

IDENTIFICATION

PRODUCT CODE: AC-9045F-MC
PRODUCT NAME: CZQMCFO 0-124K MEM EXER 16K
PRODUCT DATE: 15-FEBRUARY-1978
MAINTAINER: DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1975, 1978 by Digital Equipment Corporation

The following are trademarks of Digital Equipment Corporation:

DIGITAL	PDP	UNIBUS	MASSEUS
DEC	DECUS	DECTAPE	

REVISION HISTORY

Revision A:	May 1975
Revision B:	October 1975
Revision C:	October 1976
Revision D:	June 1977
Revision E:	December 1977
Revision F:	February 1978

TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION.
1.1	Program Purpose (Abstract)
1.2	System Requirements
1.3	Related Documents and Standards
1.4	Diagnostic Hierarchy Prerequisites
1.5	Assumptions
2.0	OPERATING INSTRUCTIONS
2.1	Loading and Starting Procedure
2.2	Special Environments
2.3	Program Options
2.4	Execution Times
3.0	ERROR INFORMATION
3.1	Error Reporting
3.2	Error Halts
4.0	PERFORMANCE AND PROGRESS REPORTS
5.0	DEVICE INFORMATION TABLES
5.1	CORE PARITY REGISTER
5.2	MOS PARITY REGISTER
5.3	MSII-K CSR
6.0	SUB-TEST SUMMARIES
6.1	Section 1: Address Tests
6.2	Section 2: Worst Case Noise Tests
6.3	Section 3: Instruction Execution Tests
6.4	Section 4: MOS Tests
6.5	Special Toggle in Tests
7.0	PROGRAM FUNCTIONAL FLOW CHARTS
8.0	PROGRAM LISTING

1.0 GENERAL PROGRAM INFORMATION.**1.1 Program Purpose (Abstract)**

This program has the ability to test memory from address 000000 to address 757777. It does so using:

- A. Unique addressing techniques
- B. Worst case noise patterns, and
- C. Instruction execution thruout memory.

There is also a special routine to type out all unibus address ranges which do not timeout, as well as two(2) toggle in address tests provided in section 6.1 of this document.

The intent of this program is to test as comprehensively as possible all memcry systems manufactured by DEC without concentrating on any one system. Although the tests relate to general designs they may be complete for certain systems. E.G. Any core memory from the 8K MM11-L on up need not have any other addressing or worst case patterns run but in order to completely test the MS11-K MOS memory another diagnostic is required. This test is also not intended to be a 100% test of the memory. Other tests that do I/O may find memory problems that this test is unable to.

1.2 System Requirements**A. Hardware Requirements**

PDP11 family processor with a minimum of 16K of memory.
optional...
Any parity memory control module.
KT11 memory management.

B. Software Requirements

The smallest unit of memory this program will recognize is 4K. If any address in a 4K bank causes a time out trap, that entire bank of memory is ignored by the program.

The program is designed to exercise the vector portion of memory (locations 0-776) in exactly the same manner as the rest of memory. To make this possible, without requiring memory management, no software traps are used in the program. This means that if memory management is not available or is disabled (M12=1), if the program is relocated out of bank 0, if location 0-776 are selected for test, and if an unexpected hardware trap occurs, the results will be unpredictable.

The program has the proper interface code to allow running under the automated manufacturing test line system - ACT11 and APT.

1.3 Related Documents and Standards

- A. Programming Practices - Document No. 175-003-009-01
- B. PDP-11 MAINDEC SYSMAC Package - MAINDEC-11-DZQAC-C2-D
- C. The applicable Memory System Maintenance Manual
- D. The applicable Circuit Schematics

1.4 Diagnostic Hierarchy Prerequisites

Before running this program, a CPU diagnostic should be run to verify the functionality of the processor and PDP-11 instruction set.

If memory management is to be used, then the KT11 diagnostic should also be run before this program.

