	102	117	122	
	113547	124	032	000	
4896	113552	200	124	111	MSG033: .ASCIZ <CRLF>/TIMEOUT TRAP/<32>
	113555	115	105	117	
	113560	125	124	040	
	113563	124	122	101	
	113566	120	032	000	
4897	113571	200	102	131	MSG034: .ASCII <CRLF>/BYPASSING CSR TESTS ON BOX /
	113574	120	101	123	
	113577	123	111	116	
	113602	107	040	103	
	113605	123	122	040	
	113610	124	105	123	
	113613	124	123	040	
	113616	117	116	040	
	113621	102	117	130	
	113624	040			

4898	113625	000			MSG034: .BYTE 0
4899	113626	040	103	117	.ASCII / CONTROLLER /
	113631	116	124	122	
	113634	117	114	114	
	113637	105	122	040	
4900	113642	000			MSG034: .BYTE 0
4901	113643	200	104	125	.ASCIIZ <CRLF>/DUE TO LACK OF SBE FREE LOCATIONS/<32>
	113646	105	040	124	
	113651	117	040	114	
	113654	101	103	113	
	113657	040	117	106	
	113662	040	123	102	
	113665	105	040	106	
	113670	122	105	105	
	113673	040	114	117	
	113676	103	101	124	
	113701	111	117	116	
	113704	123	032	000	
4902	113707	121	126	000	MSG035: .ASCIIZ /QV/
4903	113712	200	115	117	MSG036: .ASCIIZ <CRLF>/MODIFY MEMORY/
	113715	104	111	106	
	113720	131	040	115	
	113723	105	115	117	
	113726	122	131	000	
4904	113731	200	117	114	MSG037: .ASCIIZ <CRLF>/OLD DATA WAS /
	113734	104	040	104	
	113737	101	124	101	
	113742	040	127	101	
	113745	123	040	000	
4905	113750	200	104	101	MSG038: .ASCIIZ <CRLF>/DATA IS NOW /
	113753	124	101	040	
	113756	111	123	040	
	113761	116	117	127	
	113764	040	000		
4906	113766	200	111	116	MSG039: .ASCIIZ <CRLF>/INPUT NEW DATA? /
	113771	120	125	124	
	113774	040	116	105	
	113777	127	040	104	
	114002	101	124	101	
	114005	077	040	000	
4907	114010	200	123	105	MSG040: .ASCIIZ <CRLF>/SELECT BANK, MARGIN, & PATTERN TEST/
	114013	114	105	103	
	114016	124	040	102	
	114021	101	116	113	
	114024	054	040	115	
	114027	101	122	107	
	114032	111	116	054	
	114035	040	046	040	
	114040	120	101	124	
	114043	124	105	122	
	114046	116	040	124	
	114051	105	123	124	
	114054	000			
4908	114055	200	102	101	MSG041: .ASCIIZ <CRLF>/BANK NOT ACCESSABLE/
	114060	116	113	040	
	114063	116	117	124	
	114066	040	101	103	

	114071	103	105	123	
	114074	123	101	102	
	114077	114	105	000	
4909	114102	200	120	101	MSG042: .ASCIZ <CRLF>/PATTERN(0-37)? /
	114105	124	124	105	
	114110	122	116	050	
	114113	060	055	063	
	114116	067	051	077	
	114121	040	000		
4910	114123	200	120	101	MSG043: .ASCIZ <CRLF>/PATTERN 0 DATA IS? /
	114126	124	124	105	
	114131	122	116	040	
	114134	060	040	104	
	114137	101	124	101	
	114142	040	111	123	
	114145	077	040	000	
4911	114150	200	115	101	MSG044: .ASCIZ <CRLF>/MARGIN(0,2,3,4,5)? /
	114153	122	107	111	
	114156	116	050	060	
	114161	054	062	054	
	114164	063	054	064	
	114167	054	065	051	
	114172	077	040	000	
4912	114175	200	115	101	MSG045: .ASCIZ <CRLF>/MARGINS INHIBITED BY SW11! /
	114200	122	107	111	
	114203	116	123	040	
	114206	111	116	110	
	114211	111	102	111	
	114214	124	105	104	
	114217	040	102	131	
	114222	040	123	127	
	114225	061	061	041	
	114230	000			
4913	114231	200	124	117	MSG046: .ASCIZ <CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
	114234	040	105	123	
	114237	103	101	120	
	114242	105	040	124	
	114245	131	120	105	
	114250	040	101	116	
	114253	131	040	113	
	114256	105	131	200	
	114261	012	012	000	
4914	114264	200	124	105	MSG047: .ASCIZ <CRLF>/TEST COMPLETE /
	114267	123	124	040	
	114272	103	117	115	
	114275	120	114	105	
	114300	124	105	000	
4915	114303	040	116	117	MSG048: .ASCIZ / NOT AVAILABLE NOW - TRY LATER! /
	114306	124	040	101	
	114311	126	101	111	
	114314	114	101	102	
	114317	114	105	040	
	114322	116	117	127	
	114325	040	055	040	
	114330	124	122	131	
	114333	040	114	101	
	114336	124	105	122	

4916	114341	041	000	101	MSG049: .ASCIZ <CRLF>/BANK REQUIRES RELOCATION/
	114343	200	102	101	
	114346	116	113	040	
	114351	122	105	121	
	114354	125	111	122	
	114357	105	123	040	
	114362	122	105	114	
	114365	117	103	101	
	114370	124	111	117	
4917	114373	116	000		
	114375	200	102	101	MSG050: .ASCII <CRLF>/BATTERY BACKUP TEST/
	114400	124	124	105	
	114403	122	131	040	
	114406	102	101	103	
	114411	113	125	120	
	114414	040	124	105	
	114417	123	124		
4918	114421	200	127	122	.ASCIZ <CRLF>/WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND/
	114424	111	124	111	
	114427	116	107	040	
	114432	046	040	103	
	114435	110	105	103	
	114440	113	111	116	
	114443	107	040	101	
	114446	104	104	122	
	114451	105	123	123	
	114454	040	120	101	
	114457	124	124	105	
	114462	122	116	040	
	114465	101	123	040	
	114470	102	101	103	
	114473	113	107	122	
	114476	117	125	116	
4919	114501	104	000		
	114503	200	120	117	MSG051: .ASCIZ <CRLF>/POWER RECOVERY/
	114506	127	105	122	
	114511	040	122	105	
	114514	103	117	126	
	114517	105	122	131	
	114522	000			
4920	114523	200	122	105	MSG052: .ASCIZ <CRLF>/REMOVE SYSTEM POWER FOR/
	114526	115	117	126	
	114531	105	040	123	
	114534	131	123	124	
	114537	105	115	040	
	114542	120	117	127	
	114545	105	122	040	
	114550	106	117	122	
	114553	000			
4921	114554	040	123	105	MSG053: .ASCIZ / SECONDS MAX!//
	114557	103	117	116	
	114562	104	123	040	
	114565	115	101	130	
	114570	041	000		
4922	114572	200	116	117	MSG054: .ASCIZ <CRLF>/NOW STARTING READ TEST OF MEMORY BANKS/
	114575	127	040	123	
	114600	124	101	122	

	114603	124	111	116	
	114606	107	040	122	
	114611	105	101	104	
	114614	040	124	105	
	114617	123	124	040	
	114622	117	106	040	
	114625	115	105	115	
	114630	117	122	131	
	114633	040	102	101	
	114636	116	113	123	
	114641	000			
4923	114642	200	123	117	MSG055: .ASCIZ <CRLF>/SOB-A-LONG TEST/
	114645	102	055	101	
	114650	055	114	117	
	114653	116	107	040	
	114656	124	105	123	
	114661	124	000		
4924	114663	200	102	105	MSG056: .ASCIZ <CRLF>/BELL = EACH PASS COMPLETE/
	114666	114	114	040	
	114671	075	040	105	
	114674	101	103	110	
	114677	040	120	101	
	114702	123	123	040	
	114705	103	117	115	
	114710	120	114	105	
	114713	124	105	000	
4925	114716	040	077	040	MSG057: .ASCIZ / ? /
	114721	000			
4926	114722	200	040	040	MSG058: .ASCIZ <CRLF>/ CSR CSR+2 CSR CSR+2 .../
	114725	103	123	122	
	114730	040	040	040	
	114733	040	103	123	
	114736	122	053	062	
	114741	040	040	040	
	114744	040	103	123	
	114747	122	040	040	
	114752	040	040	103	
	114755	123	122	053	
	114760	062	040	056	
	114763	056	056	000	
4927	114766	200	123	125	MSG059: .ASCIZ <CRLF>/SUPER TIGHT SCOPE LOOP/
	114771	120	105	122	
	114774	040	124	111	
	114777	107	110	124	
	115002	040	123	103	
	115005	117	120	105	
	115010	040	114	117	
	115013	117	120	000	
4928	115016	200	123	127	MSG060: .ASCIZ <CRLF>/SWREG ----> MEM, MEM ----> DISPLAY/
	115021	122	105	107	
	115024	040	055	055	
	115027	055	076	040	
	115032	115	105	115	
	115035	054	040	115	
	115040	105	115	040	
	115043	055	055	055	
	115046	076	040	104	

	115051	111	123	120	
	115054	114	101	131	
	115057	000			
4929	115060	077	077	077	MSG061: .ASCIZ /??????/
	115063	077	077	077	
	115066	000			
4930	115067	111	116	120	MSG062: .ASCIZ /INPUT MUST BE A/
	115072	125	124	040	
	115075	115	125	123	
	115100	124	040	102	
	115103	105	040	101	
	115106	000			
4931	115107	116	040	117	MSG063: .ASCIZ /N OCTAL /
	115112	103	124	101	
	115115	114	040	000	
4932	115120	116	125	115	MSG064: .ASCIZ /NUMBER/<CRLF>
	115123	102	105	122	
	115126	200	000		
4933	115130	040	104	105	MSG065: .ASCIZ / DECIMAL /
	115133	103	111	115	
	115136	101	114	040	
	115141	000			
4934	115142	200	105	122	MSG066: .ASCIZ <CRLF>/ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN/
	115145	122	117	122	
	115150	040	103	117	
	115153	125	116	124	
	115156	040	105	130	
	115161	103	105	105	
	115164	104	105	104	
	115167	040	062	060	
	115172	040	055	040	
	115175	101	102	117	
	115200	122	124	111	
	115203	116	107	040	
	115206	106	117	122	
	115211	040	130	130	
	115214	104	120	040	
	115217	103	110	101	
	115222	111	116	000	
4935	115225	106	101	124	MSG067: .ASCIZ /FATAL /
	115230	101	114	040	
	115233	000			
4936	115234	115	112	061	MSG068: .ASCIZ /MJ11/
	115237	061	000		
4937	115241	115	113	061	MSG069: .ASCIZ /MK11/
	115244	061	000		
4938	115246	113	040	127	MSG070: .ASCIZ /K WORDS OF MEMORY/<CRLF>
	115251	117	122	104	
	115254	123	040	117	
	115257	106	040	115	
	115262	105	115	117	
	115265	122	131	200	
	115270	000			
4939	115271	113	040	117	MSG071: .ASCIZ /K OF MJ11/<CRLF>
	115274	106	040	115	
	115277	112	061	061	
	115302	200	000		

MESSAGES

4940	115304	113	040	117	MSG072: .ASCIZ /K OF MK11/<CRLF>
	115307	106	040	115	
	115312	113	061	061	
	115315	200	000		
4941	115317	200	122	105	MSG073: .ASCIZ <CRLF>/REFRESH TEST/
	115322	106	122	105	
	115325	123	110	040	
	115330	124	105	123	
	115333	124	000		
4942					.EVEN ;THIS STAYS WITH MSG074
4943	115336	077	077	000	MSG074: .ASCIZ /??/
4944	115341	200	122	105	MSG075: .ASCIZ <CRLF>/RELOCATION NOT POSSIBLE/<32>
	115344	114	117	103	
	115347	101	124	111	
	115352	117	116	040	
	115355	116	117	124	
	115360	040	120	117	
	115363	123	123	111	
	115366	102	114	105	
	115371	032	000		
4945	115373	200	040	040	MSG076: .ASCIZ <CRLF>/ BANK ERRORS/<CRLF>
	115376	102	101	116	
	115401	113	040	040	
	115404	105	122	122	
	115407	117	122	123	
	115412	200	000		
4946	115414	200	105	116	MSG077: .ASCIZ <CRLF>/END PASS #/
	115417	104	040	120	
	115422	101	123	123	
	115425	040	043	000	
4947	115430	200	127	101	MSG078: .ASCIZ <CRLF>/WARNING! ECC IS DISABLED ON BOX /<32>
	115433	122	116	111	
	115436	116	107	041	
	115441	040	040	105	
	115444	103	103	040	
	115447	111	123	040	
	115452	104	111	123	
	115455	101	102	114	
	115460	105	104	040	
	115463	117	116	040	
	115466	102	117	130	
	115471	040	032	000	
4948	115474	040	105	122	MSG079: .ASCIZ / ERROR(S) DETECTED/<CRLF>
	115477	122	117	122	
	115502	050	123	051	
	115505	040	104	105	
	115510	124	105	103	
	115513	124	105	104	
	115516	200	000		
4949	115520	200	105	116	MSG080: .ASCIZ <CRLF>/ENTER # OF SLAVE CPU'S (0-3)? /
	115523	124	105	122	
	115526	040	043	040	
	115531	117	106	040	
	115534	123	114	101	
	115537	126	105	040	
	115542	103	120	125	
	115545	047	123	040	

	115550	050	060	050	
	115553	063	051	077	
	115556	040	000		
4950	115560	200	123	124	MSG081: .ASCIZ <CRLF>/START SLAVE #/
	115563	101	122	124	
	115566	040	123	114	
	115571	101	126	105	
	115574	040	043	000	
4951	115577	040	101	124	MSG082: .ASCIZ / AT ADDRESS 200/
	115602	040	101	104	
	115605	104	122	105	
	115610	123	123	040	
	115613	062	060	060	
	115616	000			
4952	115617	200	120	122	MSG084: .ASCIZ <CRLF>/PROCEEDING AS A SINGLE PORT TEST/
	115622	117	103	105	
	115625	105	104	111	
	115630	116	107	040	
	115633	101	123	040	
	115636	101	040	123	
	115641	111	116	107	
	115644	114	105	040	
	115647	120	117	122	
	115652	124	040	124	
	115655	105	123	124	
	115660	000			
4953	115661	200	106	111	MSG085: .ASCIZ <CRLF>/FILL COUNT(OCTAL)? /
	115664	114	114	040	
	115667	103	117	125	
	115672	116	124	050	
	115675	117	103	124	
	115700	101	114	051	
	115703	077	040	000	
4954	115706	200	115	125	MSG086: .ASCII <CRLF>/MULTIPORT DIRECTORY/
	115711	114	124	111	
	115714	120	117	122	
	115717	124	040	104	
	115722	111	122	105	
	115725	103	124	117	
	115730	122	131		
4955	115732	200	115	101	.ASCIZ <CRLF>/MASTER = /
	115735	123	124	105	
	115740	122	040	075	
	115743	040	000		
4956	115745	200	123	114	MSG087: .ASCII <CRLF>/SLAVE/
	115750	101	126	105	
4957	115753	060			MSG087: .BYTE '0
4958	115754	040	075	040	.ASCIZ / = /
	115757	000			
4959	115760	200	113	105	MSG088: .ASCIZ <CRLF>/KERNEL STACK/
	115763	122	116	105	
	115766	114	040	123	
	115771	124	101	103	
	115774	113	000		
4960	115776	200	123	125	MSG089: .ASCIZ <CRLF>/SUPERVISOR STACK/
	116001	120	105	122	
	116004	126	111	123	

	116007	117	122	040	
	116012	123	124	101	
	116015	103	113	000	
4961	116020	200	125	123	MSG090: .ASCIZ <CRLF>/USER STACK/
	116023	105	122	040	
	116026	123	124	101	
	116031	103	113	000	
4962	116034	040	111	123	MSG091: .ASCIZ / IS EMPTY/
	116037	040	105	115	
	116042	120	124	131	
	116045	000			
4963	116046	122	105	114	MSG092: .ASCIZ /RELOCATED /
	116051	117	103	101	
	116054	124	105	104	
	116057	040	040	000	
4964	116062	102	101	116	MSG093: .ASCIZ /BANK=/ MSG094: .ASCIZ / MAR=/ MSG095: .ASCIZ / PAT=/ MSG096: .ASCIZ <CRLF>/LOADERS BEING DESTROYED/
4965	116065	113	075	000	
	116070	040	040	115	
	116073	101	122	075	
	116076	000			
4966	116077	040	040	120	
	116102	101	124	075	
	116105	000			
4967	116106	200	114	117	
	116111	101	104	105	
	116114	122	123	040	
	116117	102	105	111	
	116122	116	107	040	
	116125	104	105	123	
	116130	124	122	117	
	116133	131	105	104	
	116136	000			
4968	116137	200	127	101	MSG097: .ASCIZ <CRLF>/WARNING! ILLEGAL INTERLEAVE SELECTED (FIELD SERVICE ONLY)/<32>
	116142	122	116	111	
	116145	116	107	041	
	116150	040	040	111	
	116153	114	114	105	
	116156	107	101	114	
	116161	040	111	116	
	116164	124	105	122	
	116167	114	105	101	
	116172	126	105	040	
	116175	123	105	114	
	116200	105	103	101	
	116203	105	104	040	
	116206	050	106	111	
	116211	105	114	104	
	116214	040	123	105	
	116217	122	126	111	
	116222	103	105	040	
	116225	117	116	114	
	116230	131	051	032	
	116233	000			
4969	116234	200	123	114	MSG098: .ASCII <CRLF>/SLAVE/
	116237	101	126	105	
4970	116242	000			MSG098: .BYTE 0
4971	116243	076	000		.ASCIZ />/

4972	116245	200	115	101	MSG099: .ASCIZ <CRLF>/MASTER/
	116250	123	124	105	
	116253	122	076	000	
4973	116256	200	115	125	MSG100: .ASCIZ <CRLF>/MULTIPROCESSOR TEST #/
	116261	114	124	111	
	116264	120	122	117	
	116267	103	105	123	
	116272	123	117	122	
	116275	040	124	105	
	116300	123	124	040	
	116303	043	000		
4974	116305	200	105	116	MSG101: .ASCIZ <CRLF>/ENTERING KAMIKAZE MODE/
	116310	124	105	122	
	116313	111	116	107	
	116316	040	113	101	
	116321	115	111	113	
	116324	101	132	105	
	116327	040	115	117	
	116332	104	105	000	
4975	116335	200	114	105	MSG102: .ASCIZ <CRLF>/LEAVING KAMIKAZE MODE/
	116340	101	126	111	
	116343	116	107	040	
	116346	113	101	115	
	116351	111	113	101	
	116354	132	105	040	
	116357	115	117	104	
	116362	105	000		
4976	116364	200	114	105	MSG103: .ASCIZ <CRLF>/LEAVING FIELD SERVICE MODE/<CRLF>
	116367	101	126	111	
	116372	116	107	040	
	116375	106	111	105	
	116400	114	104	040	
	116403	123	105	122	
	116406	126	111	103	
	116411	105	040	115	
	116414	117	104	105	
	116417	200	000		
4977	116421	032	000		MSG104: .BYTE 32,0 :CONTROL Z
4978	116423	200	105	116	MSG105: .ASCIZ <CRLF>/ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINATE (177)/
	116426	124	105	122	
	116431	040	102	101	
	116434	116	113	123	
	116437	040	111	116	
	116442	040	117	103	
	116445	124	101	114	
	116450	040	055	040	
	116453	125	123	105	
	116456	040	116	125	
	116461	115	102	105	
	116464	122	040	117	
	116467	125	124	123	
	116472	111	104	105	
	116475	040	122	101	
	116500	116	107	105	
	116503	040	124	117	
	116506	040	124	105	
	116511	122	115	111	