PDP-11/20 - MAINDEC-11-DZQKC
PDP-11/34 - MAINDEC-11-DFKTH
PDP-11/-J - MAINDEC-11-DBQEA
OR MAINDEC-11-DCQKC
PDP-11/45 - MAINDEC-11-DCQKC
PDP-11/60 - MAINDEC-11-DQKDA
KT11-C - MAINDEC-11-DCKTA THRU DCKIF
KT11-D - MAINDEC-11-DBKTA THRU DBKTF

1.5 Assumptions

This program assumes the correct operation of the CPU and, if used, the memory management option.

2.0 OPERATING INSTRUCTIONS

2.1 Loading and Starting Procedures

2.1.1 Load the program using any standard absolute loader.

2.1.2 Starting address 200:

Normal program execution.

2.1.3 Starting address 204:

Allows the operator to input, via teletype conversation, first and last addresses to be exercised, and a data pattern to be used in tests 6 and 7.

2.1.4 Starting Address 210:

Restart program using previously selected parameters.

2.1.5 Starting Address 214:

Restore loaders and halt. This routine is capable of relocating the program back to banks 0 and 1 if the program was halted while running the top two banks of memory. There are special procedures required for this situation.

- A. If memory addresses 0-1000 have not been exercised, either through parameter selection (SA=204) or by running with SW05=1, then:

Load Address 214,
Press START.

- B. If running without memory management, then:

Load Address <214+relocation factor>
(Relocation factor is typed when the program is relocated),
Press START.

- C. If running with memory management and the unibus has not been initialized (via reset instruction, start switch, etc.), then:

Load Address 777707 (PC)
Deposit 214
Press CONTInue

- D. If running with memory management and the unibus has been initialized:

Load Address 772340 (KIPAR0)
Deposit <(relocation factor)/100>
(Example: Relocation factor=540000, then
deposit 005400)
Load Address 777572 (SR0)
Deposit 000001
Load Address 777707 (PC)
Deposit 214
Press Continue

2.1.6 Starting address 220:

Byte address memory map typeout routine. This routine performs DATI, DATIP, DATO, and DATOB on all possible addresses, and types the ranges of addresses which do not cause a timeout trap.

2.2 Special Environments

If the program is run in quick verify mode under ACT11 or APT11 the program is done after the first pass. Also, the

program does not relocate to test the lower 8K of memory.

2.3 Program Options

SW15 = 1 OR UP....	HALT ON ERROR
SW14 = 1 OR UP....	LOOP ON TEST
SW13 = 1 OR UP....	INHIBIT ERROR TYPEOUT
SW12 = 1 OR UP....	INHIBIT MEMORY MANAGEMENT (INITIAL START ONLY)
SW11 = 1 OR UP....	INHIBIT SUBTEST ITERATION
SW10 = 1 OR UP....	RING BELL ON ERROR
SW9 = 1 OR UP....	LOOP ON ERROR
SW8 = 1 OR UP....	LOOP ON TEST IN SWR<4:0>
SW7 = 1 OR UP....	INHIBIT PROGRAM RELOCATION
SW6 = 1 OR UP....	INHIBIT PARITY ERROR DETECTION

NOTE: With parity error detection enabled, a memory failure while running the worse case noise tests (non-parity) can cause a parity error. The error printout on a parity error does not type the good data. Thus a bit drop or pickup will not be typed as such. It is best to run the program for 1 pass with parity disabled, then, restart the program with parity enabled.

SW5 = 1 OR UP....	INHIBIT EXERCISING VECTOR AREA (LOCATIONS 0-1000).
-------------------	--

2.4 EXECUTION TIMES

Execution time is dependent on type of memory, and amount of memory. Worse case run times with 900ns memorys are:

- a. For Non-Parity Memory
 - First Pass: 65 seconds for first 16k + 15 seconds for each additional 16k.
 - Full Pass: 3 minutes 40 seconds for first 16k + 3 minutes for each additional 16k.
 - Iteration Inhibited: same as first pass
- b. For Parity Memory
 - First Pass: 1 minute 40 seconds per 16k.