	116514	116	101	124	
	116517	105	040	050	
	116522	061	067	067	
	116525	051	000		
4979	116527	200	103	101	MSG106: .ASCIZ <CRLF>/CACHE IS OFF/
	116532	103	110	105	
	116535	040	111	123	
	116540	040	117	106	
	116543	106	000		
4980	116545	200	103	101	MSG107: .ASCIZ <CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
	116550	103	110	105	
	116553	040	111	123	
	116556	040	117	116	
	116561	040	050	105	
	116564	130	103	105	
	116567	120	124	040	
	116572	104	125	122	
	116575	111	116	107	
	116600	040	101	103	
	116603	124	125	101	
	116606	114	040	120	
	116611	101	124	124	
	116614	105	122	116	
	116617	123	051	000	
4981	116622	200	117	116	MSG108: .ASCIZ <CRLF>/ONLY MULTIPOINT TESTS WILL BE RUN/
	116625	114	131	040	
	116630	115	125	114	
	116633	124	111	120	
	116636	117	122	124	
	116641	040	124	105	
	116644	123	124	123	
	116647	040	127	111	
	116652	114	114	040	
	116655	102	105	040	
	116660	122	125	116	
	116663	000			
4982	116664	200	102	117	MSG109: .ASCIZ <CRLF>/BOTH SINGLE & MULTIPOINT TESTS WILL BE RUN/
	116667	124	110	040	
	116672	123	111	116	
	116675	107	114	105	
	116700	040	046	040	
	116703	115	125	114	
	116706	124	111	120	
	116711	117	122	124	
	116714	040	124	105	
	116717	123	124	123	
	116722	040	127	111	
	116725	114	114	040	
	116730	102	105	040	
	116733	122	125	116	
	116736	000			
4983	116737	200	117	116	MSG110: .ASCIZ <CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
	116742	114	131	040	
	116745	123	105	114	
	116750	105	103	124	
	116753	105	104	040	
	116756	102	101	116	

	116761	113	123	040	
	116764	127	111	114	
	116767	114	040	102	
	116772	105	040	124	
	116775	105	123	124	
	117000	105	104	000	
4984	117003	200	101	114	MSG111: .ASCIZ <CRLF>/ALL BANKS WILL BE TESTED/
	117006	114	040	102	
	117011	101	116	113	
	117014	123	040	127	
	117017	111	114	114	
	117022	040	102	105	
	117025	040	124	105	
	117030	123	124	105	
	117033	104	000		
4985	117035	200	120	122	MSG112: .ASCIZ <CRLF>/PROGRAM HALTING BECAUSE OF SWITCH #8/
	117040	117	107	122	
	117043	101	115	040	
	117046	110	101	114	
	117051	124	111	116	
	117054	107	040	102	
	117057	105	103	101	
	117062	125	123	105	
	117065	040	117	106	
	117070	040	123	127	
	117073	111	124	103	
	117076	110	040	043	
	117101	070	000		
4986	117103	040	115	101	MSG113: .ASCIZ / MASTER/
	117106	123	124	105	
	117111	122	000		
4987	117113	040	123	114	MSG114: .ASCIZ / SLAVE1/
	117116	101	126	105	
	117121	061	000		
4988	117123	040	123	114	MSG115: .ASCIZ / SLAVE2/
	117126	101	126	105	
	117131	062	000		
4989	117133	040	123	114	MSG116: .ASCIZ / SLAVE3/
	117136	101	126	105	
	117141	063	000		
4990					.EVEN
4996	117144				\$\$END
4997	117144				END:
4998	117144	000406			.PRINT 60000-SUPLIMIT ;SUPERVISOR ADDRESSES LEFT
5003	117144	002004			.PRINT 100000-SLAVEND ;ADDRESSES LEFT IN SLAVE UNIQUE CODE
5008	117144	020634			.PRINT 140000-END ;ADDRESSES LEFT IN 24K
5012		000200			.END START3

SYMBOL TABLE

ABORTF	002142	BIT4	= 000020	B144	062364	B36	024446	CMD20A	050576
ACFLAG	002124	BIT5	= 000040	B145	062450	B37	024452	CMD5A	045456
ACK	002672	BIT6	= 000100	B146	062572	B4	010550	CMD5B	045666
ACTFLA	002442	BIT7	= 000200	B147	062630	B40	024616	CMD5C	046052
ADDPAR	014570	BIT8	= 000400	B15	013004	B41	024624	CMD7A	046336
ADDRS	002040	BIT9	= 001000	B150	062632	B42	025034	CMD8A	046550
ADD..0	071152	BLOCK1	043710	B151	062674	B43	033412	CMD8B	046740
ADD..1	071162	BLOCK2	043730	B152	063032	B44	033450	CMD8C	047014
ADD..2	071202	BLOCK3	043744	B153	063070	B45	033464	CMD9A	047240
ADD..3	071232	BMFLAG	002136	B154	063120	B46	033604	CMD9B	047456
ADD..4	071262	BOOT	043416	B155	063170	B47	033764	CMD9C	047464
ADD..5	071264	BOOTLO=	000012	B156	063172	B5	010554	CMD9LO	047444
ADD..6	071304	BOOT1	043460	B157	063330	B50	034006	CONFGE	002576
ADD..7	071334	BRGOBB	035112	B16	013014	B51	040014	CONFGT	006666
ALLCPU	002510	B0	005500	B160	063404	B52	040020	CONFG1	006304
APTCOR	002522	B1	007742	B161	063410	B53	040160	CONFG3	006520
APTDOW	043640	B10	011352	B162	063472	B54	040164	CONFG5	006712
APTFLA	002444	B100	053774	B163	063522	B55	040450	CONFIE	004336
APTHAN	015050	B101	054102	B164	063602	B56	040454	CONF IG	003016
APTHLT	043706	B102	054146	B165	063604	B57	041144	CONF SP	006754
APTMOS	002524	B103	054226	B166	063754	B6	010642	CONTFI	002332
APTSIZ	002544	B104	054234	B167	064016	B60	041152	CONTP	073112
ASRBAL=	000110	B105	054256	B17	013016	B61	041302	CONTRL=	177746
A3	= 000002	B106	054370	B170	064052	B62	041306	CONTS	073544
BACKGN	035344	B107	054416	B171	064122	B63	042530	CONTS1	073052
BAD	002056	B11	011360	B172	064124	B64	046374	CONTS2	073546
BADCSR	002000	B110	054472	B173	064246	B65	046744	CONTT	073404
BADPC	002026	B111	054474	B174	064260	B66	047612	COREHI	043300
BADPSW	002036	B112	055374	B175	064316	B67	050170	COUNT	002462
BADSP	002032	B113	055400	B176	064372	B7	010732	CPUBIT	002114
BADSTA	036770	B114	055700	B177	064512	B70	050512	CPUERR=	177766
BADXOR	002064	B115	055706	B2	010312	B71	050512	CPUID	011656
BAD2	002060	B116	057452	B20	013124	B72	050600	CR	= 000015
BAD3	002062	B117	057474	B200	070344	B73	050752	CRLF	= 000200
BAFPAF	015204	B12	012140	B201	070436	B74	053016	CSR	002144
BAFPAR	015320	B120	057622	B202	070736	B75	053024	CSRCAS	020114
BAKPAT	002762	B121	057756	B203	070740	B76	053660	CSRFB	002214
BALANC=	000155	B122	060072	B204	071610	B77	053710	CSRFBA	002410
BANK	002106	B123	060244	B205	071614	CACHK	002700	CSRFIR	002336
BANKIN	002110	B124	060256	B206	071730	CACHOF=	104424	CSRINC	002414
BANKMO	042072	B125	060264	B207	072012	CACHON=	104423	CSRIND	002416
BANKOK	042730	B126	060514	B21	013220	CACHVE=	000114	CSRLB	002234
BAWPAF	015434	B127	060754	B210	072076	CBCSR =	104474	CSRLBA	002412
BAWPAR	015572	B13	012144	B211	072174	CB1CSR=	104475	CSRLOO	002420
BDSWIT	002500	B130	061340	B22	013270	CHECK	002404	CSRNO	002256
BEGIN	005752	B131	061352	B23	013632	CHKDIS=	104504	CSROUD	002417
BGTEST	035110	B132	061412	B24	014144	CHKGEN	040540	CSROUT	040442
BIT0	= 000001	B133	061520	B25	014512	CHKTAB	040640	CSRREL	002254
BIT1	= 000002	B134	061572	B26	015114	CHK1DI=	104505	CSRSTU=	000031
BIT10	= 002000	B135	061724	B27	016722	CKEND	073316	CSRO	002154
BIT11	= 004000	B136	061736	B3	010316	CKSWR =	104410	CSR2	002174
BIT12	= 010000	B137	061774	B30	016744	CLRCR=	104502	DATBUF	002340
BIT13	= 020000	B14	012556	B31	016750	CLR1CS=	104503	DBEMSK	002354
BIT14	= 040000	B140	062102	B32	016762	CMD11A	047774	DDISP =	177570
BIT15	= 100000	B141	062154	B33	017044	CMD11B	050164	DECMAR	043152
BIT2	= 000004	B142	062312	B34	017060	CMD11C	050240	DEENER=	104421
BIT3	= 000010	B143	062350	B35	023232	CMD19L=	000070	DETAIL	071364

DETFLA	002326	DT13	102504	EM37	105760	E121	060040	E203	071110
DETPSW	002324	DT14	102530	EM38	106025	E122	060212	E204	071670
DETR0	002306	DT16	102556	EM39	106133	E123	060460	E205	071670
DETR1	002310	DT17	102610	EM4	103311	E124	060450	E206	071760
DETR2	002312	DT18	102616	EM40	106220	E125	060432	E207	072044
DETR3	002314	DT19	102634	EM41	106272	E126	060536	E21	013244
DETR4	002316	DT2	102264	EM42	106374	E127	061006	E210	072130
DETR5	002320	DT20	102652	EM43	106477	E13	012530	E211	072330
DETSF	002322	DT21	102662	EM44	106575	E130	061702	E22	013620
DF1	102740	DT22	102700	EM45	106672	E131	061672	E23	013714
DF10	103076	DT23	102706	EM46	106773	E132	061656	E24	014264
DF11	103103	DT24	102714	EM47	107062	E133	061540	E25	014562
DF12	103114	DT25	102720	EM48	107170	E134	061646	E26	015202
DF13	103116	DT3	102302	EM49	107267	E135	062264	E27	017256
DF14	103125	DT4	102312	EM5	103357	E136	062254	E3	010364
DF2	102743	DT5	102322	EM6	103434	E137	062240	E30	017230
DF3	102767	DT6	102346	EM7	103461	E14	012716	E31	017166
DF4	103004	DT7	102362	EM8	103521	E140	062122	E32	017166
DF5	103021	DT8	102416	EM9	103556	E141	062230	E33	017152
DF6	103036	DT9	102432	ENASBE=	104506	E142	062544	E34	017122
DF7	103053	DUMMY	002304	ENA1SB=	104507	E143	062530	E35	023316
DF8	103067	DUMPCS=	000204	END	117144	E144	062436	E36	024574
DF9	103071	ECCDIS=	104470	ENERGI=	104420	E145	062520	E37	024560
DH1	107366	ECCINI=	104472	ERRMAX	002702	E146	062772	E4	010612
DH10	110145	ECC1DI=	104471	ERROR =	104000	E147	062756	E40	024746
DH11	110243	ECC1IN=	104473	ERRPC	002024	E15	013066	E41	024732
DH12	110272	EMTVEC=	000030	ERRPSW	002034	E150	062662	E42	025112
DH13	110307	EM1	103134	ERRSP	002030	E151	062744	E43	033756
DH14	110415	EM10	103633	ERRVEC=	000004	E152	063254	E44	033742
DH15	110533	EM11	103677	EVEN	002456	E153	063112	E45	033556
DH16	110655	EM12	103721	EXBANK	042422	E154	063170	E46	033726
DH17	110672	EM13	103745	EXCMD3	044726	E155	063240	E47	034002
DH18	110746	EM14	103777	EXCMD4	045236	E156	063240	E5	010612
DH19	111023	EM15	104043	EXIT	043522	E157	063576	E50	034026
DH2	107423	EM16	104111	EXIT2	043526	E16	013066	E51	040066
DH20	111037	EM17	104137	E0	005530	E160	063562	E52	040066
DH21	111056	EM18	104177	E1	010242	E161	063470	E53	040232
DH22	111063	EM19	104256	E10	011444	E162	063514	E54	040232
DH23	111111	EM2	103220	E100	054364	E163	063542	E55	040504
DH3	107446	EM20	104333	E101	054124	E164	063672	E56	040504
DH4	107502	EM21	104415	E102	054216	E165	063672	E57	041302
DH5	107507	EM22	104454	E103	054342	E166	064174	E6	010672
DH6	107615	EM23	104501	E104	054330	E167	064044	E60	041262
DH7	107664	EM24	104530	E105	054304	E17	013056	E61	041442
DH8	110006	EM25	104607	E106	054412	E170	064122	E62	041422
DH9	110106	EM26	104634	E107	054466	E171	064160	E63	042562
DIAGFL	002006	EM27	104705	E11	011432	E172	064160	E64	046456
DISPLA	002770	EM28	104774	E110	054542	E173	064466	E65	047006
DISPRE=	000174	EM29	105027	E111	054542	E174	064456	E66	047720
DISPTB	014370	EM3	103256	E112	055440	E175	064354	E67	050232
DNSLAV	055170	EM30	105111	E113	055440	E176	064442	E7	011076
DOBACK	015104	EM31	105230	E114	055746	E177	064570	E70	050552
DSWR =	177570	EM32	105330	E115	055746	E2	010364	E71	050552
DT1	102252	EM33	105435	E116	057602	E20	013160	E72	050626
DT10	102444	EM34	105543	E117	057516	E200	070402	E73	051042
DT11	102466	EM35	105624	E12	012530	E201	070474	E74	053074
DT12	102476	EM36	105711	E120	057752	E202	071110	E75	053062

E76	053734	IIII =	177777	KI2	002554	L1016	070460	L127	012700
E77	053730	IISTAC	002736	KI3	002556	L1017	070542	L13	005376
FASTCI=	177640	IISTAD	002740	KI4	002560	L102	011414	L130	012700
FATALS	002070	IISTIN	061074	KI5	002562	L1020	070602	L131	012764
FCMD10	047550	IISTOF	061174	KI6	002564	L1021	070570	L132	013046
FCMD11	047734	IISTVE	002742	KI7	002566	L1022	070576	L133	013046
FCMD12	050330	IITRAP	036764	KMAP =	104422	L1023	070624	L134	013160
FCMD13	050356	INCBNK	042772	KPFLAG	002122	L1024	070640	L135	013174
FCMD14	050400	INCMAR	043006	KSTACK	002706	L1025	070654	L136	013254
FCMD15	050420	INCPAT	042746	LAST =	157776	L1026	070670	L137	013226
FCMD16	050436	INCRPT	042746	LASTB	002262	L1027	071072	L14	005470
FCMD17	050454	INTERL	002264	LASTER	002020	L103	011416	L140	013620
FCMD18	050470	INTERO	006756	LBSO =	001057	L1030	071072	L141	013472
FCMD19	050502	INTER1	006760	LBS1 =	001055	L1031	071056	L142	013472
FCMD20	050566	INTER2	006760	LBS10=	000223	L1032	071122	L143	013472
FIELDS	043774	INTER3	006766	LBS11=	000034	L1033	071200	L144	013416
FINDBA=	000061	INTER4	006766	LBS2 =	001050	L1034	071230	L145	013434
FIRST =	060000	INTER5	006774	LBS3 =	001051	L1035	071260	L146	013452
FIRSTB	002260	INTER6	006766	LBS4 =	001032	L1036	071302	L147	013470
FIRSTP=	000010	INTER7	006774	LBS5 =	000712	L1037	071332	L15	005510
FLIPLO	002734	INTKRA	017610	LBS6 =	000714	L104	011532	L150	013604
FLIPWA	035214	INT..0	020234	LBS7 =	000033	L1040	071362	L151	013534
FLUSH	014656	INT..1	020260	LCSROU=	000055	L1041	071646	L152	013542
FSCMD0	044170	INT..2	020324	LCSRRE=	000114	L1042	072046	L153	013604
FSCMD1	044256	INT..3	020350	LCSRSA=	000112	L1043	072052	L154	013702
FSCMD2	044346	INT..4	020414	LF =	000012	L1044	072132	L155	013656
FSCMD3	044520	INT..5	020440	LKS =	177546	L1045	072136	L156	013702
FSCMD4	044764	INT..6	020504	LOADBA	002532	L1046	072316	L157	014032
FSCMD5	045274	INT..7	020530	LOADCS=	104425	L1047	072264	L16	005570
FSCMD6	046152	INT.10	020574	LOADER=	000057	L105	011574	L160	014112
FSCMD7	046160	INT.11	020574	LOADHO	002710	L1050	072316	L161	014116
FSCMD8	046510	INT.12	020616	LOADRS=	177740	L1051	072316	L162	014336
FSCMD9	047104	INT.13	020642	LOCK	002670	L1052	073034	L163	014254
FSINFL	002542	INT.14	020706	LOOP	014342	L1053	073452	L164	014172
FSPAT	045752	INT.15	020732	LOWMAP	042026	L1054	073540	L165	014174
FSSTAC	002374	INT.16	020776	LSTSS =	000000	L1055	073464	L166	014234
FS1	044052	INT.17	021022	LWDBE =	000053	L1056	073600	L167	014250
FS7FLA	002546	INVALI=	104511	LWSBE =	000051	L1057	073602	L17	005654
GBLENG=	000076	IOTVEC=	000020	LO	004426	L106	011614	L170	014304
GDSWIT	002476	ISECCO	017300	L1	004422	L107	011634	L171	014306
GETDAT	055030	J	002604	L10	005376	L11	005376	L172	014336
GETDA1	055156	KAMIKA	002010	L100	011552	L110	011654	L173	014456
GETDIS	066514	KAMITE	026006	L1000	067226	L111	011672	L174	014566
GETSIZ	017736	KDIAG =	000010	L1001	067240	L112	011714	L175	014546
GOOD	002050	KDPAR0=	172360	L1002	067276	L113	011766	L176	014546
GOOD2	002052	KDPAR7=	172376	L1003	067306	L114	012034	L177	014622
GOOD3	002054	KDPDR7=	172336	L1004	067630	L115	012130	L2	004536
GTSWR =	104407	KERNEL=	104417	L1005	070316	L116	012250	L20	005616
HEADER	002724	KERSTK=	002000	L1006	070234	L117	012372	L200	014622
HIACBA	002514	KIPAR0=	172340	L1007	070234	L12	005370	L201	014656
HIADRS=	177742	KIPAR4=	172350	L101	011416	L120	012422	L202	014734
HIBANK	002512	KIPAR5=	172352	L1010	070274	L121	012416	L203	014744
HIPAT	042762	KIPAR6=	172354	L1011	070262	L122	012420	L204	015100
HT =	000011	KIPAR7=	172356	L1012	070270	L123	012514	L205	015166
HUNGMS	002572	KIPDR0=	172300	L1013	070360	L124	012474	L206	015306
HUNGTI	002574	KIO	002550	L1014	070366	L125	012474	L207	015422
I	002602	KI1	002552	L1015	070452	L126	012550	L21	005634

SYMBOL TABLE	
L210	015560
L211	015716
L212	016054
L213	016234
L214	016372
L215	016552
L216	016674
L217	017230
L22	005636
L220	017152
L221	017034
L222	017152
L223	017134
L224	017214
L225	017366
L226	017372
L227	017724
L23	005714
L230	020016
L231	020030
L232	020072
L233	020110
L234	020224
L235	020306
L236	020314
L237	020376
L24	005774
L240	020404
L241	020466
L242	020474
L243	020556
L244	020564
L245	020614
L246	020670
L247	020676
L25	006010
L250	020760
L251	020766
L252	021050
L253	021056
L254	021174
L255	021216
L256	021232
L257	021250
L26	006160
L260	021254
L261	021256
L262	021322
L263	021344
L264	021372
L265	021372
L266	021522
L267	021566
L27	006160
L270	022522
L271	022536
L272	022546
L273	022546
L274	022600
L275	022614
L276	022624
L277	022624
L3	005046
L30	006242
L300	022662
L301	022672
L302	022672
L303	022732
L304	022746
L305	022756
L306	022756
L307	023010
L31	006356
L310	023024
L311	023034
L312	023034
L313	023066
L314	023102
L315	023112
L316	023112
L317	023172
L32	006406
L320	023202
L321	023202
L322	023300
L323	023270
L324	023400
L325	023402
L326	023432
L327	023436
L33	006476
L330	023466
L331	023472
L332	023526
L333	023532
L334	023562
L335	023566
L336	023744
L337	023776
L34	006564
L340	024042
L341	024206
L342	024216
L343	024216
L344	024544
L345	024530
L346	024544
L347	024610
L35	006614
L350	024716
L351	024702
L352	024716
L353	025076
L354	025076
L355	025200
L356	025152
L357	025224
L36	006630
L360	025350
L361	025540
L362	025756
L363	025770
L364	026030
L365	026036
L366	026042
L367	033434
L37	006712
L370	033444
L371	033556
L372	033506
L373	033520
L374	033536
L375	033544
L376	033570
L377	033726
L4	005144
L40	006734
L400	033626
L401	033640
L402	033672
L403	033656
L404	033670
L405	033714
L406	033702
L407	033714
L41	007152
L410	034026
L411	035504
L412	036150
L413	036220
L414	036246
L415	036240
L416	036246
L417	037136
L42	007156
L420	037142
L421	037202
L422	037206
L423	037424
L424	037432
L425	037510
L426	037544
L427	037634
L43	007160
L430	037670
L431	037730
L432	037772
L433	040044
L434	040044
L435	040106
L436	040114
L437	040210
L44	007700
L440	040210
L441	040252
L442	040260
L443	040376
L444	040432
L445	040462
L446	041114
L447	041144
L45	007714
L450	041246
L451	041276
L452	041410
L453	041410
L454	041410
L455	041410
L456	041410
L457	041436
L46	010226
L460	041534
L461	042550
L462	042720
L463	043026
L464	043114
L465	043134
L466	043230
L467	043232
L47	010226
L470	043256
L471	043264
L472	043410
L473	043410
L474	043364
L475	043410
L476	043444
L477	043542
L5	005222
L50	010166
L500	043546
L501	043554
L502	043574
L503	043606
L504	044016
L505	044026
L506	044160
L507	044220
L51	010166
L510	044224
L511	044246
L512	044252
L513	045416
L514	045456
L515	045510
L516	045512
L517	045536
L52	010166
L520	045574
L521	045714
L522	045716
L523	046232
L524	046234
L525	046442
L526	046602
L527	046604
L53	010202
L530	046630
L531	046644
L532	046646
L533	046772
L534	047154
L535	047156
L536	047272
L537	047274
L54	010300
L540	047320
L541	047720
L542	047704
L543	047654
L544	050026
L545	050030
L546	050054
L547	050070
L55	010304
L550	050072
L551	050216
L552	050534
L553	050536
L554	050724
L555	050726
L556	051030
L557	050774
L56	010306
L560	051010
L561	051026
L562	051056
L563	051170
L564	051414
L565	051514
L566	051644
L567	051656
L57	010350
L570	052414
L571	052442
L572	052510
L573	052522
L574	053076
L575	053202
L576	053200
L577	053232
L6	005246
L60	010346
L600	053232
L601	053734

SYMBOL TABLE

L602	053722	L665	061636	L747	065762	MHALT2	005770	MSG024	113316
L603	053726	L666	061630	L75	011306	MHALT3	012760	MSG025	113340
L604	054350	L667	062240	L750	065774	MINDEX	002666	MSG026	113371
L605	054200	L67	011110	L751	066006	MJPAT	021600	MSG027	113403
L606	054272	L670	062040	L752	066030	MJSIZE	002466	MSG028	113420
L607	054314	L671	062066	L753	066076	MJTEST	021512	MSG029	113434
L61	010576	L672	062220	L754	066156	MKCONT	016664	MSG030	113454
L610	054450	L673	062212	L755	066172	MKCSRS	002334	MSG031	113473
L611	054526	L674	062530	L756	066174	MKCSRT	020124	MSG032	113533
L612	054532	L675	062402	L757	066250	MKFLAG	002126	MSG033	113552
L613	054626	L676	062502	L76	011320	MKPAT	021402	MSG034	113571
L614	054642	L677	062756	L760	066316	MKSIZE	002470	MSG035	113707
L615	054646	L7	005316	L761	066312	MKTEST	021222	MSG036	113712
L616	054664	L70	011224	L762	066316	MK11AD=	172100	MSG037	113731
L617	055024	L700	062726	L763	066350	MMR0 =	177572	MSG038	113750
L62	010672	L701	063010	L764	066360	MMR1 =	177574	MSG039	113766
L620	055174	L702	063240	L765	066624	MMR2 =	177576	MSG040	114010
L621	055312	L703	063152	L766	066746	MMR3 =	172516	MSG041	114055
L622	055416	L704	063224	L767	066746	MMTRAP	036740	MSG042	114102
L623	055550	L705	063230	L77	011332	MMVEC =	000250	MSG043	114123
L624	055724	L706	063272	L770	067116	MOVELO=	000124	MSG044	114150
L625	057152	L707	063306	L771	067050	MP1	012070	MSG045	114175
L626	057136	L71	011236	L772	067074	MP2	012224	MSG046	114231
L627	057152	L710	063562	L773	067076	MP3	012716	MSG047	114264
L63	011076	L711	063562	L774	067116	MP4	012552	MSG048	114303
L630	057314	L712	063454	L775	067300	MP5	012764	MSG049	114343
L631	057202	L713	063450	L776	067164	MRUNSE	053640	MSG050	114375
L632	057216	L714	063454	L777	067276	MSEEDH	002720	MSG051	114503
L633	057232	L715	063662	MAINT =	177750	MSEEDL	002722	MSG052	114523
L634	057326	L716	063662	MAPH0 =	170202	MSGA34	113625	MSG053	114554
L635	057604	L717	063722	MAPH36=	170372	MSGA87	115753	MSG054	114572
L636	057564	L72	011250	MAPH37=	170376	MSGA98	116242	MSG055	114642
L637	060040	L720	064160	MAPL0 =	170200	MSGB34	113642	MSG056	114663
L64	011076	L721	064104	MAPL1 =	170204	MSG000	111176	MSG057	114716
L640	057724	L722	064150	MAPL36=	170370	MSG001	111254	MSG058	114722
L641	060022	L723	064224	MAPPER	040700	MSG002	111336	MSG059	114766
L642	060174	L724	064230	MARGIN	002112	MSG003	111413	MSG060	115016
L643	060174	L725	064442	MASTER=	000020	MSG004	111520	MSG061	115060
L644	060416	L726	064336	MAST01	057372	MSG005	111626	MSG062	115067
L645	060366	L727	064344	MAST02	060046	MSG006	111640	MSG063	115107
L646	060554	L73	011262	MAST03	060220	MSG007	111675	MSG064	115120
L647	060572	L730	064424	MAST04	061322	MSG008	111707	MSG065	115130
L65	011060	L731	064552	MAST05	061706	MSG009	111721	MSG066	115142
L650	060610	L732	065346	MAST06	062270	MSG010	111733	MSG067	115225
L651	060626	L733	065360	MAST07	062550	MSG011	111745	MSG068	115234
L652	060644	L734	065376	MAST10	062776	MSG012	112033	MSG069	115241
L653	060662	L735	065400	MAST11	063260	MSG013	112130	MSG070	115246
L654	060700	L736	065420	MAST12	063732	MSG014	112132	MSG071	115271
L655	060716	L737	065432	MAST13	064200	MSG015	112134	MSG072	115304
L656	060734	L74	011274	MAST14	064204	MSG016	112136	MSG073	115317
L657	060746	L740	065450	MAST15	064472	MSG017	112150	MSG074	115336
L66	011060	L741	065452	MASWR	002674	MSG018	112161	MSG075	115341
L660	061162	L742	065466	MDISPL	002676	MSG019	112164	MSG076	115373
L661	061162	L743	065676	MEMDON	014410	MSG020	112170	MSG077	115414
L662	061656	L744	065704	MEMERR=	177744	MSG021	112224	MSG078	115430
L663	061456	L745	065732	MEMSIZ=	000004	MSG022	113252	MSG079	115474
L664	061504	L746	065750	MHALT1	004532	MSG023	113274	MSG080	115520

SYMBOL TABLE

MSG081	115560	MTPC26	034646	MT0027	024370	O.CSR2	075575	O.STM =	000340
MSG082	115577	MTPD03	026630	MT0030	024754	O.CT	075730	O.SVR	100360
MSG084	115617	MTPD21	033354	MT0031	025214	O.CTLC	076162	O.SVTT	100500
MSG085	115661	MTPD25	034412	MT0033	025340	O.CTLE	076242	O.T	075572
MSG086	115706	MTPD26	034666	MT0034	025530	O.CTLF	076210	O.TBIT	077644
MSG087	115745	MTPE25	034434	MT0035	025652	O.C1	100000	O.TBT =	000020
MSG088	115760	MTP000	026376	MT020X	023604	O.DCD	076520	O.TCL2	076442
MSG089	115776	MTP001	026422	MT020Y	023472	O.DCDA	077144	O.TCSR=	177564
MSG090	116020	MTP002	026454	MT020Z	023322	O.DCD1	076550	O.TDB =	177566
MSG091	116034	MTP005	026724	MT0999	025772	O.DCD3	076530	O.TL	101345
MSG092	116046	MTP006	026760	MULHLT	056150	O.DOT	075562	O.TRTC	101370
MSG093	116062	MTP007	027150	MULTIE	060746	O.ERR	076510	O.TVEC=	000014
MSG094	116070	MTP010	027250	MULTIO	002004	O.ERR1	077572	O.TYPE	101126
MSG095	116077	MTP011	027356	MULTIP	003014	O.ERR2	077164	O.TYP1	101004
MSG096	116106	MTP012	027766	MULTIU	056142	O.ERR3	077150	O.UIN	075752
MSG097	116137	MTP013	030344	MUT	002116	O.FIL	075576	O.UPC	075700
MSG098	116234	MTP014	030716	M13BAL	054472	O.FTYP	100730	O.URO	075662
MSG099	116245	MTP015	031312	NEMCNT	002072	O.GET	101006	O.USP	075676
MSG100	116256	MTP016	031700	NEWBAN	002400	O.GO	077576	O.UST	075702
MSG101	116305	MTP017	032302	NEWKER	042322	O.GOGO	077624	O.WRD	076760
MSG102	116335	MTP020	032360	NEWLOA	042370	O.GO2	077676	O.WRD1	077036
MSG103	116364	MTP022	033404	NOBALA=	000164	O.ID	101300	O.WST	101202
MSG104	116421	MTP025	034152	NOCLOC	002104	O.IDND=	101315	O.XXX	075564
MSG105	116423	MTP030	034704	NOECCF	002474	O.LG =	000020	PADDRE	002042
MSG106	116527	MTP031	034714	NOERRO	002530	O.LGCH	101321	PAFBAF	015730
MSG107	116545	MTP033	035054	NOFSMO	002526	O.MIN	076502	PAFBAW	016066
MSG108	116622	MTP034	035106	NOIIST	002756	O.MINS	075603	PARBAF	016246
MSG109	116664	MTP20A	032360	NONEM	002102	O.ODT	075774	PARBAW	016404
MSG110	116737	MTP20B	032376	NONEXI	036646	O.OFST	077466	PARCNT	002074
MSG111	117003	MTP20C	032430	NOPAR	002100	O.OLD	077154	PARITY	036454
MSG112	117035	MTP20D	032462	NOSCOP	002540	O.OP1	077160	PARTHE	002372
MSG113	117103	MTP20E	032514	NOTAB	002464	O.OP2	077220	PARVEC=	000114
MSG114	117113	MTP20F	032546	NULLFL	002436	O.OP2A	077226	PASFLG	002362
MSG115	117123	MTV020	023320	OLDCAC	002370	O.ORAB	076364	PATERR	002076
MSG116	117133	MT0000	021704	OLDMAR	002376	O.ORPC	076342	PATPLU	005472
MST11A	063602	MT0001	021740	ONCE	002732	O.ORRB	076374	PATTER	002120
MTA030	024760	MT0002	022000	ONES	002726	O.P	075573	PCBUMP	002406
MTEST	016564	MT0003	022050	O.ADR1	075706	O.PRI	075704	PCONF1	035402
MTPA03	026506	MT0004	022216	O.BACK	077302	O.PROC	077732	PCONFS	035662
MTPA04	026650	MT0005	022300	O.BD	101366	O.PR1	077764	PCONF1	035572
MTPA20	032572	MT0006	022360	O.BIAS	075604	O.RALL	077430	PCONF2	035630
MTPA21	033234	MT0007	022414	O.BKP =	000016	O.RCSR=	177560	PDP110	036752
MTPA24	034060	MT0010	022456	O.BKPT	077324	O.RDB =	177562	PENDBO	002330
MTPA25	034464	MT0011	022512	O.BK1	100060	O.REGT	076270	PERA05	064732
MTPA26	034576	MT0012	022570	O.BRK	100042	O.REM	100602	PERBNK	065564
MTPA32	034772	MT0013	022646	O.BW	075566	O.RORA	101272	PERECC	065650
MTPB03	026550	MT0014	022722	O.BYT	076772	O.RRST	101172	PERRAB	065402
MTPB04	026704	MT0015	023000	O.CAD	075560	O.RSB	100450	PERRAW	065330
MTPB21	033264	MT0016	023056	O.CADV	100634	O.RSR	100416	PERRA3	054652
MTPB24	034120	MT0017	023134	O.CLGT=	000024	O.RSTT	100534	PERRA7	065454
MTPB25	034504	MT0020	023156	O.CLSE	101214	O.S	075571	PERR01=	104427
MTPB26	034612	MT0021	023612	O.CMFD	075600	O.SCAN	076554	PERR02=	104430
MTPB32	035020	MT0022	023734	O.CR	101316	O.SCRN	075602	PERR03=	104431
MTPC03	026610	MT0023	023766	O.CRF	077140	O.SEMI	076742	PERR04=	104432
MTPC21	033320	MT0024	024032	O.CRLS	101246	O.SEQ	075570	PERR05	064726
MTPC24	034134	MT0025	024172	O.CRS	101254	O.SMFD	075601	PERR06	064754
MTPC25	034536	MT0026	024240	O.CSR1	075574	O.SNGL	076446	PERR07=	104433

PERR10=	104434	RESTAR	002752	START1	000300	TAG72\$	067656	TST1	006260
PERR11=	104435	RESVEC=	000010	START2	000310	TAG73\$	067722	TST2	007004
PERR12=	104436	RLFLAG	002134	START3	000200	TAG74\$	067730	TST3	007160
PERR13=	104437	RRFLAG	002132	START4	000322	TAG75\$	067742	TST4	010310
PERR14=	104440	RTNVAL=	%000000	STOPCP	011720	TAG76\$	067754	TST5	014342
PERR15=	104441	RUNCOU=	000171	STOPOK	002520	TAG77\$	067776	TST6	014414
PERR16=	104442	SAMESA=	000015	STRIPE	002460	TAG78\$	070000	TST7	014456
PERR17=	104443	SAVREG=	104415	STRTDI	002364	TAG79\$	070006	TWCSR	002600
PERR20=	104444	SBEC SR	002150	SUBAAA	005532	TAG80\$	070066	TYPDS =	104405
PERR21=	104445	SBFLA	002022	SUBAAB	005200	TAG81\$	070100	TYPEIT=	104401
PERR22=	104446	SBEMSK	002350	SUBAAQ	007720	TAG82\$	070122	TYPOC =	104402
PERR23=	104447	SBETES	017374	SUBAAR	013160	TAG83\$	070324	TYPOS =	104403
PERR24=	104450	SCOPE =	000004	SUBAAS	007002	TAG84\$	070410	TYP SO =	000000
PERR25=	104451	SDPAR0=	172260	SUBAAT	011334	TAG85\$	070502	TYP S1 =	000000
PERR26=	104452	SDPAR7=	172276	SUBAAU	014602	TAG86\$	070612	TYP S10=	000000
PERR27=	104453	SDPDR7=	172236	SUCCESS	002424	TAG9\$	007232	TYP S11=	000002
PERR30=	104454	SECOND=	011661	SUPDOA	002360	TASK	002616	TYP S2 =	000000
PERR31=	104455	SEEDHI	002714	SUPDO1	026042	TASKDO	050726	TYP S3 =	000000
PERR32=	104456	SEEDLO	002716	SUPDO2	026056	TASKRE	050710	TYP S4 =	000000
PERR33=	104457	SELONL	002002	SUPDO3	026220	TASK00	051234	TYP S5 =	000000
PERR34=	104460	SETMAR	043212	SUPDO4	026234	TASK01	051236	TYP S6 =	000000
PERR35=	104461	SETPAT	042762	SUPDR0	002266	TASK02	051330	TYP S7 =	000002
PERR36=	104462	SHALT	004420	SUPDR1	002270	TASK03	051430	T12A	031700
PERR37=	104463	SHUTUP	043554	SUPDR2	002272	TASK04	051664	T12B	031722
PERR40=	104464	SIDE	002422	SUPDR3	002274	TASK05	051774	UDPAR0=	177660
PERR41=	104465	SIDOK	021066	SUPDR4	002276	TASK06	052300	UDPAR1=	177662
PERR42=	104466	SIPAR0=	172240	SUPDR5	002300	TASK07	052326	UDPAR7=	177676
PERR43=	104467	SIPAR3=	172246	SUPDR6	002302	TASK10	052336	UDPDR7=	177636
PERXOR	065540	SIPDR0=	172200	SUPLIM	057372	TASK11	052536	UIPAR0=	177640
PFLAG	002130	SIZE =	040000	SUPM13	053752	TASK12	052644	UIPAR2=	177644
PHYADD	002044	SIZELO=	177760	SUPSTK=	000740	TASK13	052726	UIPAR3=	177646
PORTCO	002646	SKIPKA	002012	SWAPAT	002764	TASK14	053276	UIPAR4=	177650
PORTDI	002606	SLAVEM	061244	SWR	002766	TASK15	053310	UIPAR5=	177652
PROTEC=	000002	SLAVEN	075774	SWREG =	000176	TASK16	053316	UIPAR6=	177654
PSW =	177776	SLAVER	002570	SW0 =	000001	TASK17	053334	UIPAR7=	177656
PTYBUF	002636	SLAVES	002730	SW1 =	000002	TASK20	053422	UIPDR0=	177600
PTYFLA	002626	SLAVE1=	000040	SW10 =	002000	TASK7A	052344	UNITOP	002516
PWLOCK	002760	SLAVE2=	000100	SW11 =	004000	TBG4\$	026314	UNRELO	041604
PWRVEC=	000024	SLAVE3=	000200	SW12 =	010000	TCFIG1	036012	UPPFLG	002363
QUE	072154	SLAVUP	061216	SW13 =	020000	TCONF I	035664	UP2	056154
QUE1	072222	SLFLAG	002656	SW14 =	040000	TEMP	002534	USERMA	042240
QUICK	002536	SLPAT	051570	SW15 =	100000	TESTAD	002506	USESTK=	000700
QVFLAG	002440	SLPAT0	051530	SW2 =	000004	TIME	002434	USP =	%000006
RANODD	034626	SLPAT1	051534	SW3 =	000010	TIMEOU	036726	WAITST	012034
RDCHR =	104411	SLPAT2	051542	SW4 =	000020	TKVEC =	000060	WAIT5	063674
RDDEC =	104414	SLPAT3	051550	SW5 =	000040	TMFLAG	002140	WAKEUP	054544
RDLIN =	104412	SLPAT4	051556	SW6 =	000100	TOOMAN	002472	WARN1	007320
RDOCT =	104413	SLPAT5	051564	SW7 =	000200	TRAPVE=	000034	WARN2	026522
READCS=	104426	SOBK	002704	SW8 =	000400	TRT =	000003	WARN3	026536
READON	002504	SOBLEN=	000056	SW9 =	001000	TSK05A	052240	WARN4	026562
REALPA	002366	SOFTPA	002744	SYSSIZ	011210	TSK13A	053100	WARN5	026576
REFRES	033760	SOURCE	002402	SYSTID=	177764	TSK13B	053136	WARN6	035364
REFSUB	034030	SPECIA	002432	TAG2\$	007366	TSK20A	053602	WARN7	024416
REGCOP	035204	SSP =	%000006	TAG3\$	007412	TSTBAN	007606	WASDBE=	104500
RELOCA	041130	ST =	177776	TAG4\$	026136	TSTDAT	002344	WASSBE=	104476
RELOC1	041456	STACK =	002000	TAG70\$	067636	TSTRD1	037434	WAS1DB=	104501
RESREG=	104416	START	004352	TAG71\$	067646	TSTREA=	104510	WAS1SB=	104477