Full Pass: 8 minutes per 16K

Iteration Inhibited: same as first pass

3.0 ERROR INFORMATION

3.1 Error Reporting

There are a total of 31(8) types of error reports generated by the program. Some of the key column heading mnemonics are described below for clarity:

PC = Program Counter of error detection code.
(V/PC=P/PC)

V/PC = Virtual Program Counter. This is where the error detection code can be found in the program listing.

P/PC = Physical Program Counter. This is where the error detection code is actually located in memory.

TRP/PC = Physical Program Counter of the code which caused a trap.

MA = Memory Address

REG = Parity REGISTER address.

PS = Processor Status word.

IUT = Instruction Under Test.

S/B = What contents Should Be.

WAS = What contents WAS.

3.2 Error Halts

With the 'HALT ON ERROR' switch (SW15) not set there are several programmed 'HALTS' in the program:

- A. In the error trap service routine for unexpected traps to vector 4. This one will occur if a 2nd trap to 4 occurs before the error report for the first has had a chance to be printed out.
- B. In the relocation routine if the program is being relocated back to the first 8K of memory and the program code was not able to be transferred properly.
- C. In the case of error reporting and there is no terminal to allow the information transfer.

- D. In the power fail routine if the power up sequence was started before the power down sequence had a chance to complete itself.
- E. In the Memory mapping routine or any of the address control routines, failures to find a meaningful map.

4.0 PERFORMANCE AND PROGRESS REPORTS

Not applicable

5.0 DEVICE INFORMATION TABLES

The following is a picture view of a parity control status registers, which will show bit assignments and definitions, to provide a handy reference:

5.1 CORE PARITY REGISTER

```

-----
I I I I I I I I I I I I I I I I I I
!PE! ! ! ! ADDRESS ! ! !WP! !AE!
I I I I I I I I I I I I I I I I I I
-----
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

```

Bit assignments are defined as follows:

- | | | |
|-----------|--------------------|--|
| BIT15 | PARITY ERROR | |
| BITS 11-5 | ERROR ADDRESS | HIGH ORDER ADDRESS BITS OF ADDRESS OF PARITY ERROR (BITS 17-11 OF ADDRESS) |
| BIT02 | WRITE WRONG PARITY | NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET |
| BIT00 | ACTION ENABLE | NO ACTION WHEN CLEAR TRAP TO VECTOR 114 WHEN SET |

5.2 MOS PARITY REGISTER

```

-----
I I I I I I I I I I I I I I I I I I
!PE! ! ! ! ! ! ! ! ! ! ! ! !WP! !AE!
I I I I I I I I I I I I I I I I I I
-----
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

```

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	PARITY ERROR	
BIT02	WRITE WRONG PARITY	NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET
BIT00	ACTION ENABLE	NO ACTION WHEN CLEAR TRAP TO VECTOR 114 WHEN SET

5.3 MS11-K CSR

```

-----
I I I I I I I I I I I I I I I I I
!DE! !SI! ! ADDRESS !SE! !P! !DC! !EC! !EE!
I I I I I I I I I I I I I I I I I
-----
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
    
```

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	DOUBLE ERROR	
BIT 13	SET INHIBIT MODE	WHEN THIS BIT IS SET TO A 1, IT ENABLES THE INH MODE POINTER TO INHIBIT EITHER THE FIRST OR SECOND 16K FROM EVER GOING INTO THE DIAG. CHECK OR ECC DISABLE MODE.
BITS 11-5	ERROR ADDRESS	WHEN BIT02 CLEARED CONTAINS HIGH ORDER BITS OF ADDRESS OF PARITY ERROR(BITS 17-11); WHEN BIT02 SET CONTAINS CHECK BITS FOR ECC.
BIT04	SINGLE ERROR	SET WHENEVER SINGLE ERROR OCCURS
BIT03	INHIBIT MODE POINTER	THE INHIBIT MODE POINTER WORKS IN CONJUNCTION WITH THE SET INHIBIT MODE BIT. WHEN BIT 13 IS SET TO A 1, A 16K PORTION OF MEMORY IS INHIBITED FROM OPERATING IN THE ECC DISABLE MODE OR

DIAGNOSTIC CHECK MODE.
THE INHIBIT MODE
POINTER INDICATES
WHICH 16K IS BEING
INHIBITED,,BIT 3 =1

THE SECOND 16K OF
MEMORY IS INHIBITED.
WHEN BIT 13 IS SET TO
A 0, BIT 3 BECOMES
INOPERATIVE.