SYMBOL TABLE

WHICHC	050630	\$DDW5	075232	\$IS	= 000001	\$PER01	064574	\$SV\$	= 000000
WHOBX	070672	\$DDW6	075234	\$KERNE	037020	\$PER02	064622	\$SWR	= 163000
WHODUN=	000202	\$DEENE	037040	\$KMAP	041016	\$PER03	064650	\$SWREG	075156
WOOPEN	056512	\$DEVCT	075144	\$KS	= 000204	\$PER04	064700	\$T	= 001060
WOOPS	056160	\$DEVVM	075212	\$L	= 000212	\$PER07	064762	\$TESTN	075140
WOOPSA	056542	\$DIDDO=	000000	\$LF	003011	\$PER10	065004	\$TKB	002774
WOOPUP	056336	\$DOAGA	015040	\$LL	= 000210	\$PER11	065034	\$TKS	002772
WORST	002712	\$DOAGN	015010	\$LOADC	037076	\$PER12	065054	\$TN	= 000010
WRITED	002502	\$DOWN	055532	\$LOAD1	037146	\$PER13	065076	\$TPB	003000
XXDPCH	002446	\$DTBL	073002	\$LPADR	002746	\$PER14	065116	\$TPFLG	002452
ZERO	002426	\$ECCDI	037466	\$LPERR	002750	\$PER15	065140	\$TPS	002776
ZEROS	002430	\$ECCIN	037554	\$LS	= 000000	\$PER16	065162	\$TRAP	075252
\$APTHD	075236	\$ECC1D	037522	\$MADR1	075166	\$PER17	065202	\$TRAP2	075274
\$AUTO	002066	\$ECC1I	037574	\$MADR2	075172	\$PER20	065220	\$TRPAD	075314
\$BANK	002015	\$ENASB	037612	\$MADR3	075176	\$PER21	065236	\$TSTM	075242
\$BASE	075210	\$ENAS	037646	\$MADR4	075202	\$PER22	065256	\$TSTRD	037310
\$BELL	003003	\$ENDAD	015000	\$MAIL	075134	\$PER23	065274	\$TTYIN	074200
\$CACHF	037066	\$ENERG	037030	\$MAMS1	075164	\$PER24	065312	\$TYPDS	072576
\$CACHN	037050	\$ENV	075154	\$MAMS2	075170	\$PER25	054570	\$TYPE	056752
\$CBCSR	037700	\$ENVM	075155	\$MAMS3	075174	\$PER26	065502	\$TYPEC	057076
\$CB1CS	037742	\$EOP	014662	\$MAMS4	075200	\$PER27	065522	\$TYPEX	057360
\$CDW1	075214	\$ERFLG	002016	\$MBADR	075240	\$PER30	054762	\$TYPOC	072374
\$CDW2	075216	\$ERROR	066662	\$MNEW	074254	\$PER31	065720	\$TYPON	072410
\$CHARC	057356	\$ERRTB	101372	\$MSGAD	075150	\$PER32	066016	\$TYPOS	072350
\$CHKDI	040354	\$ERRTY	067324	\$MSGLG	075152	\$PER33	066064	\$T1	= 000000
\$CHK1D	040410	\$ERTTL	002754	\$MSGTY	075134	\$PER34	066144	\$T2	= 001057
\$CKSWR	073022	\$ESCAP	002454	\$MSWR	074243	\$PER35	066176	\$UNIT	075146
\$CLRCS	040322	\$ETABL	075154	\$MTYP1	075165	\$PWRDN	055160	\$UNITM	075246
\$CLR1C	040340	\$ETEND	075236	\$MTYP2	075171	\$PWRUP	055536	\$USWR	075160
\$CMTAG	002000	\$EXHAL	043546	\$MTYP3	075175	\$QUES	003007	\$VECT1	075204
\$CMTGE	002700	\$ES	= 000001	\$MTYP4	075201	\$R	= 177777	\$VECT2	075206
\$CNTLC	074224	\$FATAL	075136	\$NOTRA	075306	\$RAND	074720	\$WASDB	040146
\$CNTLG	074236	\$FILLC	003002	\$NULL	002450	\$RDCHR	073602	\$WASSB	040002
\$CNTLU	074231	\$FILLS	002451	\$NWTST=	000001	\$RDDEC	074436	\$WAS1D	040262
\$CPUOP	075162	\$FS	= 000000	\$OCNT	072572	\$RDLIN	073722	\$WAS1S	040116
\$CRLF	003010	\$GTSWR	073160	\$OCTVL	075116	\$RDOCT	074266	\$XTSTR	066370
\$DBLK	073012	\$HALT	067136	\$OCT8	= 075122	\$READC	037220	\$YS	= 000000
\$DB20	075014	\$HALT2	075312	\$OMODE	072574	\$RESRE	074662	\$ZAP42	014760
\$DDW0	075220	\$HIBTS	075236	\$OVER	066502	\$SAVRE	074624	\$Z\$	= 000000
\$DDW1	075222	\$HIOCT	074434	\$OS	= 000000	\$SAVR6	056140	\$SS	= 000000
\$DDW2	075224	\$ILLUP	056134	\$PASS	075142	\$SCOPE	066232	\$ST	= 001040
\$DDW3	075226	\$INVAL	040510	\$PASTM	075244	\$STN	= 000001	\$STT	= 001042
\$DDW4	075230	\$ITEMB	002017	\$PATMA	002014	\$SVLAD	066466	\$OFILL	072573

. ABS. 117144 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 30717 WORDS (120 PAGES)
 DYNAMIC MEMORY: 21872 WORDS (84 PAGES)
 ELAPSED TIME: 01:51:56
 .CEMKAB.LS2/CRF:SYM/-SP=CEMKAB.BIG,CEMKAB.MLB/ML,CEMKAB.MA1,CEMKAB.MA2

CEMKAB		CREATED BY MACRC ON 3-AUG-79 AT 17:42		PAGE 1		H 4		SEQ 1081			
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	CREF	V01						
ABORTF		002142	#39-1166	*282-6181	394-570	396-607	*396-625	449-1990	449-1993	449-2002	449-2005
ACFLAG		002124	#39-1159	72-2036	76-2113	*78-2125	78-2126	82-2210	95-2576	95-2590	103-2754
			110-2847	112-2863	114-2897	116-2931	118-2972	121-3018	123-3060	125-3109	127-3151
			132-3240	170-4136	170-4159	172-4190	314-6654	*325-6878	*325-6889	352-7479	354-7533
			358-7682	390-466	413-1076	421-1252	424-1317	426-1377	428-1425	430-1467	434-1534
			438-1623	440-1684							
ACK		002672	#41-1308	*46-1404	46-1410	*90-2385	91-2424	91-2431	*91-2440	*91-2479	
ACTFLA		002442	#39-1239	*53-1534	66-1826	70-2022	150-3655	150-3668	159-3888	159-3897	159-3905
			159-3915	161-3926	161-3935	163-3950	163-3989	163-3995	163-4002	163-4008	166-4091
			180-4326	336-7055	461-2263						
			103-2751	#103-2768							
ADDPAR		014570	#39-1134	*66-1820	*80-2174	*207-4742	*207-4756	*210-4814	*210-4828	*213-4901	*284-6213
ADDRES		002040	*386-408	*386-416	*394-569	*394-569	*394-591	*396-606	*396-606	396-609	396-619
			*446-1883	*446-1884	*446-1889	*446-1890	*446-1895	*446-1896	*446-1901	*446-1902	*446-1909
			*446-1917	*446-1922	*446-1922	*446-1927	*446-1932	*448-1938	*448-1943	*448-1948	*448-1953
			*448-1958	*448-1963	*448-1968	*448-1973	*448-1978	*448-1983	*453-2053	*455-2083	*455-2092
			469-2434	478-2631	478-2633	478-2633	478-2635	478-2635	478-2638	478-2640	478-2640
			478-2642	478-2642	561-4700	561-4703	561-4706	561-4708	561-4713	561-4721	
ADD..0		071152	#476-2604	#478-2629							
ADD..1		071162	476-2605	#478-2631							
ADD..2		071202	476-2606	#478-2633							
ADD..3		071232	476-2607	#478-2635							
ADD..4		071262	476-2608	#478-2637							
ADD..5		071264	476-2609	#478-2638							
ADD..6		071304	476-2610	#478-2640							
ADD..7		071334	476-2611	#478-2642							
ALLCPU		002510	#39-1261	*91-2496	95-2572	390-463	*415-1113	*415-1115	*415-1116	421-1249	424-1314
			426-1374	428-1422	430-1464	434-1529	438-1620	440-1681			
APTCOP		002522	#39-1266	*99-2686	99-2694	561-4705					
APTDOW		043640	53-1529	#336-7085	336-7088	*336-7090					
APTFLA		002444	#39-1240	*53-1528	55-1547	99-2665	159-3888	159-3897	159-3905	159-3915	161-3926
			161-3935	163-3950	166-4091	180-4326	336-7055	461-2263			
			93-2558	#108-2832	108-2838						
APTHAN		015050	#336-7092								
APTHLT		043706	#39-1267	*99-2689	99-2694	561-4705					
APTMOS		002524	#39-1275	*53-1524	99-2665						
APTSIZ		002544	155-3840	155-3851	157-3870	165-4025	166-4070	#271-5940	332-7015		
BACKGN		035344	#39-1140	*80-2176	*91-2447	*91-2484	*99-2703	*386-407	*386-415	*390-484	*390-530
BAD		002056	*394-572	*394-594	*396-609	*396-623	*409-975	*409-1018	*411-1050	*413-1084	*413-1098
			*421-1267	*421-1289	*424-1331	*424-1353	*426-1403	*428-1444	*430-1480	*436-1591	*436-1605
			*438-1638	*440-1703	*442-1728	*446-1885	*446-1891	*446-1898	*446-1903	*446-1908	*446-1918
			*446-1923	*446-1928	*446-1933	*448-1939	*448-1944	*448-1949	*448-1954	*448-1959	*448-1964
			*448-1969	*448-1974	*448-1979	*448-1984	451-2031	*453-2054	*455-2084	*455-2093	*455-2094
			*455-2113	475-2567	475-2570	475-2573	475-2576	561-4700	561-4706	561-4708	561-4716
			561-4720	561-4721	561-4723	561-4725					
BADCSR		002000	#39-1115	471-2475	*476-2597	*476-2598					
BADPC		002026	#39-1129	*284-6240	394-583	449-1991	449-2003	451-2014	453-2064	455-2082	455-2091
			455-2102	460-2234	460-2235	*460-2239					
BADPSW		002036	#39-1133	*284-6241	455-2113	460-2238					
BADSP		002032	#39-1131	*284-6238	*284-6239	460-2237					
BADSTA		036770	282-6182	282-6196	284-6223	284-6227	284-6230	#284-6237	394-583	446-1911	449-1991
			449-2003	451-2014	453-2064	455-2082	455-2091	455-2102	455-2112		

K 4

CEMKAB CREATED BY MACRO ON 3-AUG-79 AT 17:42 PAGE 4 SEQ 1084
SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES	91-2508	93-2530	132-3257	136-3372	136-3379	147-3645	183-4361	186-4395	249-5504
		91-2508	93-2530	132-3257	136-3372	136-3379	147-3645	183-4361	186-4395	249-5504	
		360-7730	398-645								
CACHVE	= 000114	#12-330	12-331								
CBCSR	= 104474	#10-236									
CB1CSR	= 104475	#10-237	147-3629	207-4716	210-4802	213-4893	218-4966	221-5044	224-5121	261-5709	
CHECK	002404	#39-1223	*147-3627	294-6389	294-6396	*306-6551	561-4713				
CHKDIS	= 104504	#10-244	70-1971	70-1980	70-1988	70-1996					
CHKGEN	040540	207-4705	210-4793	213-4882	218-4954	221-5031	224-5109	259-5649	#306-6528		
CHKTAB	040640	306-6538	#308-6556								
CHK1DI	= 104505	#10-245	237-5291	237-5312	237-5320	239-5335	239-5356	239-5365	241-5381	241-5400	
		394-585									
CKEND	073316	488-2953	488-2955	488-2957	#488-2986						
CKSWR	= 104410	#10-175	457-2134	460-2214	461-2255	484-2755					
CLRCSR	= 104502	#10-242	70-1974	70-1983	70-1991	70-1999	70-2013	#150-3658	170-4128	172-4199	
		#349-7386	#352-7453	#354-7514	#356-7557	#358-7663	370-132	376-254			
CLR1CS	= 104503	#10-243	134-3293	136-3341	136-3359	136-3375	147-3634	147-3643	163-3981	163-4014	
		207-4688	207-4733	207-4746	207-4777	210-4785	210-4804	210-4855	213-4862	213-4876	
		213-4895	215-4924	218-4948	218-4970	218-5005	221-5025	221-5046	221-5082	224-5125	
		226-5162	241-5411	259-5684	394-590						
CMD11A	047774	#358-7649	358-7653	358-7657							
CMD11B	050164	#358-7680	358-7688								
CMD11C	050240	358-7668	#358-7690								
CMD20A	050576	362-7746	#362-7768								
CMD5A	045456	#349-7349	349-7353	349-7357							
CMD5B	045666	#349-7381	349-7394								
CMD5C	046052	349-7365	349-7369	#349-7429							
CMD7A	046336	352-7463	#352-7472								
CMD8A	046550	#354-7500	354-7504	354-7508							
CMD8B	046740	#354-7531	354-7539								
CMD8C	047014	354-7519	#354-7541								
CMD9A	047240	#356-7573	356-7577	356-7581							
CMD9B	047456	356-7586	#356-7607								
CMD9C	047464	356-7592	#356-7610								
CMD9LO	047444	356-7585	#356-7603	356-7605	356-7607						
CONFGE	002576	#39-1288	*68-1910	*68-1929	95-2576	*95-2577	95-2617	*95-2618	97-2659		
CONFGT	005666	65-1705	#65-1767								
CONFG1	006304	#65-1707	65-1773								
CONFG3	006520	#65-1742	65-1765								
CONFG5	006712	65-1736	65-1766	#65-1772							
CONFIE	004336	#45-1398	51-1486								
CONF IG	003016	#45-1395	51-1484	*57-1579	*57-1580	*65-1743	*65-1744	*65-1745	*65-1747	*65-1752	
		*65-1756	*65-1761	*66-1822	68-1859	*68-1909	*68-1911	*68-1928	*68-1932	70-1965	
		70-1966	70-1966	*70-2008	*82-2206	*82-2221	95-2571	95-2579	95-2580	95-2581	
		95-2582	95-2591	95-2593	95-2610	97-2630	97-2631	97-2632	97-2632	178-4309	
		*178-4315	277-6016	277-6030	279-6051	279-6058	279-6090	279-6096	279-6113	279-6136	
		279-6140	*303-6522	*314-6661	*314-6662	314-6673	314-6674	314-6675	314-6676	314-6677	
		314-6677	314-6678	314-6678	*314-6692	*318-6729	*318-6733	*318-6737	325-6883	325-6888	
		325-6893	325-6898	325-6904	325-6910	349-7334	356-7628	356-7634	*362-7756	*362-7769	
		*362-7770	*413-1093	*413-1094	*415-1170	*415-1177	434-1533	*453-2042	*453-2044	*453-2046	
		453-2047									
CONF SP	006754	*65-1706	65-1767	#65-1777							
CONTF L	002332	#39-1200	*100-2711	130-3203	*132-3278						

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES								
CONTP	073112	#488-2948	*509-3660	511-3708						
CONTRL	= 177746	#12-338	*46-1426	65-1704	*65-1775	*286-6262	*286-6263	*286-6268	289-6289	*289-6297
		291-6313	*291-6317	291-6331	*291-6334	339-7133	*341-7182	*386-423	*386-429	*396-621
		*396-627	398-644	*402-780	408-909	*408-945	484-2756	*484-2784	561-4714	
CONTS	073544	488-2951	#490-3033							
CONTS1	073052	#488-2938	490-3040							
CONTS2	073546	#490-3035	490-3036	490-3042						
CONTT	073404	480-2679	488-2943	#490-3007						
COREHI	043300	330-6957	330-6971	330-6984	332-7001	#332-7006	349-7367	349-7434		
COUNT	002462	#39-1250	*249-5505	249-5508	*249-5508	*249-5509	249-5510	*249-5517	*251-5526	251-5530
		*251-5530	*251-5531	251-5532	*251-5551					
CPUBIT	002114	#39-1155	46-1407	*51-1490	59-1625	63-1665	63-1682	66-1822	68-1911	68-1932
		70-1965	82-2205	84-2257	84-2261	84-2265	*90-2382	*91-2470	*91-2495	*91-2506
		95-2563	*95-2572	*95-2574	97-2630	311-6640	314-6675	325-6888	349-7334	366-4
		380-309	*390-463	*390-465	398-638	408-921	408-922	408-923	*411-1040	*411-1051
		*411-1060	*413-1072	*413-1085	*415-1112	417-1195	*421-1249	*421-1251	*424-1314	*424-1316
		*426-1374	*426-1376	*428-1422	*428-1424	*430-1464	*430-1466	*434-1529	*434-1531	*438-1620
		*438-1622	*440-1681	*440-1683	457-2135	488-2931				
CPUERR	= 177766	#12-348	*51-1502	*51-1513	*57-1612	*61-1646	65-1768	*65-1774	*66-1823	*68-1874
		*76-2098	*341-7194	*341-7198	*343-7210	*343-7227	*345-7266	*347-7318	*457-2158	*458-2195
		*461-2277	561-4704	561-4709	561-4714					
		84-2259	84-2263	84-2267	#84-2271					
CPUID	011656	#12-271	408-937							
CR	= 000015	#12-272	408-890	565-4782	569-4839	569-4840	569-4841	569-4842	569-4844	569-4846
CRLF	= 000200	569-4847	569-4848	569-4849	569-4850	569-4851	569-4857	569-4861	569-4862	569-4863
		569-4864	569-4865	569-4866	569-4867	569-4868	569-4869	569-4870	569-4871	569-4872
		569-4873	569-4874	569-4875	569-4876	569-4877	569-4878	569-4879	569-4880	569-4881
		569-4882	569-4883	569-4885	569-4886	569-4887	569-4888	569-4889	569-4890	569-4891
		569-4892	569-4893	569-4894	569-4895	569-4896	569-4897	569-4901	569-4903	569-4904
		569-4905	569-4906	569-4907	569-4908	569-4909	569-4910	569-4911	569-4912	569-4913
		569-4913	569-4914	569-4916	569-4917	569-4918	569-4919	569-4920	569-4922	569-4923
		569-4924	569-4926	569-4927	569-4928	569-4932	569-4934	569-4938	569-4939	569-4940
		569-4941	569-4944	569-4945	569-4945	569-4946	569-4947	569-4948	569-4949	569-4950
		569-4952	569-4953	569-4954	569-4955	569-4956	569-4959	569-4960	569-4961	569-4967
		569-4968	569-4969	569-4972	569-4973	569-4974	569-4975	569-4976	569-4976	569-4978
		569-4979	569-4980	569-4981	569-4982	569-4983	569-4984	569-4985		
CSR	002144	#39-1167	65-1712	65-1731	65-1732	65-1751	*70-1968	*70-1979	*70-1987	*70-1995
		*78-2142	*78-2143	*78-2145	*78-2146	*78-2147	*78-2149	*78-2149	*78-2150	78-2153
		*93-2549	*93-2550	134-3295	136-3349	136-3368	138-3396	*140-3424	*140-3425	*140-3428
		140-3429	140-3432	147-3637	150-3672	150-3673	237-5283	237-5284	*237-5290	237-5304
		237-5306	*237-5311	*237-5319	239-5326	*239-5334	239-5348	239-5350	*239-5355	*239-5364
		241-5372	*241-5380	241-5391	241-5393	*241-5399	289-6280	289-6282	289-6291	289-6293
		*291-6309	*291-6310	*291-6315	*291-6316	291-6337	291-6337	*291-6348	*291-6349	*294-6355
		*294-6356	*294-6357	*294-6361	*294-6362	*294-6363	*294-6367	*294-6368	*294-6372	*294-6373
		*294-6377	*294-6378	*294-6379	*294-6383	*294-6384	*294-6385	*294-6389	*294-6390	*294-6391
		*294-6392	*294-6396	*294-6397	*294-6398	*294-6399	297-6414	297-6434	299-6449	299-6468
		299-6470	*301-6480	*301-6480	*301-6484	*301-6484	*301-6489	*301-6490	*301-6491	*301-6496
		*301-6497	*301-6498	*343-7213	*343-7216	394-587	399-685	399-685	*401-739	*401-739
		455-2103	471-2474	471-2474	471-2478	471-2479	471-2482	*471-2495	*471-2495	480-2682
		480-2682	480-2691	480-2693	*480-2699	*480-2699	561-4711	561-4711	561-4712	561-4712
		561-4713	561-4713							
CSRCAS	020114	132-3255	#143-3448							

CEMKAB SYMBOL	CREATED BY	MACRO	ON	DATE	TIME	PAGE	CREF	V01	SEQ	1086
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES							
CSRFB	002214	#39-1171	*65-1733	476-2595						
CSRFBA	002410	#39-1225	132-3264	*138-3392	*145-3505	*145-3521	*145-3537	*145-3553	*145-3581	*145-3597
		*145-3613								
CSRFIR	002336	#39-1207	132-3246	*145-3499	*145-3507	*145-3509	*145-3515	*145-3523	*145-3525	*145-3531
		*145-3539	*145-3541	*145-3547	*145-3555	*145-3557	*145-3575	*145-3583	*145-3585	*145-3591
		*145-3599	*145-3601	*145-3607	*145-3615	*145-3617	147-3628	163-3975	163-3999	237-5285
		349-7323	*349-7382	*349-7433						
CSRINC	002414	#39-1227	132-3261	*145-3501	*145-3511	*145-3517	*145-3527	*145-3533	*145-3543	*145-3549
		*145-3559	*145-3577	*145-3587	*145-3593	*145-3603	*145-3609	*145-3619	*163-3964	163-3977
		237-5286								
CSRIND	002416	#39-1228	*132-3229	132-3232	*132-3274					
CSRLB	002234	#39-1172	*65-1738	476-2596						
CSRLBA	002412	#39-1226	132-3238	*138-3393						
CSRLOO	002420	#39-1230	*132-3231	132-3267	*132-3276	132-3276	134-3297			
CSRNO	002256	#39-1174	*65-1703	65-1729	65-1749	*65-1772	65-1773	*78-2138	*78-2161	78-2161
		*93-2551	*93-2556	93-2556	*134-3292	*163-3958	289-6277	291-6306	291-6325	*297-6410
		*297-6420	297-6420	*299-6445	*299-6455	299-6455	*303-6507	*303-6513	303-6513	339-7133
		*341-7178	352-7448	*352-7491	*364-7779	*364-7791	398-648	*399-681	*399-688	399-688
		*401-736	*401-743	401-743	*402-776	471-2474	*471-2475	*471-2495	476-2588	*476-2625
		480-2682	*480-2687	*480-2697	480-2697	*480-2699	561-4709			
CSROUD	002417	#39-1229	*132-3230	*132-3275	134-3287					
CSROUT	040442	294-6358	294-6369	294-6380	294-6393	301-6481	301-6492	#303-6503		
CSRREL	002254	#39-1173	*61-1640	*61-1645	289-6278	291-6307	291-6326			
CSRO	002154	#39-1169	*65-1731							
CSR2	002174	#39-1170	*65-1732	476-2600						
DATBUF	002340	#39-1208	*199-4553	*199-4554	199-4555	199-4556	199-4558	199-4563	199-4567	*199-4569
		*199-4569	*199-4572	*199-4573	199-4574	199-4575	199-4577	199-4582	199-4586	*199-4588
		*199-4588	*207-4690	*207-4691	207-4696	207-4697	207-4765	*207-4767	*207-4767	*210-4786
		*210-4787	210-4790	210-4791	210-4847	*210-4849	*210-4849	*213-4864	*213-4865	213-4870
		213-4871	*218-4936	*218-4937	218-4942	218-4943	*221-5013	*221-5014	221-5019	221-5020
		*224-5092	*224-5093	224-5098	224-5099	446-1919	446-1924			
DBEMSK	002354	#39-1211	*213-4868	*213-4869	213-4877	213-4879	213-4887	213-4889	215-4906	*215-4908
		*215-4908	*218-4940	*218-4941	218-4949	218-4951	218-4959	218-4961	218-4987	*218-4989
		*218-4989	*221-5017	*221-5018	221-5026	221-5028	221-5036	221-5038	221-5064	*221-5066
		*221-5066	*224-5096	*224-5097	224-5104	224-5106	224-5114	224-5116	226-5144	*226-5146
		*226-5146								
DDISP	= 177570	#12-265	43-1347	51-1497						
DECMAR	043152	114-2907	118-2984	125-3117	127-3161	#330-6976				
DEENER	= 104421	#10-188	61-1648	334-7037	336-7080					
DETAIL	071364	465-2360	*478-2645	478-2674						
DETFLA	002326	#39-1196	461-2262	465-2359	*478-2646	478-2647	478-2649	*482-2749		
DETPSW	002324	#39-1195	561-4710							
DETRO	002306	#39-1188	*478-2654	478-2663	561-4710					
DETR1	002310	#39-1189	478-2655	561-4710						
DETR2	002312	#39-1190	561-4710							
DETR3	002314	#39-1191	561-4710							
DETR4	002316	#39-1192	561-4710							
DETR5	002320	#39-1193	561-4710							
DETPSP	002322	#39-1194	561-4710							
DF1	102740	#563-4729								
DF10	103076	551-4513	*563-4738							
DF11	103103	549-4421	551-4488	551-4493	551-4498	553-4520	553-4525	553-4540	557-4649	557-4654

CEMKAB SYMBOL	VALUE	REFERENCES	CREF	V01	SEQ
DF12	103114	#563-4739			
DF13	103116	555-4622 #563-4740	555-4587	557-4634	557-4639
DF14	103125	553-4555 553-4570			557-4679 #563-4741
DF2	102743	559-4696 #563-4742	549-4446	549-4451	549-4456
		549-4426 549-4441	555-4577	555-4627	557-4644
		553-4545 553-4550			551-4473 551-4503
		#563-4730			557-4664 557-4669
DF3	102767	551-4463 553-4565	555-4592	555-4597	555-4607
DF4	103004	551-4468 #563-4732			555-4612 555-4617
DF5	103021	551-4478 551-4508	555-4602	#563-4733	557-4684 #563-4731
DF6	103036	551-4483 #563-4734			
DF7	103053	553-4560 #563-4735			
DF8	103067	555-4582 #563-4736			
DF9	103071	549-4431 549-4436	#563-4737		
DH1	107366	549-4424 549-4454	#567-4807		
DH10	110145	551-4471 #567-4818			
DH11	110243	551-4501 553-4533	#567-4819		
DH12	110272	553-4528 #567-4820			
DH13	110307	549-4419 551-4486	551-4491	551-4496	553-4518
DH14	110415	553-4543 #567-4823			553-4523 553-4538
DH15	110533	553-4558 #567-4825			#567-4821
DH16	110655	553-4563 555-4580	557-4682	#567-4827	
DH17	110672	553-4553 553-4568	555-4585	557-4632	557-4637
DH18	110746	557-4647 557-4652	#567-4829		557-4677 #567-4828
DH19	111023	555-4620 #567-4830			
DH2	107423	557-4642 #567-4808			
DH20	111037	557-4662 557-4672	#567-4831		
DH21	111056	557-4667 #567-4832			
DH22	111063	559-4689 #567-4833			
DH23	111111	559-4694 #567-4834			
DH3	107446	549-4429 549-4434	#567-4809		
DH4	107502	#567-4810			
DH5	107507	549-4439 549-4444	549-4449	553-4548	#567-4811
DH6	107615	555-4625 #567-4813			
DH7	107664	551-4461 551-4466	551-4476	551-4481	551-4506
		555-4610 555-4615	#567-4814		555-4590
DH8	110006	551-4511 #567-4816			555-4595
DH9	110106	555-4575 #567-4817			555-4600
DIAGFL	002006	#39-1118 *165-4059	*165-4061	249-5522	
DISPLA	002770	#43-1347 *51-1497	*51-1504	356-7604	
		411-1035 413-1067	415-1180	421-1244	366-34
		438-1618 440-1675	442-1720	458-2192	376-223
		#14-549 51-1504	*402-775	*458-2193	390-461
DISPRE	= 000174	100-2714 #100-2716			402-774
DISPTB	014370	#398-639 398-640			409-954
DNSLAV	055170	99-2708 103-2729	105-2781	#110-2843	430-1462
DOBACK	015104	#12-264 43-1346	51-1496		432-1524
DSWR	= 177570	549-4425 549-4455	#561-4700		
DT1	102252	551-4472 #561-4710			
DT10	102444	551-4502 553-4534	#561-4711		
DT11	102466	553-4529 #561-4712			
DT12	102476	549-4420 551-4487	551-4492	551-4497	553-4519
DT13	102504				553-4524 553-4539

CEMKAB SYMBOL	CROSS REFERENCE VALUE	REFERENCES
EM29	105027	555-4594 #565-4781
EM3	103256	549-4428 #565-4754
EM30	105111	555-4599 #565-4782
EM31	105230	555-4604 #565-4784
EM32	105330	555-4609 #565-4785
EM33	105435	555-4614 #565-4786
EM34	105543	555-4619 #565-4787
EM35	105624	555-4589 #565-4788
EM36	105711	555-4624 #565-4789
EM37	105760	557-4631 #565-4790
EM38	106025	553-4552 #565-4791
EM39	106133	557-4636 #565-4792
EM4	103311	549-4433 #565-4755
EM40	106220	557-4641 #565-4793
EM41	106272	557-4646 #565-4794
EM42	106374	557-4651 #565-4795
EM43	106477	557-4661 #565-4796
EM44	106575	557-4666 #565-4797
EM45	106672	557-4671 #565-4798
EM46	106773	557-4676 #565-4799
EM47	107062	557-4681 #565-4800
EM48	107170	559-4688 #565-4801
EM49	107267	559-4693 #565-4802
EM5	103357	549-4438 #565-4756
EM6	103434	549-4443 #565-4757
EM7	103461	549-4448 #565-4758
EM8	103521	553-4567 #565-4759
EM9	103556	555-4574 #565-4761
ENASBE =	104506	#10-246 66-1827 70-2023
ENATSB =	104507	#10-247 259-5658 259-5691 261-5702
END	117144	46-1430 #569-4997 569-5007 569-5008
ENERGI =	104420	#10-187 63-1686 108-2827 417-1215 419-1236
ERRMAX	002702	#43-1316 453-2047
ERROR =	104000	#12-259 65-1770 66-1817 66-1821 72-2038 80-2184 90-2395 91-2448 91-2485
		93-2521 #95-2583 95-2619 99-2695 99-2704 103-2758 105-2775 210-4815 213-4902
		259-5663 261-5720 261-5731 282-6197 284-6224 284-6228 284-6231 284-6234 299-6472
		341-7193 343-7209 343-7223 366-28 386-409 386-417 390-485 390-511 390-531
		392-550 394-600 409-976 409-1002 409-1019 411-1052 413-1086 413-1099 417-1197
		421-1268 421-1290 424-1332 424-1354 426-1404 428-1445 430-1481 430-1495 436-1592
		436-1606 438-1639 438-1652 440-1704 442-1729 449-1994 #449-1996 449-2006 #449-2008
		451-2017 453-2067 453-2070 453-2073 453-2076 455-2087 455-2098 455-2104 #455-2106
		455-2115 478-2665 478-2671 484-2778
ERRPC	002024	#39-1128 *460-2228 *460-2229 460-2232 *460-2235 *460-2236 460-2241 463-2295 561-4700
		561-4701 561-4702 561-4703 561-4704 561-4705 561-4706 561-4709 561-4711 561-4713
		561-4715 561-4717 561-4718 561-4719 561-4720 561-4721 561-4722 561-4723 561-4724
		561-4725
ERRPSW	002034	#39-1132 *460-2231 *460-2238 478-2662
ERRSP	002030	#39-1130 *460-2230 *460-2237 478-2661
ERRVEC =	000004	#12-318 *48-1469 *48-1470 457-2152 *457-2153 *457-2155 *457-2159
EVEN	002456	#39-1248 *249-5493 249-5494 *251-5557 251-5557
EXBANK	042422	72-2035 76-2112 78-2128 82-2209 82-2217 95-2565 95-2573 95-2589 103-2753
		110-2846 112-2862 114-2896 116-2930 118-2971 121-3015 123-3057 125-3106 127-3148

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	132-3239	132-3280	170-4135	170-4158	172-4188	172-4207	314-6653	318-6727	318-6731
			#325-6861	349-7362	349-7437	352-7478	354-7532	354-7546	356-7615	358-7681	358-7695
			390-464	413-1075	415-1168	415-1175	421-1250	424-1315	426-1375	428-1423	430-1465
			434-1530	438-1621	440-1682						
EXCMD3	044726		345-7255	345-7258	345-7262	#345-7264					
EXCMD4	045236		347-7298	347-7302	#347-7316						
EXIT	043522		#336-7053	457-2143	461-2265						
EXIT2	043526		#336-7055								
FASTCI =	177640		#12-378	68-1869	68-1899	84-2291	155-3824	183-4360	241-5410	291-6328	291-6333
			338-7103	386-399							
FATAL\$	002070		#39-1145	*66-1817	*66-1821	*259-5663	*282-6197	*284-6224	*284-6228	*284-6231	*366-28
			461-2263	463-2308							
FCMD10	047550		339-7154	#356-7618							
FCMD11	047734		339-7155	#358-7645							
FCMD12	050330		339-7156	#360-7700							
FCMD13	050356		339-7157	#360-7710							
FCMD14	050400		339-7158	#360-7715							
FCMD15	050420		339-7159	#360-7721							
FCMD16	050436		339-7160	#360-7727							
FCMD17	050454		339-7161	#360-7733							
FCMD18	050470		339-7162	#362-7739							
FCMD19	050502		339-7163	#362-7744							
FCMD20	050566		339-7164	#362-7763							
FIELDS	043774		#339-7124	488-2940	511-3699						
FIRST =	060000		#14-556	68-1862	68-1891	68-1917	70-1959	70-1970	70-1970	*70-2014	*70-2014
			103-2749	145-3499	145-3507	145-3509	145-3515	145-3523	145-3525	145-3531	145-3539
			145-3541	145-3547	145-3555	145-3557	145-3575	145-3583	145-3585	145-3591	145-3599
			145-3601	145-3607	145-3615	145-3617	153-3783	153-3791	153-3804	155-3815	155-3841
			155-3852	157-3871	165-4030	166-4074	168-4107	170-4139	170-4162	172-4192	174-4226
			174-4231	174-4232	176-4248	176-4264	176-4268	178-4285	178-4288	178-4289	178-4290
			178-4291	178-4299	229-5174	241-5407	249-5506	251-5527	251-5561	251-5563	251-5564
			271-5943	*336-7088	*336-7089	*336-7090	345-7245	347-7284	349-7382	352-7482	356-7569
			366-23	366-31	366-33	366-35	*366-60	368-68	368-82	368-83	368-86
			370-102	370-103	370-106	372-147	372-151	*372-159	374-166	374-170	374-179
			374-181	*374-191	376-233	376-234	376-238	376-246	378-264	378-265	378-281
			378-291	380-305	*380-306	380-310	380-313	380-319	*382-350	*382-352	384-372
			384-376	384-377	386-388	386-391	386-392	386-408	386-416	388-436	*390-469
			390-494	390-514	*392-558	404-800	404-801	404-802	*404-803	*404-804	404-805
			404-807	*406-837	*406-838	406-843	408-926	*408-928	*408-929	426-1381	426-1384
			469-2435								
FIRSTB	002260		#39-1175	65-1733	65-1734	65-1739	*65-1763	65-1764	138-3392	*140-3438	140-3439
			140-3441	*140-3441	145-3505	145-3521	145-3537	145-3553	145-3581	145-3597	145-3613
			*476-2595	476-2599							
FLIPLO	002734		#43-1329	*48-1445	155-3831	*271-5915	*271-5916	271-5917	271-5919	271-5921	
FLIPWA	035214		155-3814	#271-5913							
FLUSH	014656		#105-2784	415-1184							
FSCMD0	044170		#339-7144	#341-7169							
FSCMD1	044256		339-7145	#341-7187							
FSCMD2	044346		339-7146	#343-7202							
FSCMD3	044520		339-7147	#345-7231							
FSCMD4	044764		339-7148	#347-7270							
FSCMD5	045274		339-7149	#349-7322							

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
FSCMD6		046152	339-7150 #349-7440
FSCMD7		046160	339-7151 #352-7447
FSCMD8		046510	339-7152 #354-7496
FSCMD9		047104	339-7153 #356-7550
FSINFL		002542	#39-1274 *108-2797 145-3568 *145-3569
FSPAT		045752	349-7392 #349-7396
FSSAC		002374	#39-1219 *341-7189 341-7197 *343-7204 343-7226 *349-7325 349-7429 *352-7468 352-7472
			*354-7498 354-7541 *356-7552 356-7610 *358-7647 358-7690
FS1		044052	#339-7136 339-7142 339-7166
FS7FLA		002546	#39-1276 170-4152 *352-7449 *352-7490
GBLENG	=	000076	178-4285 178-4288 #270-5902
GDSWIT		002476	#39-1256 *80-2179 561-4708
GETDAT		055030	#396-614 449-1990 449-2002
GETDA1		055156	*396-616 396-626 #396-631
GETDIS		066514	183-4336 186-4370 457-2172 #458-2176 460-2217
GETSIZ		017736	65-1727 138-3391 #140-3421
GOOD		002050	#39-1137 *65-1769 *80-2175 *99-2702 *386-406 *386-414 *394-576 *394-578 *394-592
			*396-605 *446-1886 *446-1892 *446-1897 *446-1904 *446-1907 *446-1914 *446-1919 *446-1924
			*446-1929 *446-1934 *448-1940 *448-1945 *448-1950 *448-1955 *448-1960 *448-1965 *448-1970
			*448-1975 *448-1980 *448-1985 451-2030 *453-2056 *453-2058 *455-2085 *455-2095 *455-2114
			561-4700 561-4706 561-4708 561-4715 561-4720 561-4721
GOOD2		002052	#39-1138 *451-2020 *451-2024 561-4715
GOOD3		002054	#39-1139 *451-2021 *451-2025
GTSWR	=	104407	#10-174 63-1667
HEADER		002724	#43-1325 *48-1446 *103-2734 *103-2736 *110-2848 *110-2851 *130-3197 *130-3210 *132-3252
			*207-4741 *207-4744 *207-4755 *207-4758 *210-4813 *210-4816 *210-4827 *210-4830 *213-4900
			*213-4903 *261-5721 *261-5732 *349-7380 *354-7529 *358-7678 *453-2061 *453-2078 *455-2086
			*455-2088 *455-2097 *455-2099 *461-2275 463-2306 463-2318 *465-2357 478-2651 *478-2652
			478-2667 *478-2668 *478-2672
HIACBA		002514	#39-1263 *78-2124 *78-2131 78-2135 78-2143
HIADRS	=	177742	#12-336 561-4704 561-4714
HIBANK		002512	#39-1262 *76-2120
HIPAT		042762	125-3102 127-3144 #328-6937
HT	=	000011	#12-269 408-887
HUNGMS		002572	#39-1286 *91-2441 *91-2445 *91-2477 *91-2482 *411-1044 *411-1048
HUNGTI		002574	#39-1287 *90-2391 *91-2442 *91-2478 *390-481 *390-527 *409-972 *409-999 *409-1015
			*411-1045 *413-1081 *417-1193 *421-1264 *421-1286 *424-1328 *424-1350 *426-1400 *428-1441
			*430-1477 *436-1602 *438-1635 *440-1700 *442-1725 *484-2785
I		002602	#39-1290 *55-1568 *55-1574 55-1574 *97-2650 *97-2650 97-2651 *97-2651 97-2653
			*132-3250 132-3254 *132-3256 132-3256 *170-4133 170-4143 170-4146 *170-4151 170-4151
			*170-4156 170-4166 170-4169 *170-4174 170-4174 *380-311 *380-318 380-318 *390-495
			*390-513 390-513 *434-1537 434-1539 *434-1568 434-1568 436-1581 436-1589 436-1591
IISTAC		002736	#43-1330 57-1603 57-1604 84-2274 84-2275 84-2295 90-2392 90-2403 91-2418
			366-18 408-915 409-994 409-1000 417-1189 417-1190 417-1194 417-1209
IISTAD		002740	#43-1331 57-1608 84-2296 90-2399 90-2404 90-2405 91-2419 91-2420 409-995
			409-996 409-1007 417-1191 417-1192 417-1202 417-1210
IISTIN		061074	84-2293 #417-1187 417-1208
IISTOF		061174	90-2402 409-993 #417-1207
IISTVE		002742	#43-1332 90-2386 91-2464 380-303 380-316 382-341 382-342 382-343 382-343
			382-346 382-347 382-347 384-358 392-563 409-985
IITRAP		036764	48-1473 91-2464 #284-6233
INCBNK		042772	112-2879 114-2913 116-2949 118-2990 121-3029 123-3073 125-3120 127-3164 #328-6941