BIT02	DIAGNOSTIC CHECK A	WHEN SET ENABLES READ-WRITE OF CHECK BITS(SEE BITS 11-5)
BIT01	DISABLE ERROR CORRECTION	WHEN SET NO ERROR CORRECTION TAKES PLACE
BIT00	DOUBLE ERROR ENABLE	WHEN SET ENABLES TRAP TO VECTOR 114 ON DOUBLE ERROR.

6.0 SUB-TEST SUMMARIES

6.1 Section 1: Address Tests.

These tests verify the uniqueness of every memory address.

TEST 1 Writes and reads the value of each memory Word Address into that Memory location. After all memory has been written, all locations are checked again.

TEST 2 Writes the byte value of each address into that byte location and checks it.

TEST 3 Writes the complement of each word address into that location and checks it.

TEST 4 Writes the 4K bank number into each byte of that bank and checks it.

TEST 5 Writes the complement of the bank number into each byte of that bank and checks it.

6.2 Section 2: Worst Case Noise Tests.

These are intended to apply maximum stress to the various types of PDF-11 core memories.

TEST 6 and TEST 7 Are supplied to allow the operator to select a single word data pattern (SA=204) and SCOPE on

either the writing (DATO) in TEST 6 or the reading (DATI) in TEST 7 of that data.

TEST 10 Writes and then checks a series of single word patterns which are designed to stress parity memory.

TEST 11 Writes all memory with 1's in every bit and then "Ripples" a "0" through it.

TEST 12 Writes all memory with 0's in every bit and then "Ripples" a "1" through it.

TEST 13,14,15, AND 16 Write a pattern which complements when address BIT 3 XOR BIT 9 complements.

TEST 17 Writes wrong parity in each byte of memory and checks that the parity detection logic works. This test is skipped for non-parity memory.

TEST 20 Write "random" program code through memory and checks it.

6.3 Section 3: Instruction Execution Tests.

This group of tests place instructions in the memory under test, then executes the instructions, and finally, checks that they executed correctly.

TEST 21 Executes an instruction which does a DATI and a DATO on the memory under test.

TEST 22 Executes an instruction which does a DATI and a DATOB on the low byte of memory under test.

TEST 23 Executes an instruction which does a DATI and a DATOB on the high byte.

TEST 24 Executes an instruction which does a DATIP and a DATO.

TEST 25 Executes an instruction which does a DATIP and a DATOB on the low byte.

TEST 26 EXECUTES AN INSTRUCTION WHICH DOES A DATIP and a DATOB on the high byte.

6.4 Section 4: Mos Tests

TEST 27 -Writes a pattern of 000377 through memory, then compliments it addressing downward, compliments the new pattern addressing upward, compliments the third pattern addressing upward and finally compliments this new AB patterns addressing downward.

TEST 30-31 Write a checkerboard through memory then stalls for 2 seconds and then verifies no data has changed.

6.5 Special Toggle In Tests

6.5.1 Toggle-in-program #1

The following is a toggle in memory address test. This test is useful when an address selection failure is suspected involving the first 8K of memory. This program writes the value of each address into itself starting with the lower limit and continuing to the upper limit. After all addresses have been written each address is checked for the correct contents starting with the upper limit and continuing to the lower limit.