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
KIPAR6	=	172354	#12-465
KIPAR7	=	172356	#12-466
KIPDRO	=	172300	#12-439 311-6632 322-6819 398-661 401-725 401-746 406-823 419-1227
KI0		002550	#39-1277 82-2233 419-1224
KI1		002552	#39-1278
KI2		002554	#39-1279
KI3		002556	#39-1280
KI4		002560	#39-1281
KI5		002562	#39-1282
KI6		002564	#39-1283
KI7		002566	#39-1284
KMAP	=	104422	#10-190 63-1683
KPFLAG		C02122	#39-1158 *70-1958 72-2037 82-2211 294-6356 294-6362 294-6378 294-6384 294-6391
			294-6398 301-6490 *325-6877 *325-6885
KSTACK		002706	#43-1318 46-1406 57-1611 *90-2378 90-2379 *91-2471 *91-2507 108-2825 *368-71
			368-73 *409-962 *409-978 *409-990 *409-1004 *409-1021 *409-1028 *413-1105 417-1213
			482-2707
LAST	=	157776	#14-557 132-3261 153-3800 163-3980 165-4028 166-4075 176-4249 176-4263 229-5176
			237-5289 249-5507 251-5529 251-5565 266-5805 270-5893 345-7247 347-7286 356-7571
			376-244 388-437 426-1384
LASTB		002262	#39-1176 65-1738 65-1764 138-3393 *140-3431 *140-3434 *140-3435 *140-3439 *140-3440
			140-3442 *140-3442 *476-2596 476-2599
LASTER		002020	#39-1126 *80-2185 105-2774 *108-2798
LF	=	000012	#12-270 408-941
LKS	=	177546	#12-266 51-1510 *374-177 *374-184
LOADBA		002532	#39-1270 *314-6656 318-6721 318-6726
LOADCS	=	104425	#10-195 78-2148 78-2151 93-2554 140-3426 294-6364 294-6374 294-6386 294-6400
			301-6485 301-6499 303-6510 343-7218 401-740
LOADHO		002710	#43-1319 *82-2219 82-2222 82-2224 108-2828 311-6641 314-6657 318-6722 318-6730
			336-7078 415-1174
LOADRS	=	177740	#12-335 282-6176 561-4704 561-4714
LOCK		002670	#41-1307 *46-1436 46-1437 *57-1614 *59-1620 59-1621 *78-2157 *84-2287 *84-2299
			*90-2368 *91-2501 91-2502 *336-7082 *409-961 *409-989 *411-1036 *413-1068 *415-1183
LOOP		014342	#100-2710 108-2840
LOWMAP		042026	63-1678 316-6709 318-6749 #318-6754 401-722
LSTSS	=	000000	#1-1
MAINT	=	177750	#12-339 *63-1672 *103-2768 *130-3196 *170-4127 *172-4182 *332-7012 *334-7034 *336-7072
			*349-7378 *354-7527 *356-7600 *358-7676 *370-131 *374-193 *376-253 *384-366 *396-620
			*396-628 398-644 *402-780 458-2182 490-3023 561-4714
MAPHO	=	170202	#14-484 *63-1677 *316-6708 *318-6748 399-698 *401-721
MAPH36	=	170372	#14-544 *63-1675 *76-2084 *76-2091 76-2092 *76-2097 *76-2101 76-2102 *76-2105
			399-698 *401-721
MAPH37	=	170376	#14-546
MAPLO	=	170200	#14-483 *63-1677 *316-6706 *318-6748 318-6756 399-698 *401-721
MAPL1	=	170204	#14-485 318-6757
MAPL36	=	170370	#14-543 *63-1676 *76-2085 *76-2090 *76-2096 *76-2100 *76-2106 399-698 *401-721
MAPPER		040700	63-1685 68-1858 70-1967 70-2019 132-3241 174-4224 176-4244 178-4282 183-4335
			186-4369 #311-6604 336-7086 345-7250 347-7289 349-7360 349-7436 354-7545 356-7587
			356-7614 358-7694 366-11 366-58 368-75 372-156 374-168 374-188 380-308
			382-349 382-351 384-374 386-390 388-435 390-467 390-477 404-798 406-835
			408-924 408-930
MARGIN		002112	#39-1154 *112-2869 *116-2939 *121-3022 *123-3066 130-3194 *170-4127 *172-4182 330-6955

CEMKAB SYMBOL	CREATED BY	MACRO ON	PAGE 15	I 5	SEQ 1095
CROSS REFERENCE	VALUE	REFERENCES	CREF	V01	
MMRO	= 177572	#12-352 *286-6254 *286-6258 *316-6711 *316-6713 *318-6743 *318-6745 *376-221 399-691			
		*401-731 404-814 *406-831 561-4704 561-4714			
MMR1	= 177574	#12-353 399-691 *401-731 404-813 *406-830 561-4704 561-4714			
MMR2	= 177576	#12-354 399-691 *401-731 404-812 *406-829 561-4704 561-4714			
MMR3	= 172516	#12-355 *63-1671 *63-1679 *108-2826 *316-6702 *316-6710 *318-6747 *318-6750 *334-7038			
		*336-7081 399-691 *401-731 404-811 *406-828 561-4704 561-4714			
MMTRAP	036740	48-1471 #284-6227			
MMVEC	= 000250	#12-333 *48-1471 *48-1472			
MP1	012070	84-2284 84-2294 #88-2319			
MP2	012224	#90-2392 90-2400 91-2437			
MP3	012716	91-2466 91-2487 #91-2494			
MP4	012552	90-2373 #91-2467			
MP5	012764	#91-2506			
MRUNSE	053640	#388-434 438-1624			
MSEEDH	002720	#43-1323 *48-1447 48-1449			
MSEEDL	002722	#43-1324 *48-1448 48-1450			
MSG A34	113625	*132-3267 #569-4898			
MSG A87	115753	*93-2536 *93-2540 #569-4957			
MSG A98	116242	*484-2763 #569-4970			
MSG B34	113642	*132-3268 #569-4900			
MSG 000	111176	57-1582 #569-4839			
MSG 001	111254	274-5962 #569-4840			
MSG 002	111336	274-5963 #569-4841			
MSG 003	111413	274-5964 #569-4842			
MSG 004	111520	274-5970 274-5988 #569-4844			
MSG 005	111626	277-6015 #569-4846			
MSG 006	111640	498-3467 #569-4847			
MSG 007	111675	279-6049 #569-4848			
MSG 008	111707	277-6029 #569-4849			
MSG 009	111721	279-6089 #569-4850			
MSG 010	111733	279-6135 #569-4851			
MSG 011	111745	274-5974 274-5987 #569-4852			
MSG 012	112033	274-5977 274-5989 #569-4853			
MSG 013	112130	277-6018 #569-4854			
MSG 014	112132	274-5986 277-6020 279-6145 460-2222 467-2388 469-2456 #569-4855			
MSG 015	112134	*134-3297 134-3298 *277-6038 277-6039 *279-6056 *279-6077 279-6078 *279-6094 *279-6097			
		*279-6099 279-6102 *279-6116 *279-6120 *279-6125 279-6126 *279-6138 *279-6142 279-6143			
		*473-2510 *473-2512 473-2514 *473-2530 *473-2532 473-2534 #569-4856			
MSG 016	112136	279-6112 #569-4857			
MSG 017	112150	274-5973 274-5976 #569-4858			
MSG 018	112161	465-2352 469-2454 473-2505 473-2523 480-2692 480-2694 482-2712 482-2726 482-2742			
		#569-4859			
MSG 019	112164	#569-4860			
MSG 020	112170	339-7126 #569-4861			
MSG 021	112224	339-7141 #569-4862			
MSG 022	113252	364-7782 #569-4885			
MSG 023	113274	343-7211 #569-4886			
MSG 024	113316	343-7214 #569-4887			
MSG 025	113340	341-7196 343-7225 #569-4888			
MSG 026	113371	339-7136 #569-4889			
MSG 027	113403	343-7207 #569-4890			
MSG 028	113420	343-7221 #569-4891			

CEMKAB
SYMBOL CROSS REFERENCE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 16
CREF V01

J 5

SEQ 1096

SYMBOL	VALUE	REFERENCES
MSG029	113434	345-7234 #569-4892
MSG030	113454	349-7327 362-7749 #569-4893
MSG031	113473	345-7235 347-7274 356-7559 #569-4894
MSG032	113533	345-7257 347-7297 #569-4895
MSG033	113552	345-7261 347-7301 #569-4896
MSG034	113571	132-3269 #569-4897
MSG035	113707	108-2803 #569-4902
MSG036	113712	347-7273 #569-4903
MSG037	113731	347-7304 #569-4904
MSG038	113750	347-7313 #569-4905
MSG039	113766	347-7306 #569-4906
MSG040	114010	349-7326 #569-4907
MSG041	114055	349-7335 #569-4908
MSG042	114102	349-7339 #569-4909
MSG043	114123	349-7344 #569-4910
MSG044	114150	349-7349 354-7500 356-7573 358-7649 #569-4911
MSG045	114175	349-7356 354-7507 356-7580 358-7656 #569-4912
MSG046	114231	349-7368 352-7462 354-7518 356-7591 358-7667 #569-4913
MSG047	114264	352-7489 #569-4914
MSG048	114303	339-7129 #569-4915
MSG049	114343	349-7364 #569-4916
MSG050	114375	352-7455 #569-4917
MSG051	114503	402-778 #569-4919
MSG052	114523	352-7459 #569-4920
MSG053	114554	352-7461 #569-4921
MSG054	114572	352-7476 #569-4922
MSG055	114642	354-7499 #569-4923
MSG056	114663	354-7516 358-7665 #569-4924
MSG057	114716	476-2585 *476-2621 476-2623 *476-2624 #569-4925
MSG058	114722	480-2683 #569-4926
MSG059	114766	356-7553 #569-4927
MSG060	115016	356-7589 #569-4928
MSG061	115060	469-2428 #569-4929
MSG062	115067	492-3191 492-3250 #569-4930
MSG063	115107	492-3192 #569-4931
MSG064	115120	492-3193 492-3252 #569-4932
MSG065	115130	492-3251 #569-4933
MSG066	115142	461-2268 #569-4934
MSG067	115225	463-2310 #569-4935
MSG068	115234	467-2414 #569-4936
MSG069	115241	467-2416 #569-4937
MSG070	115246	97-2654 #569-4938
MSG071	115271	97-2656 #569-4939
MSG072	115304	97-2658 #569-4940
MSG073	115317	358-7648 #569-4941
MSG074	115336	*471-2472 *471-2493 471-2497 *475-2544 *475-2558 475-2560 #569-4943
MSG075	115341	314-6667 314-6688 #569-4944
MSG076	115373	356-7630 #569-4945
MSG077	115414	108-2801 #569-4946
MSG078	115430	134-3296 #569-4947
MSG079	115474	356-7621 #569-4948
MSG080	115520	57-1592 #569-4949

CEMKAB
SYMBOL CROSS REFERENCE
SYMBOL VALUE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 18
L 5
CREF V01

SEQ 1098

REFERENCES

SYMBOL	VALUE	REFERENCES
MTPB32	035020	#268-5826 378-295
MTPC03	026610	155-3827 #194-4494
MTPC21	033320	165-4038 #244-5450
MTPC24	034134	166-4079 #256-5634
MTPC25	034536	259-5692 261-5701 261-5703 #261-5725
MTPC26	034646	168-4111 #264-5763
MTPD03	026630	155-3828 #194-4503
MTPD21	033354	165-4041 #246-5468
MTPD25	034412	259-5665 259-5667 259-5670 259-5672 #259-5688
MTPD26	034666	168-4112 #264-5776
MTPE25	034434	259-5676 259-5678 259-5681 259-5683 #261-5696
MTP000	026376	68-1846 153-3786 170-4129 178-4306 #189-4413 189-4415 271-5945 370-133
MTP001	026422	153-3795 #189-4424 376-237
MTP002	026454	153-3806 #189-4436 376-243
MTP005	026724	155-3854 #196-4535
MTP006	026760	157-3864 #199-4550
MTP007	027150	157-3874 #202-4594
MTP010	027250	157-3881 #204-4634
MTP011	027356	159-3892 #207-4671
MTP012	027766	159-3901 #210-4782
MTP013	030344	159-3909 #213-4860
MTP014	030716	159-3919 #218-4929
MTP015	031312	161-3930 #221-5010
MTP016	031700	161-3939 #224-5087
MTP017	032302	161-3944 #229-5167
MTP020	032360	#232-5201
MTP022	033404	165-4050 165-4058 #249-5487
MTP025	034152	166-4095 #259-5645
MTP030	034704	172-4184 241-5406 #266-5784
MTP031	034714	174-4226 #266-5790 266-5812
MTP033	035054	176-4258 #268-5839
MTP034	035106	178-4285 #270-5859 270-5902
MTP20A	032360	163-3982 #232-5206 232-5210
MTP20B	032376	163-3987 #232-5214 232-5223
MTP20C	032430	163-3993 #232-5227 232-5236
MTP20D	032462	163-4000 #232-5240 232-5249
MTP20E	032514	163-4006 #234-5254 234-5263
MTP20F	032546	163-4012 #234-5267 234-5274
MTV020	023320	*163-3957 163-3958 *163-3971 163-3971 #163-3973
MT0000	021704	#153-3781 349-7396
MT0001	021740	150-3688 152-3745 #153-3789 349-7397
MT0002	022000	150-3689 152-3746 #153-3798 349-7398
MT0003	022050	152-3747 #155-3811 349-7399
MT0004	022216	150-3690 152-3748 #155-3836 349-7400
MT0005	022300	150-3691 152-3749 #155-3847 349-7401
MT0006	022360	143-3454 #157-3860 349-7402
MT0007	022414	150-3687 152-3744 #157-3867 349-7403
MT0010	022456	143-3455 #157-3877 349-7404
MT0011	022512	143-3457 #159-3886 349-7405
MT0012	022570	143-3458 #159-3895 349-7406
MT0013	022646	143-3459 #159-3904 349-7407
MT0014	022722	143-3460 #159-3913 349-7408

CEMKAB
SYMBOL CROSS REFERENCE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 19
CREF V01

M 5

SEQ 1099

SYMBOL	VALUE	REFERENCES
MT0015	023000	143-3461 #161-3924 349-7409
MT0016	023056	143-3462 #161-3933 349-7410
MT0017	023134	150-3686 152-3743 #161-3942 349-7411
MT0020	023156	150-3693 #163-3949 349-7412
MT0021	023612	150-3692 152-3750 #165-4021 349-7413
MT0022	023734	150-3694 #165-4046 349-7414 358-7684
MT0023	023766	152-3751 #165-4054 349-7415
MT0024	024032	150-3696 152-3753 #166-4064 349-7416
MT0025	024172	143-3456 #166-4090 349-7417
MT0026	024240	150-3695 152-3752 #168-4100 349-7418
MT0027	024370	103-2735 #170-4123 349-7419 352-7457
MT0030	024754	105-2785 #172-4179 334-7036 336-7075 349-7420
MT0031	025214	150-3697 152-3754 #174-4218 349-7421 354-7535
MT0033	025340	150-3698 152-3755 #176-4238 349-7423
MT0034	025530	150-3699 152-3756 #178-4276 349-7424
MT0035	025652	150-3664 150-3685 150-3700 152-3730 152-3742 152-3757 #178-4297
MT020X	023604	163-3991 163-3997 163-4004 163-4010 #163-4017
MT020Y	023472	163-3985 #163-3999
MT020Z	023322	163-3968 #163-3975
MT0999	025772	55-1571 143-3463 143-3464 143-3465 143-3466 143-3467 143-3468 143-3469 143-3470 143-3471 143-3472 143-3473 143-3474 143-3475 143-3476 143-3477 143-3478 143-3479 143-3480 143-3481 143-3482 143-3483 143-3484 143-3485 150-3664 150-3702 150-3703 150-3704 150-3705 150-3706 150-3707 150-3708 150-3709 150-3710 150-3711 150-3712 150-3713 150-3714 150-3715 150-3716 150-3717 150-3718 150-3719 150-3720 150-3721 152-3730 152-3759 152-3760 152-3761 152-3762 152-3763 152-3764 152-3765 152-3766 152-3767 152-3768 152-3769 152-3770 152-3771 152-3772 152-3773 152-3774 152-3775 152-3776 #180-4320 349-7422 349-7425 349-7426 349-7427
MULHLT	056150	#404-791 404-792
MULTIE	060746	409-979 409-1005 409-1022 411-1054 413-1088 413-1102 415-1121 415-1126 415-1131 415-1136 415-1141 415-1146 415-1151 415-1156 415-1161 #415-1167
MULTIO	002004	#39-1117 150-3654 152-3725 *360-7735 *362-7741
MULTIP	003014	#43-1358 46-1407 *57-1610 84-2256 *91-2463 *91-2486 *93-2524 93-2533 105-2783 401-711 408-911 408-920 *409-977 *409-1003 *409-1020 *411-1053 *413-1087 *413-1101 *415-1120 *415-1125 *415-1130 *415-1135 *415-1140 *415-1145 *415-1150 *415-1155 *415-1160 *415-1165 *417-1198
MULTIU	056142	401-711 #404-788
MUT	002116	#39-1156 *103-2734 *103-2737 *110-2848 *110-2850 *130-3198 *130-3209 *349-7380 *354-7529 *358-7678 465-2355 478-2651 *478-2653 478-2667 *478-2669 *478-2672
M13BAL	054472	390-516 #392-541
NEMCNT	002072	#39-1146 *66-1805 66-1818 *68-1857 68-1877 68-1889 68-1907 68-1931 *284-6210 284-6211 *284-6216 561-4703
NEWBAN	002400	#39-1221 172-4206 *316-6697 316-6703 *318-6738 322-6825 322-6838
NEWKER	042322	316-6712 318-6744 #322-6835
NEWLOA	042370	82-2227 311-6642 314-6659 318-6724 #322-6848
NOCLOC	002104	#39-1151 *51-1512 430-1460 432-1518
NOECCF	002474	#39-1255 *134-3299 *134-3301 159-3887 159-3896 159-3914 161-3925 161-3934
NOERRO	002530	#39-1269 *341-7192 *343-7208 *343-7222 460-2213 460-2233 461-2251 *461-2277 463-2304 463-2316 465-2359 *478-2664 *478-2670
NOFSMO	002526	#39-1268 *170-4132 *170-4153 *170-4175 *172-4186 *172-4203 *172-4214 339-7128
NOIIST	002756	*37-1106 #43-1338 57-1590 84-2271 84-2283 90-2373 366-15 408-912 409-959
NONEM	002102	#39-1150 *66-1806 *68-1844 *68-1916 *68-1925 284-6208
NONEXI	036646	66-1807 68-1845 282-6172 #284-6208

CEMKAB
SYMBOL CROSS REFERENCE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 22
CREF V01

C 6

SEQ 1102

SYMBOL	VALUE	REFERENCES
O.SVTT	100500	509-3659 531-4094 533-4110 #537-4167
O.T	075572	#505-3597 *527-4019 *529-4057 531-4067
O.TBIT	077644	#527-4019 529-4052 529-4054 531-4068
O.TBT	= 000020	#502-3576 527-4020 527-4023 527-4030 529-4058 531-4080
O.TCL2	076442	#513-3757 513-3762
O.TCSR	= 177564	#502-3581 537-4168 *537-4170 537-4171 537-4177 *537-4184 539-4229
O.TDB	= 177566	#502-3580 *539-4243
O.TL	101345	511-3717 511-3720 511-3728 #547-4381 547-4397
O.TRTC	101370	523-3937 523-3965 537-4159 #547-4400
O.TVEC	= 000014	#502-3574 509-3650 *509-3677 *509-3678
O.TYPE	101126	509-3665 509-3688 511-3698 511-3707 531-4098 #543-4286 543-4291 543-4297 545-4334
O.TYP1	101004	#539-4244 543-4287
O.UIN	075752	#507-3637 *523-3969 *537-4158 537-4193
O.UPC	075700	#507-3627 *509-3653 *527-4015 527-4027 *531-4063 531-4083 *531-4087 *531-4100
O.URO	075662	#507-3620 509-3662 511-3725
O.USP	075676	#507-3626 509-3685 511-3695 *535-4130 535-4131 535-4150
O.UST	075702	#507-3628 *527-4020 *527-4023 527-4026 527-4029 *529-4058 *531-4064 531-4072 *545-4302
O.WRD	076760	517-3823 #519-3858
O.WRD1	077036	519-3839 #519-3872 521-3922
O.WST	101202	509-3651 509-3674 527-4017 527-4031 529-4049 529-4056 531-4082 #545-4312
O.XXX	075564	#505-3587 *535-4129 535-4137 *535-4144 535-4151
PADDRE	002042	#39-1135 *282-6176 561-4702
PAFBAF	015730	100-2720 #121-3010 121-3039 121-3047
PAFBAW	016066	100-2721 #123-3052 123-3083 123-3088 123-3096
PARBAF	016246	100-2722 #125-3101 125-3130 125-3138
PARBAW	016404	100-2723 #127-3143 127-3174 127-3179 127-3187
PARCNT	002074	#39-1147 *66-1803 66-1815 *68-1856 68-1887 68-1905 68-1927 *82-2229 *103-2755
		103-2757 *282-6173 282-6174 561-4702
PARITY	036454	48-1465 #282-6168
PARTHE	002372	#39-1218 *259-5652 *261-5714 *261-5725 *261-5728 282-6189 345-7232 *345-7248 *345-7265
		347-7271 *347-7287 *347-7317 *356-7585
PARVEC	= 000114	#12-331 *48-1465 *48-1466
PASFLG	002362	#39-1213 *130-3199 *132-3253 *172-4180 172-4197 *172-4198 207-4698 207-4769 *207-4771
		213-4872 215-4916 *215-4918 218-4944 218-4997 *218-4999 221-5021 221-5074 *221-5076
		224-5100 226-5154 *226-5156 *349-7381
PATERR	002076	#39-1148 68-1848 *68-1855 68-1885 68-1903
PATPLU	005472	55-1551 55-1553 55-1556 55-1558 55-1561 55-1563 #55-1567
PATTER	002120	#39-1157 *110-2844 *112-2867 *114-2910 *116-2937 *118-2987 *121-3011 *123-3053 *125-3123
		*127-3167 150-3662 152-3728 *328-6932 328-6933 *328-6938 349-7323 *349-7341 349-7342
		349-7343 349-7390 *349-7435 352-7448 *352-7491 354-7497 *354-7544 358-7646 *358-7693
PCBUMP	002406	#39-1224 *150-3661 *152-3727 *155-3813 *155-3838 *155-3849 *157-3862 *157-3879 *168-4102
		282-6183 349-7323 *349-7389 *349-7433
PCONF1	035402	97-2660 #274-5953 349-7441
PCONFS	035662	*274-5955 274-5992 #274-5998
PCONF1	035572	274-5966 #274-5983
PCONF2	035630	274-5956 274-5981 #274-5992
PDP110	036752	48-1467 #284-6230
PENDBO	002330	#39-1197 *65-1708 *65-1735 78-2136 78-2137
PERA05	064732	#446-1908 446-1915
PERBNK	065564	394-599 449-1989 449-2001 451-2016 #453-2037 455-2111
PERECC	065650	#453-2053 453-2065
PERRAB	065402	446-1930 446-1935 #449-2000

CEMKAB
SYMBOL
SYMBOL
PERRAW

CREATED BY
CROSS REFERENCE
VALUE

MACRO ON 3-AUG-79 AT 17:42

PAGE 23
D 6
CREF V01

SEQ 1103

SYMBOL	VALUE	REFERENCES	396-612	446-1887	446-1893	446-1899	446-1905	446-1920	446-1925	448-1941
PERRAW	065330	394-580	396-612	446-1887	446-1893	446-1899	446-1905	446-1920	446-1925	448-1941
		448-1946	448-1951	448-1956	448-1961	448-1966	448-1976	448-1981	448-1986	#449-1988
PERRA3	054652	#394-582	451-2022	451-2026						
PERRA7	065454	448-1971	#451-2013							
PERR01	= 104427	#10-198	189-4432	268-5854						
PERR02	= 104430	#10-199	189-4420	189-4445	268-5848					
PERR03	= 104431	#10-200	194-4498	194-4506						
PERR04	= 104432	#10-201	196-4522	196-4544						
PERR05	064726	#446-1907								
PERR06	064754	#446-1914								
PERR07	= 104433	#10-202	199-4560	199-4579						
PERR10	= 104434	#10-203	199-4565	199-4584						
PERR11	= 104435	#10-204	202-4602	204-4652	237-5309	239-5353	241-5396			
PERR12	= 104436	#10-205	202-4608	204-4659						
PERR13	= 104437	#10-206	202-4616							
PERR14	= 104440	#10-207	202-4622							
PERR15	= 104441	#10-208								
PERR16	= 104442	#10-209								
PERR17	= 104443	#10-210	244-5421	244-5443	244-5454	246-5478	251-5535	251-5539		
PERR20	= 104444	#10-211	244-5427	244-5437	244-5460	246-5472	251-5544	251-5548		
PERR21	= 104445	#10-212	239-5332	241-5378						
PERR22	= 104446	#10-213	229-5189	268-5834						
PERR23	= 104447	#10-214	254-5611	256-5640						
PERR24	= 104450	#10-215	254-5616							
PERR25	= 104451	#10-216	264-5751	264-5756						
PERR26	= 104452	#10-217	232-5219	232-5244	234-5271					
PERR27	= 104453	#10-218	232-5231	234-5259						
PERR30	= 104454	#10-219	266-5797							
PERR31	= 104455	#10-220	207-4725	207-4731	218-4975	218-4981	221-5055	221-5060	224-5131	224-5137
PERR32	= 104456	#10-221	207-4737	207-4751						
PERR33	= 104457	#10-222	210-4821							
PERR34	= 104460	#10-223	207-4743	207-4757	210-4829					
PERR35	= 104461	#10-224	270-5871	270-5878						
PERR36	= 104462	#10-225								
PERR37	= 104463	#10-226								
PERR40	= 104464	#10-227								
PERR41	= 104465	#10-228								
PERR42	= 104466	#10-229								
PERR43	= 104467	#10-230								
PERXOR	065540	449-1992	449-2004	451-2015	#451-2028	453-2060	455-2096			
PFLAG	002130	#39-1161	110-2847	314-6654	*325-6879	*325-6900	390-466	413-1076	421-1252	424-1317
		426-1377	428-1425	430-1467	434-1534	438-1623	440-1684			
PHYADD	002044	#39-1136	*469-2434	*469-2435	*469-2436	*469-2437	*469-2439	469-2440		
PORTCO	002646	#41-1304	*374-172	374-191	*374-191	*374-198	*390-469	*390-469	*390-469	*390-469
		*390-504	392-546	392-547	430-1491	430-1492	436-1583	436-1584	561-4718	561-4718
		561-4718	561-4718	561-4725	561-4725	561-4725	561-4725			
PORTDI	002606	#41-1300	*57-1600	*57-1607	*84-2279	93-2520	93-2520	93-2535	93-2542	366-23
		408-920	561-4719	561-4719	561-4719	561-4719				
PSW	= 177776	#12-261	*61-1651	*61-1652	*61-1658	*68-1868	*68-1881	*68-1898	*68-1920	*70-1969
		*136-3332	*147-3630	*174-4225	*176-4251	*178-4284	*183-4357	*186-4391	*274-5959	*291-6345
		*291-6347	*316-6699	*318-6740	*320-6791	*320-6807	*336-7087	*345-7251	*347-7291	*347-7309
		*349-7372	*352-7466	*354-7522	*356-7595	*356-7602	*358-7671	*366-22	*366-30	*366-59

CEMKAB
SYMBOL CROSS REFERENCE
SYMBOL VALUE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 25
CREF V01

F 6

SEQ 1105

SYMBOL	VALUE	REFERENCES								
SBFLA	002022	#39-1127	105-2773	*108-2796	*150-3671	*237-5282				
SBEMSK	002350	#39-1210	*207-4693	*207-4694	207-4709	207-4711	207-4759	*207-4761	*207-4761	*210-4788
		*210-4789	210-4794	210-4796	210-4806	210-4841	*210-4843	*210-4843	*213-4866	*213-4867
		213-4877	213-4879	213-4883	213-4885	215-4912	*215-4914	*215-4914	*218-4938	*218-4939
		218-4949	218-4951	218-4955	218-4957	218-4993	*218-4995	*218-4995	*221-5015	*221-5016
		221-5026	221-5028	221-5032	221-5034	221-5070	*221-5072	*221-5072	*224-5094	*224-5095
		224-5104	224-5106	224-5110	224-5112	226-5150	*226-5152	*226-5152		
SBETES	017374	132-3247	#136-3306							
SCOPE	= 000004	#12-260	65-1689	66-1796	68-1833	70-1956	72-2031	100-2710	103-2730	103-2740
		183-4363	186-4397							
SDPAR0	= 172260	#12-429	155-3828	166-4078	166-4080	168-4112	194-4501	254-5623		
SDPAR7	= 172276	#12-436								
SDPDR7	= 172236	#12-416								
SECOND	= 011661	#14-559	90-2391	390-481	390-527	409-972	409-999	409-1015	413-1081	417-1193
		421-1264	421-1286	424-1328	424-1350	426-1400	428-1441	430-1477	436-1602	438-1635
		440-1700	442-1725							
SEEDHI	002714	#43-1321	*48-1449	168-4104	*168-4119	388-439	494-3304	494-3311	*494-3316	
SEEDLO	002716	#43-1322	*48-1450	168-4103	*168-4118	494-3303	494-3309	*494-3315		
SELONL	002002	#39-1116	103-2733	103-2748	132-3227	325-6910	*362-7760	*362-7765		
SETMAR	043212	114-2903	118-2980	125-3113	127-3157	#332-6989				
SETPAT	042762	114-2901	118-2978	#328-6936						
SHALT	004420	#46-1412								
SHUTUP	043554	108-2813	336-7054	#336-7062						
SIDE	002422	#39-1231	*132-3234	132-3268	*132-3271	132-3271	145-3506	145-3522	145-3538	145-3554
		145-3582	145-3598	145-3614	147-3647	147-3648	*163-3961			
SIDEOK	021066	132-3244	#147-3624							
SIPARO	= 172240	#12-419	311-6606							
SIPAR3	= 172246	#12-422	311-6620							
SIPDRO	= 172200	#12-399	311-6608	398-663	401-748					
SIZE	= 040000	#14-558	66-1809	68-1864	68-1893	68-1918	153-3784	153-3792	153-3801	155-3818
		155-3842	155-3853	168-4108	170-4141	170-4164	172-4191	176-4247	176-4266	178-4300
		241-5408	268-5850	271-5944	316-6700	318-6741	320-6792	320-6808	352-7484	368-87
		370-107	376-249	378-269	378-290	426-1383				
SIZELO	= 177760	#12-343	80-2172	80-2174	80-2176	80-2180				
SKIPKA	002012	#39-1120	341-7172	*341-7174	*360-7712	*360-7718				
SLAVEM	061244	59-1626	417-1214	#419-1221						
SLAVEN	075774	#507-3639	569-5002	569-5003						
SLAVER	002570	#39-1285	460-2242	*484-2777	*484-2779					
SLAVES	002730	#43-1327	*57-1594	57-1596	57-1596	57-1599	76-2108	*76-2109	*78-2130	*90-2396
		*91-2449	*91-2455	91-2462	*91-2486	91-2492	93-2515	93-2543	99-2699	325-6892
		382-341	382-342	390-475	390-490	390-523	390-536	392-553	409-983	409-1026
		411-1059	413-1104	415-1117	421-1278	421-1296	421-1300	424-1342	424-1360	424-1364
		426-1393	426-1409	428-1434	428-1450	430-1472	430-1486	430-1498	432-1521	434-1549
		434-1556	434-1564	434-1568	436-1596	438-1630	438-1644	438-1655	440-1693	440-1709
		440-1712	442-1734							
SLAVE1	= 000040	#14-562	84-2257	95-2571	95-2580	279-6096	408-921			
SLAVE2	= 000100	#14-563	84-2261	95-2571	95-2581	279-6096	408-922			
SLAVE3	= 000200	#14-564	84-2265	95-2571	95-2582	279-6096	408-923			
SLAVUP	061216	86-2316	384-358	409-986	#417-1213					
SLFLAG	002656	#41-1305	*95-2578	*95-2578	*95-2578	*95-2578	*95-2579	*95-2580	*95-2581	*95-2582
		*382-340	*382-345	382-350	*382-350	390-503	561-4701	561-4701	561-4701	561-4701
SLPAT	051570	370-120	370-122	370-124	370-126	370-128	#370-130			

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
SLPAT0		051530	#368-89 #370-109 #370-119 378-271
SLPAT1		051534	368-90 370-110 #370-121 378-27
SLPAT2		051542	368-91 370-111 #370-123 378-275
SLPAT3		051550	368-92 370-112 #370-125 378-274
SLPAT4		051556	368-93 370-113 #370-127 378-275
SLPAT5		051564	368-94 370-114 #370-129
SOBK		002704	#43-1317 174-4228
SOBLEN	=	000056	174-4226 174-4231 #266-5812
SOFTPA		002744	#43-1333 *48-1453 *105-2780 178-4301
SOURCE		002402	#39-1222 *207-4704 *210-4792 *213-4881 *218-4953 *221-5030 *224-5108 *259-5648 306-6539
SPECIA		002432	#39-1235 132-3244 *138-3394 *145-3500 *145-3516 *145-3532 *145-3548 *145-3576 *145-3592
			*145-3608 163-3963
SSP	=	%000006	#12-277 *61-1655 *183-4358 *186-4392 399-675 *402-757 482-2719
ST	=	177776	#502-3572 545-4302 *545-4312
STACK	=	002000	#12-255 12-256 37-1094 37-1108 43-1318 46-1428 53-1531 91-2507 108-2809
			368-71 409-978 409-1004 409-1021 409-1028 413-1105
START		004352	37-1103 37-1105 #46-1403 78-2158 90-2387 336-7059 404-793
START1		000300	37-1098 #37-1102 37-1107
START2		000310	37-1099 #37-1104
START3		000200	#37-1098 569-5012
START4		000322	37-1100 #37-1106
STOPCP		011720	#84-2282 376-224
STOPOK		002520	#39-1265 *82-2239 457-2140 *457-2141
STRIPE		002460	#39-1249 *249-5501 249-5505 251-5526 *251-5556 251-5556
STRTDI		002364	#39-1215
SUBAAA		005532	55-1565 #57-1578
SUBAAB		005200	#53-1516
SUBAAQ		007720	68-1876 #70-1937
SUBAAR		013160	84-2256 #93-2546
SUBAAS		007002	65-1776 #66-1795
SUBAAT		011334	80-2173 80-2186 #82-2201
SUBAAU		014602	103-2764 #105-2773
SUCCE		002424	#39-1232 *132-3237 *132-3258 132-3266 *356-7623 356-7629 *356-7631 *476-2589 476-2613
			476-2620 *478-2629 *478-2631 *478-2633 *478-2635 *478-2638 *478-2640 *478-2642
SUPDOA		002360	#39-1212 *155-3824 *157-3864 *157-3874 *157-3881 *159-3892 *159-3901 *159-3909 *159-3919
			*161-3930 *161-3939 *161-3944 *163-3955 *165-4050 *165-4058 *166-4080 *166-4095 *170-4131
			*174-4232 *176-4264 *178-4289 *178-4308 186-4394 *370-135 *372-151 *374-181 *376-252
SUPD01		026042	103-2756 153-3787 153-3796 153-3807 163-3983 165-4032 168-4113 168-4117 170-4144
			170-4167 172-4193 178-4314 #183-4334 271-5947 370-138 376-255 378-294
SUPD02		026056	155-3830 155-3845 155-3856 163-3988 163-3994 163-4001 163-4007 163-4013 165-4035
			165-4039 165-4042 176-4269 #183-4336 378-297
SUPD03		026220	155-3825 157-3865 157-3875 157-3882 159-3893 159-3902 159-3911 159-3920 161-3931
			161-3940 161-3945 165-4051 165-4060 166-4096 170-4147 170-4170 178-4311 #186-4369
			352-7486 370-139 372-154 376-256
SUPD04		026234	163-4017 166-4081 166-4087 174-4233 176-4265 178-4293 #186-4370 374-182 374-186
SUPDR0		002266	#39-1179 *183-4338 183-4346 *186-4372 186-4380
SUPDR1		002270	#39-1180 183-4339 186-4373
SUPDR2		002272	#39-1181
SUPDR3		002274	#39-1182
SUPDR4		002276	#39-1183
SUPDR5		002300	#39-1184
SUPDR6		002302	#39-1185 183-4349 186-4383