LOCATION	CONTENTS	MNEMONIC	COMMENT
10	012700	MOV #50,R0	;GET FIRST ADDRESS
* 12	000050		;TO TEST ;(EXAMPLE START ADDRESS)
14	010001	MOV R0,R1	;SAVE IN R1
16	020037	1\$: CMP R0,@#SWR	;CHECK UPPER LIMIT
20	177570		; (IN SWITCH REGISTER)
22	001403	BEQ 2\$;BRANCH IF AT UPPER LIMIT
24	010010	MOV R0,(R0)	;LOAD VALUE INTO ADDRESS
26	005720	TST (R0)+	;STEP TO NEXT ADDRESS
30	000772	BR 1\$;LOOP UNTIL DONE
32	010004	2\$: MOV R0,R4	;SAVE UPPER LIMIT
34	020001	3\$: CMP R0,R1	;CHECK IF AT LOWER LIMIT
* 36	001767	3EQ 1\$;BRANCH IF DONE
40	024000	CMP -(R0),R0	;CHECK DATA WRITTEN
42	001774	BEQ 3\$;BRANCH IF OK
44	000000	HALT	;ERROR
46	000772	BR 3\$;LOOP BACK

After toggling the program LA=10**set upper limit**, start

NOTES: The upper limit address obtained from the switch register may be changed during program operation. However occasionally the program may halt because of 'SWITCH BOUNCE'. (The best procedure when changing limits is to stop the program make the change and continue.) The lower limit address (12) may be patched to any desired address.

6.5.2 Toggle-in-Program #2

The following is also a toggle in program to be used with toggle-in-program #1 for more complete address testing. This program writes the complement value of each address into itself starting with the upper limit and continuing to the lower limit. After all addresses have been written each address is checked for the correct contents starting with the

lower limit address and continuing to the upper limit.
Toggle in the following patches to the program above.

These are the patches to toggle-in-program #1:

LOCATION	CONTENTS	MNEMONIC	COMMENT
12	100		;CHANGE LOWER LIMIT
36	001404	BEQ 4\$;BRANCH TO PROGRAM #2

These are the additions to toggle-in-program #1:

LOCATION	CONTENTS	MNEMONIC	COMMENT
50	010402	4\$: MOV R4,R2	;GET UPPER LIMIT
52	005142	5\$: COM -(R2)	;COMPLEMENT ADDRESS
54	020201	CMP R2,R1	;CHECK IF AT LOWER LIMIT
56	001375	BNE 5\$;LOOP UNTIL DONE
60	020204	6\$: CMP R2,R4	;CHECK IF AT UPPER LIMIT
62	001755	BEQ 1\$;GO TO PROGRAM 1 IF DONE
64	010203	MOV R2,R3	;GET VALUE OF ADDRESS
66	005103	COM R3	;COMPLEMENT VALUE
70	020322	CMP R3,(R2)+	;CHECK ADDRESS
72	001772	BEQ 6\$;BRANCH IF OK
74	000000	HALT	;ERROR
76	000770	BR 6\$;GO CHECK NEXT ADDRESS

7.0 PROGRAM FUNCTIONAL FLOW CHARTS
Attached

8.0 PROGRAM LISTING
Attached

FLOW CHART

CZQMCFO 0-124K MEM EXER 16K

COPYRIGHT 1978
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

TABLE OF CONTENTS

PAGE 01	DEFINITIONS, TRAP CATCHER, STARTING ADDRESSES.
PAGE 02	RESTART AND RESTORE ROUTINES
PAGE 04	POWER FAIL ROUTINES
PAGE 05	COMMON TAGS
PAGE 06	SETUP
PAGE 08	MAP MEMORY
PAGE 09	MEMORY BYTE MAP ROUTINE
PAGE 12	MAP PARITY REGISTERS
PAGE 13	MAP PARITY MEMORY
PAGE 14	TEST PARITY REGISTERS
PAGE 15	USER PARAMETER SELECTION SECTION
PAGE 16	START1: START OF PASS
PAGE 17	SECTION 1: ADDRESS TESTS. TEST 1
PAGE 18	TEST 2
PAGE 19	TEST 3
PAGE 20	TEST 4
PAGE 21	TEST 5
PAGE 22	SECTION 2: WORSE CASE NOISE TESTS. TEST 6
PAGE 23	TEST 7
PAGE 24	TEST 10
PAGE 25	TEST 11
PAGE 26	TEST 12
PAGE 27	TEST 13: 3 XOR 9
PAGE 29	TEST 14: 3 XOR 9
PAGE 31	TEST 15: 3 XOR 9 (FOR PARITY)

TABLE OF CONTENTS

PAGE 33	TEST 16: 3 XOR 9 (FOR PARITY)
PAGE 35	TEST 17: PARITY BYTE TEST
PAGE 39	TEST 20
PAGE 40	TEST 21: EXECUTE DATI, DATO
PAGE 41	TEST 22: EXECUTE DATI, DATOB (LO BYTE)
PAGE 42	TEST 23: EXECUTE DATI, DATOB (HI BYTE)
PAGE 43	TEST 24: EXECUTE DATIP, DATO
PAGE 44	TEST 25: EXECUTE DATIP, DATOB (LO BYTE)
PAGE 45	TEST 26: EXECUTE DATIP, DATOB (HI BYTE)
PAGE 46	TEST 27: MARCHING 1'S AND 0'S
PAGE 49	TEST 30: MOS REFRESH TEST
PAGE 51	TEST 31: MOS REFRESH TEST
PAGE 53	DONE
PAGE 54	END OF PASS
PAGE 55	MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES
PAGE 57	SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS
PAGE 58	RELOCATION SUBROUTINES
PAGE 60	PARITY ROUTINES
PAGE 62	SPECIAL PRINTOUT ROUTINES
PAGE 63	SYSMAC AND STANDARD UTILITY ROUTINES

CZQMCFO 0-124K MEM EXER 16K
DEFINITIONS, TRAP CATCHER, STARTING ADDRESSES.

DECFLD VER 00.07 20-FEB-78 07:58 PAGE 01

* SWITCH SETTINGS AND *
* BASIC DEFINITIONS *
* *
* *****

* TRAP CATCHER AND *
* STARTING ADDRESSES *
* *
***** .****
.=0

SA=210 . =300

```

*****
**RESTAR **
*****
RESTAR I
*****
* SET RESTART *-----
* FLAG (R5=0) *              I
* *              I
** * * *              I
*****              I
    
```

SA=214

```

*****
**RESTOR **
*****
I
*****
* SET RESTORE *
* FLAG (R5=PC) *
* *
*****              I
I<-----
    
```

```

REST1 V
*****
* *
* SETUP STACK *
* *
*****
I
V
    
```

```

HAS MEMORY BEEN \NO
  MAPPED?       /
-----> *STARTA *
*****
    
```

```

I YES
REST2 V
    
```

```

/MEMORY MANAGEMENT \NO
  AVAILABLE?       /
----->
I
V
*****
*RESET SP AND JUMP TO *
* RELOCATED PROGRAM *
*****
I
I
    
```

```

*****
* SET UP MEMORY MGMT. *
* MAP PROGRAM INTO *
* VIRTUAL BANKS 0 & 1 *
*****
I<-----
I
I
V
    
```

```
-----
PROGRAM MAP \YES
/POINTING TO BANKS 0 \-----
& 1?
-----
I NO
V RELO(59)
*****
** RELOCATE PROGRAM TO **
** BANKS 0 & 1 **
** **
** **
I<-----
V
-----
RESTART FLAG \YES
(R5=0)? ----->*START1(16)*
-----
I NO
V RESLDR(59)
*****
** RESTORE LOADERS **
** **
** **
*****
I
V
*****
**HALT **
*****
```

.=572

```
*****  
**$PWRDN **  
*****  
  I  
  V  
*****  
* $ILLUP -> VECTOR *  
*   SAVE REGISTERS *  
* $PWRDN -> VECTOR *  
*****  
  I  
  V  
*****  
**HALT **  
*****
```

```
*****  
**$PWRUP **  
*****  
  I  
  V  
*****  
* WAIT LOOP FOR TTY *  
* RESTORE REGISTERS *  
* $PWRDN -> VECTOR *  
*****  
  I  
  V $PRINT(63)  
*****  
  / TYPE POWER FAIL /  
  / MESSAGE /  
*****  
  I  
  V  
*****  
**RETURN **  
*****
```

```
*****  
**$ILLUP **  
*****  
  I  
  V  
*****  
**HALT **  
*****
```