CEMKAB
 SYMBOL CROSS REFERENCE
 SYMBOL VALUE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 27
 CREF V01

H 6

SEQ 1107

SYMBOL	VALUE	REFERENCES
SUPLIM	057372	#408-947 569-4997 569-4998
SUPM13	053752	#390-458 438-1663
SUPSTK	= 000740	#12-257 61-1654 183-4358 186-4392 482-2722 482-2728
SWAPAT	002764	#43-1343 *48-1452
SWR	002766	#43-1346 *51-1496 51-1498 *51-1503 *53-1525 63-1666 66-1826 70-2022 90-2370
		*90-2371 90-2372 *91-2494 97-2659 100-2712 108-2802 150-3655 150-3668 163-3984
		163-3989 163-3995 163-4002 163-4008 207-4763 210-4845 215-4910 218-4991 221-5068
		226-5148 274-5966 314-6650 314-6666 314-6687 330-6952 332-6992 349-7355 349-7383
		352-7450 354-7506 354-7511 356-7554 356-7579 356-7603 358-7655 358-7660 399-700
		401-719 409-956 *409-957 409-958 *413-1107 457-2140 457-2148 457-2164 460-2219
		460-2242 460-2245 461-2252 461-2256 465-2359 488-2952 488-2960 488-2982
SWREG	= 000176	#14-550 51-1503 63-1666 90-2371 409-957 488-2952
SW0	= 000001	#12-297 66-1826 70-2022 150-3655 150-3668 163-3989 163-3995 163-4002 163-4008
		349-7383 352-7450 354-7511 356-7554 358-7660
SW1	= 000002	#12-296
SW10	= 002000	#12-287 460-2219
SW11	= 004000	#12-286 108-2802 163-3984 207-4763 210-4845 215-4910 218-4991 221-5068 226-5148
		330-6952 332-6992 349-7355 354-7506 356-7579 358-7655
SW12	= 010000	#12-285 314-6650
SW13	= 020000	#12-284 314-6666 314-6687 460-2242
SW14	= 040000	#12-283 457-2148
SW15	= 100000	#12-282
SW2	= 000004	#12-295
SW3	= 000010	#12-294
SW4	= 000020	#12-293 274-5966
SW5	= 000040	#12-292 460-2245
SW6	= 000100	#12-291 97-2659
SW7	= 000200	#12-290 465-2359
SW8	= 000400	#12-289 457-2140
SW9	= 001000	#12-288 457-2164 461-2256
SYSS1Z	011210	80-2178 80-2182 #80-2189
SYSTID	= 177764	#12-347 57-1600 84-2272 366-16 408-913
TAG2\$	007366	68-1851 #68-1873
TAG3\$	007412	68-1872 #68-1877
TAG4\$	026136	183-4347 #183-4349
TAG70\$	067636	465-2332 #467-2369
TAG71\$	067646	465-2333 #467-2375
TAG72\$	067656	465-2334 #467-2381
TAG73\$	067722	465-2335 #467-2394
TAG74\$	067730	465-2336 #467-2400
TAG75\$	067742	465-2337 #467-2406
TAG76\$	067754	465-2338 #467-2412
TAG77\$	067776	465-2339 #469-2423
TAG78\$	070000	465-2340 #469-2428
TAG79\$	070006	465-2341 #469-2434
TAG80\$	070066	465-2342 #469-2448
TAG81\$	070100	465-2343 #469-2454
TAG82\$	070122	465-2344 #471-2472
TAG83\$	070324	465-2345 #473-2504
TAG84\$	070410	465-2346 #473-2522
TAG85\$	070502	465-2347 #475-2543
TAG86\$	070612	465-2348 #475-2567

CEMKAB
SYMBOL CROSS REFERENCE
SYMBOL VALUE
TSTDAT 002344

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 29
CREF V01

J 6

SEQ 1109

REFERENCES

#39-1209	*207-4696	*207-4697	*207-4700	*207-4701	207-4702	207-4703	207-4704	*207-4710
*207-4712	207-4715	207-4717	207-4723	207-4729	*210-4790	*210-4791	210-4792	*210-4795
*210-4797	210-4801	210-4803	*213-4870	*213-4871	*213-4874	*213-4875	213-4881	*213-4884
*213-4886	*213-4888	*213-4890	213-4892	213-4894	*218-4942	*218-4943	*218-4946	*218-4947
218-4953	*218-4956	*218-4958	*218-4960	*218-4962	218-4965	218-4967	218-4973	218-4979
*221-5019	*221-5020	*221-5023	*221-5024	221-5030	*221-5033	*221-5035	*221-5037	*221-5039
221-5042	221-5045	221-5053	221-5058	*224-5098	*224-5099	*224-5102	*224-5103	224-5108
*224-5111	*224-5113	*224-5115	*224-5117	224-5120	224-5122	224-5129	224-5135	*259-5646
*259-5647	259-5648	*259-5664	*259-5666	*259-5668	*259-5669	*259-5671	*259-5674	*259-5675
*259-5677	*259-5679	*259-5680	*259-5682	261-5708	261-5710	448-1950	448-1955	453-2056
453-2058	561-4713	561-4713						
291-6323	#291-6344							
#10-248	136-3348	136-3367	207-4739	207-4753	210-4811	210-4823		
#65-1689								
#66-1796								
#68-1833								
#72-2031								
#100-2710								
#103-2730								
#103-2740								
#39-1289	*78-2141	*78-2149	289-6281	289-6292	*343-7217	*343-7219		
#10-171	97-2653	97-2655	97-2657	108-2806	352-7460	356-7620	356-7636	467-2375
#10-167	57-1582	57-1592	91-2474	91-2476	93-2522	93-2534	93-2541	97-2652
97-2654	97-2656	97-2658	108-2801	108-2803	132-3269	134-3296	134-3298	145-3570
274-5962	274-5963	274-5964	274-5970	274-5972	274-5973	274-5974	274-5975	274-5976
274-5977	274-5986	274-5987	274-5988	274-5989	277-6015	277-6018	277-6020	277-6029
277-6039	279-6049	279-6078	279-6089	279-6102	279-6112	279-6126	279-6135	279-6143
279-6145	314-6667	314-6688	339-7126	339-7129	339-7136	339-7141	341-7170	341-7196
343-7207	343-7211	343-7214	343-7221	343-7225	345-7234	345-7235	345-7257	345-7261
347-7273	347-7274	347-7297	347-7301	347-7304	347-7306	347-7313	349-7326	349-7327
349-7335	349-7339	349-7344	349-7349	349-7356	349-7364	349-7368	352-7455	352-7459
352-7461	352-7462	352-7476	352-7489	354-7499	354-7500	354-7507	354-7516	354-7518
354-7538	356-7553	356-7559	356-7573	356-7580	356-7589	356-7591	356-7621	356-7630
356-7637	358-7648	358-7649	358-7656	358-7665	358-7667	358-7687	360-7702	360-7711
360-7716	360-7722	360-7728	360-7734	362-7740	362-7745	362-7749	362-7759	362-7764
364-7782	402-778	408-893	457-2142	460-2221	460-2222	461-2268	463-2290	463-2310
463-2311	463-2313	463-2320	463-2322	465-2352	465-2362	467-2388	467-2414	467-2416
469-2428	469-2443	469-2454	469-2456	471-2497	473-2505	473-2514	473-2523	473-2534
475-2560	475-2568	475-2571	475-2574	475-2577	476-2585	476-2623	480-2683	480-2684
480-2692	480-2694	482-2706	482-2710	482-2712	482-2721	482-2724	482-2726	482-2730
482-2737	482-2740	482-2742	482-2746	482-2748	484-2764	484-2774	486-2849	487-2914
488-2958	488-2959	488-2961	488-2970	488-2975	488-2984	488-3002	490-3009	490-3014
490-3018	490-3020	490-3022	490-3027	491-3089	491-3098	491-3104	491-3109	491-3113
491-3118	491-3119	491-3121	491-3124	491-3128	492-3189	492-3191	492-3192	492-3193
492-3248	492-3250	492-3251	492-3252	498-3467				
#10-168	463-2295	467-2369	480-2691	480-2693	482-2711	482-2713	482-2725	482-2727
482-2741	482-2743	488-2960						
#10-169	91-2475	93-2535	93-2542	345-7254	347-7305	347-7314	356-7633	467-2400
467-2406	469-2448	469-2455	490-3015	490-3021	490-3026	490-3028		
#224-5092	226-5157							
#224-5096	226-5153							
#12-389	256-5641							

TSTRD1 037434
TSTREA = 104510
TST1 006260
TST2 007004
TST3 007160
TST4 010310
TST5 014342
TST6 014414
TST7 014456
TWOCSSR 002600
TYPDS = 104405
TYPEIT = 104401

TYPOC = 104402
TYPOS = 104403
T12A 031700
T12B 031722
UDPARO = 177660

CEMKAB
SYMBOL CROSS REFERENCE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 30
CREF V01

K 6

SEQ 1110

SYMBOL	VALUE	REFERENCES
UDPAR1	= 177662	#12-390 382-347 382-347
UDPAR7	= 177676	#12-396
UDPDR7	= 177636	#12-375 155-3822
UIPAR0	= 177640	12-378 #12-379 322-6816 380-303 382-343 382-343
UIPAR2	= 177644	#12-381 *68-1861 *68-1896 *170-4130 *178-4307 194-4510 264-5781 *271-5946 *370-134
UIPAR3	= 177646	#12-382 68-1882 170-4131 178-4308 370-135
UIPAR4	= 177650	#12-383 320-6778 320-6795 *376-251
UIPAR5	= 177652	#12-384 376-252
UIPAR6	= 177654	#12-385 *68-1847 196-4533
UIPAR7	= 177656	#12-386 *68-1848
UIPDRO	= 177600	#12-358 322-6818 398-662 401-747
UNITOP	002516	#39-1264 *76-2102 80-2167 80-2167 *97-2645 *97-2646 *97-2647 *97-2648 *97-2649
UNRELO	041604	97-2651 97-2651 112-2889 114-2923 116-2964 118-3005 121-3048 123-3097 125-3139 127-3188 172-4210
UPPFLG	002363	#318-6718 334-7035 336-7073
UP2	056154	#39-1214 *218-4968 218-4983 *218-4985 *224-5123 226-5140 *226-5142
USERMA	042240	404-790 #404-793
USESTK	= 000700	316-6698 318-6739 320-6773 #322-6815
USP	=%000006	#12-258 61-1660 376-220 482-2738 482-2744
WAITST	012034	#12-278 *61-1661 399-671 *402-761 482-2735
WAIT5	063674	84-2286 #86-2305
WAKEUP	054544	434-1546 434-1563 #436-1600
WARN1	007320	390-498 #392-558
WARN2	026522	#68-1861
WARN3	026536	#192-4465 *271-5933
WARN4	026562	#192-4470 *271-5934
WARN5	026576	#192-4482 *271-5935
WARN6	035364	#192-4487 *271-5936
WARN7	024416	#271-5946
WASDBE	= 104507	#170-4130
WASSBE	= 104476	#10-240 70-1976 70-1985 70-1993 70-2001
WAS1DB	= 104501	#10-238 150-3669
WAS1SB	= 104477	#10-241 213-4898
WHICHC	050630	#10-239 163-3990 163-3996 163-4003 163-4009
WHOBOX	070672	341-7188 343-7203 #364-7775
WOOPEN	056512	467-2394 #476-2581
WOOPS	056160	404-802 404-805 404-807 406-821 406-842 #406-850 406-853
WOOPSA	056542	398-659 #404-795
WOOPUP	056336	*404-800 *404-801 404-802 406-837 406-838 406-841 #406-853
WORST	002712	404-802 404-802 404-803 404-805 404-805 404-805 #406-820 406-842 406-843
WRITEO	002502	406-853
XXDPCH	002446	#43-1320 *48-1444 *116-2952 *118-2993 *123-3085 *127-3176 325-6907
ZERO	002426	#39-1258 *105-2779 *105-2782 178-4309 *368-85 *368-96 370-138 *376-227 376-255
ZEROS	002430	*376-258 *378-270 *378-276 *415-1181 *415-1183
\$APTHD	075236	#39-1241 *53-1536 461-2267
\$AUTO	002066	#39-1233
\$BANK	002015	#39-1234 147-3641 147-3642 249-5495 249-5499
\$BASE	075210	37-1096 #497-3436
\$BELL	003003	#39-1144 *53-1528 *53-1532 63-1565 488-2956
		#39-1123 *458-2177
		#497-3408
		#43-1353 354-7538 358-7687 460-2221

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
\$ESCAP		002454	#39-1247 *457-2171 461-2259 461-2261
\$ETABL		075154	#497-3375
\$ETEND		075236	#497-3432 497-3442
\$EXHAL		043546	#336-7058
\$FATAL		075136	*460-2241 #497-3368
\$FILLC		003002	#43-1352 408-898 543-4290
\$FILLS		002451	#39-1244 *360-7706 543-4292
\$GTSWR		073160	#488-2959 500-3486
\$HALT		067136	#461-2254
\$HALT2		075312	#498-3468
\$HIBTS		075236	#497-3437
\$HIOCT		074434	345-7237 347-7276 356-7561 *492-3184 #492-3195
\$ILLUP		056134	398-646 401-712 #402-782 402-783
\$INVAL		040510	#303-6519 500-3560
\$ITEMB		002017	#39-1125 *460-2232 463-2292
\$KERNE		037020	#286-6250 500-3497
\$KMAP		041016	#311-6628 500-3501
\$LF		003011	#43-1356 491-3128
\$LOADC		037076	#289-6275 500-3506
\$LOAD1		037146	289-6279 #289-6288
\$LPADR		002746	#43-1334 *48-1454 183-4337 *183-4347 183-4348 *183-4364 186-4371 *186-4381 186-4382
			*186-4398 *457-2166 *457-2169 457-2173
\$LPERR		002750	#43-1335 *48-1455 183-4337 *183-4348 *183-4364 186-4371 *186-4382 *186-4398 457-2166
			*457-2170 461-2257
\$MADR1		075166	#497-3395
\$MADR2		075172	#497-3399
\$MADR3		075176	#497-3402
\$MADR4		075202	#497-3405
\$MAIL		075134	#497-3366 497-3438 497-3442
\$MAMS1		075164	99-2667 #497-3388
\$MAMS2		075170	#497-3397
\$MAMS3		075174	#497-3400
\$MAMS4		075200	#497-3403
\$MBADR		075240	#497-3438
\$MNEW		074254	488-2961 #491-3144
\$MSGAD		075150	#497-3373
\$MSGLG		075152	#497-3374
\$MSGTY		075134	*461-2264 #497-3367
\$MSWR		074243	488-2959 #491-3143
\$MTYP1		075165	#497-3389
\$MTYP2		075171	#497-3398
\$MTYP3		075175	#497-3401
\$MTYP4		075201	#497-3404
\$NOTRA		075306	#498-3467 500-3482 500-3484 500-3536 500-3537 500-3538 500-3539 500-3540 500-3541
			500-3561 500-3562 500-3563 500-3564 500-3565 500-3566 500-3567 500-3568
\$NULL		002450	#39-1243 408-900
\$NWTST	=	000001	#65-1689 65-1689 #65-1689 #66-1796 66-1796 #66-1796 #68-1833 68-1833 #68-1833
			#72-2031 72-2031 #72-2031 #100-2710 100-2710 #100-2710 #103-2730 103-2730 #103-2730
			#103-2740 103-2740 #103-2740
\$OCNT		072572	*486-2821 *486-2850 #486-2863
\$OCTVL		075116	496-3333 #496-3358 496-3361
\$OCT8	=	075122	469-2443 #496-3361

CEMKAB SYMBOL	CREATED BY	MACRO	ON	3-AUG-79	AT	17:42	PAGE 33	N 6	SEQ 1113
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	CREF	V01				
\$MODE		072574	*486-2816	*486-2820	486-2825	*486-2828	*486-2839	#486-2865	
\$OVER		066502	457-2148	457-2167	#457-2172				
\$PASS		075142	*53-1519	*93-2557	*108-2799	*108-2800	108-2806	108-2831	*108-2837 159-3889 159-3898
			159-3906	159-3916	161-3927	161-3936	163-3951	166-4092	#497-3370
\$PASTM		075244	#497-3440						
\$PATMA		002014	#39-1122	402-774	402-775	*458-2178	*458-2187	*458-2190	*458-2191 458-2192 458-2193
			460-2218						
\$PER01		064574	#446-1883	500-3509					
\$PER02		064622	#446-1889	500-3510					
\$PER03		064650	#446-1895	500-3511					
\$PER04		064700	#446-1901	500-3512					
\$PER07		064762	#446-1917	500-3513					
\$PER10		065004	#446-1922	500-3514					
\$PER11		065034	#446-1927	500-3515					
\$PER12		065054	#446-1932	500-3516					
\$PER13		065076	#448-1938	500-3517					
\$PER14		065116	#448-1943	500-3518					
\$PER15		065140	#448-1948	500-3519					
\$PER16		065162	#448-1953	500-3520					
\$PER17		065202	#448-1958	500-3521					
\$PER20		065220	#448-1963	500-3522					
\$PER21		065236	#448-1968	500-3523					
\$PER22		065256	#448-1973	500-3524					
\$PER23		065274	#448-1978	500-3525					
\$PER24		065312	#448-1983	500-3526					
\$PER25		054570	#394-569	500-3527					
\$PER26		065502	#451-2020	500-3528					
\$PER27		065522	#451-2024	500-3529					
\$PER30		054762	#396-605	500-3530					
\$PER31		065720	#453-2064	500-3531					
\$PER32		066016	#455-2082	500-3532					
\$PER33		066064	#455-2091	500-3533					
\$PER34		066144	#455-2102	500-3534					
\$PER35		066176	#455-2111	500-3535					
\$PWRDN		055160	48-1463	#398-638	402-777				
\$PWRUP		055536	399-704	#401-711	406-849				
\$QUES		003007	#43-1354	488-3002	491-3121				
\$RAND		074720	388-438	#494-3302					
\$RDCHR		073602	#491-3053	500-3489					
\$RDDEC		074436	#492-3210	500-3492					
\$RDLIN		073722	#491-3080	500-3490					
\$RDOCT		074266	#492-3159	500-3491					
\$READC		037220	#291-6304	500-3507					
\$RESRE		074662	#493-3283	500-3495					
\$SAVRE		074624	#493-3272	500-3494					
\$SAVR6		056140	*399-702	401-714	*401-715	*401-716	#402-784	406-832	
\$SCOPE		066232	48-1457	#457-2129					
\$STN	=	000001	#65-1689	#66-1796	#68-1833	#72-2031	#100-2710	#103-2730	#103-2740
\$SVLAD		066466	457-2156	457-2163	#457-2169				
\$SWR	=	163000	#8-131	65-1689	66-1796	68-1833	72-2031	100-2710	103-2730 103-2740
\$SWREG		075156	53-1525	#497-3379					
\$TESTN		075140	*460-2218	#497-3369					

CEMKAB
SYMBOL CROSS REFERENCE

CREATED BY MACRO ON 3-AUG-79 AT 17:42

PAGE 34
CREF V01

B 7

SEQ 1114

SYMBOL	VALUE	REFERENCES								
\$TKB	002774	#43-1349	274-5958	274-5994	349-7371	349-7432	352-7465	352-7475	354-7521	354-7543
		356-7594	356-7612	358-7670	358-7692	488-2936	488-2966	490-3037	491-3057	491-3063
\$TKS	002772	#43-1348	274-5960	274-5993	349-7373	349-7430	352-7467	352-7473	354-7523	354-7542
		356-7596	356-7611	358-7672	358-7691	488-2934	488-2964	490-3035	491-3055	491-3061
\$TN	= 000010	#8-132	65-1689	65-1689	#65-1689	66-1796	66-1796	#66-1796	68-1833	68-1833
		#68-1833	72-2031	72-2031	#72-2031	100-2710	100-2710	#100-2710	103-2730	103-2730
		#103-2730	103-2740	103-2740	#103-2740					
\$TPB	003000	#43-1351	408-935							
\$TPFLG	002452	#39-1245	*53-1521	408-877						
\$TPS	002776	#43-1350	408-933							
\$TRAP	075252	48-1461	#498-3452							
\$TRAP2	075274	#498-3463	500-3478							
\$TRPAD	075314	498-3457	#500-3478							
\$STSM	075242	#497-3439								
\$STSTRD	037310	#291-6322	500-3559							
\$TTYIN	074200	491-3082	491-3083	491-3101	491-3119	491-3133	#491-3137			
\$TYPDS	072576	#487-2877	500-3483							
\$TYPE	056752	#408-877	500-3479							
\$TYPEC	057076	408-897	408-904	#408-907	484-2768	488-2988				
\$TYPEX	057360	408-940	408-942	#408-945						
\$TYPOC	072374	#486-2819	500-3480							
\$TYPON	072410	486-2818	#486-2821							
\$TYPOS	072350	#486-2814	500-3481							
\$UNIT	075146	*108-2835	*457-2132	#497-3372						
\$UNITM	075246	#497-3441								
\$USWR	075160	108-2831	#497-3380							
\$VECT1	075204	#497-3406								
\$VECT2	075206	#497-3407								
\$WASDB	040146	#299-6441	500-3551							
\$WASSB	040002	#297-6406	500-3549							
\$WAS1D	040262	#299-6467	500-3552							
\$WAS1S	040116	#297-6432	500-3550							
\$XTSTR	066370	#457-2150								
\$ZAP42	014760	#108-2809	461-2271							
\$OFILL	072573	*486-2815	*486-2819	486-2829	#486-2864					