.#1100

```
*****  
* STANDARD 'SYSMAC' *  
* COMMON TAGS *  
*****
```

```
*****  
* APT MAILBOX AND *  
* ETABLE *  
*****
```

```
*****  
*COMMON TAGS FOR THIS *  
* PROGRAM *  
*****
```

```
*****  
* RELATIVE ADDRESSING *  
* TABLE, ERROR DATA *  
* POINTER *  
*****
```

```
*****  
* MEMORY PARITY WORSE *  
* CASE PATTERNS TABLE *  
*****
```

```
*****  
* MEMORY PARITY *  
*REGISTER ADDRESS AND *  
* MAP TABLE *  
*****
```

```
*****  
*ERROR MESSAGE POINTER*  
* TABLE *  
*****
```

SA=204

```
*****  
* SELECT *  
*****  
I  
*****  
* SET FLAG FOR *  
* SELECTING *  
* PARAMETERS *  
*****  
I
```

SA=200

```
*****  
**START **  
*****  
I  
*****  
*CLEAR FLAG FOR *  
* SELECTING *  
* PARAMETERS *  
*****  
I
```

```
I  
STARTA V  
*****  
* CLEAR COMMON TAGS *  
* INIT SP INIT PF *  
* VECTOR *  
*****  
I  
V
```

```
*****  
* SET UP SOFTWARE *  
* SWITCH REGISTER IF NO *  
* HARDWARE SWR *  
*****  
I  
V
```

```
*****  
/TYPE PROGRAM TITLE/  
(ONCE ONLY)  
*****  
I  
V
```

```
-----  
HAS PROGRAM RELOCATED? YES  
----->*****  
*RESTAR(02)*  
*****
```

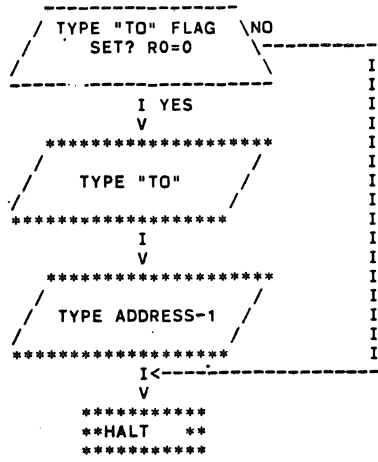
```
I NO  
V  
*****  
** SAVE LOADERS **  
*****
```

```
I  
I  
I  
V
```

```
-----  
/ CLEAR 'MMAVA'. \ YES  
MEMORY MANAGEMENT -----  
EXITS AND DESIRED? \  
I NO I  
I V  
I *SET UP MEM MGMT. SET *  
I * 'MMAVA' FLAG TYPE *  
I * 'KT11 AVAILABLE' *  
I *-----*  
I I  
I-----  
I  
I  
V  
*****  
*  
*TURN OFF CACHE *  
*  
*****  
I  
I  
V
```



```
                                $PRINT(63)
*****
// TYPE "FROM" //
// *****
//           I
//           V $STYPAD(63)
// *****
// TYPE ADDRESS //
// *****
//           I<-----I
//           V
// *****
// * COMPLEMENT *
// * TYPEOUT FLAG *
// * (RO) *
// *****
***
//B----->I
***
// *****
// *UPDATE ADDRESS * 0
// * POINTER *
// *****
//           I
//           V NOT=0
// *****
// NO
//A<----- / 4K BANK BOUNDRY? \
// *****
//           I YES
//           V
// *****
// NO
//A<----- / MEMORY MANAGEMENT \
// *****
//           I YES
//           V
// *****
// * UPDATE MEM MGMT *
// * REGISTERS + ADDRESS *
// * POINTER *
// *****
// NO
//A<----- / END OF MEMORY? \
// *****
//           I YES
//           I<-----I
//           V
// *****
//A*
//A*
//A*
```

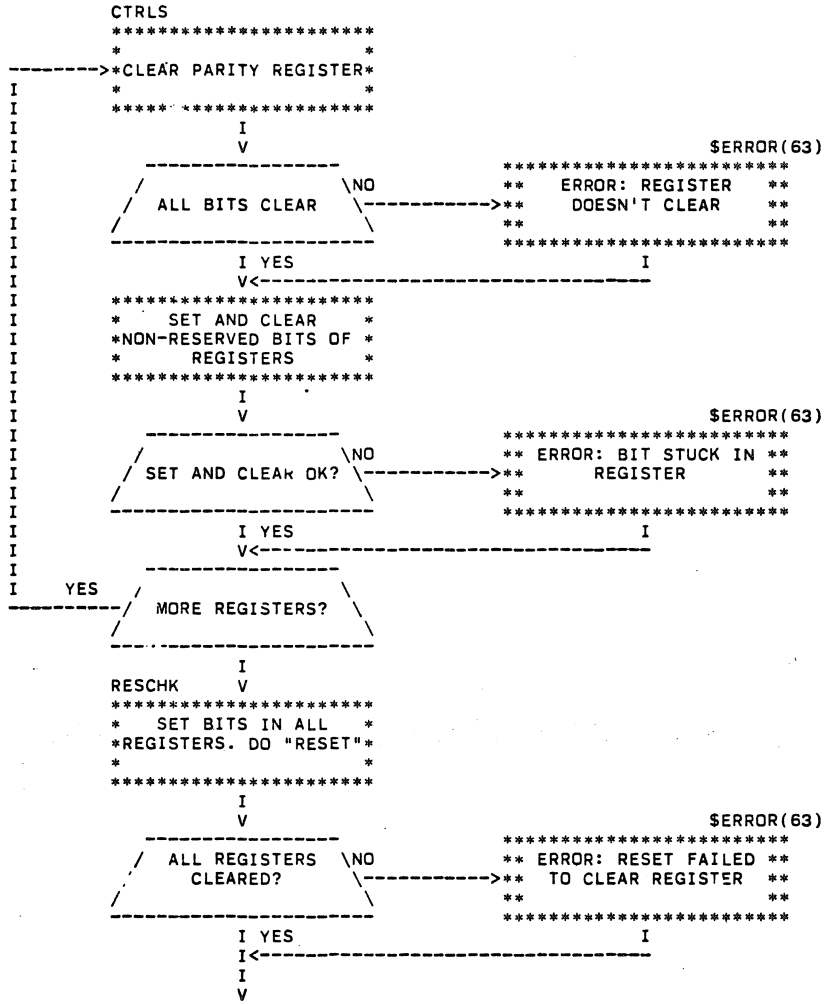


```

*****
*GMPR(08) *
*****
GMPR      I
*****
*SET UP TABLE POINTERS*
*SET UP TIMEOUT VECTOR*
*
*
*****
----->I
I          GMPRA      V
I          *****
I          * CLEAR FLAG. TEST * TIMECUT TRAP
I          * REGISTER ADR FOR *--->>>>>>>>>>>>-----
I          *          TIMEOUT *
I          *****
I          I                      GMPRB      I
I          V                      *****
I          * ADJUST SP. SET FLAG *
I          * TO INDICATE REGISTER *
I          *          NOT PRESENT *
I          *          *****
I          *          I
I          *          I
I          I<-----
I          GMPRC      V
I
I          NO /CLEAR MAP. END OF\
I          /TABLE REACHED? \
I          /-----\
I          I YES
I          V
I          *****
I          * TERMINATE SECOND *
I          * TABLE (MPRX) RESE1 *
I          * TIMEOUT VECTOR *
I          *          *****
I          I
I          V
I          /ANY PARITY\ \NO
I          /REGISTERS FOUND? \-----
I          /-----\
I          I YES
I          I
I          I
I          I
I          I
I          I
I          I
I          V
I          /-----\
I          /TYPE "NO MEMORY\
I          /PARITY REGISTERS \
I          /FOUND". /
I          /-----\
I          I
I          V
I          *****
I          *MANUAL(15)*
I          *****

```

```
*****  
* INIT ALL REGISTERS *  
* SET UP POINTERS *  
* *  
* *****  
      I  
MAPRB  V  
*****  
*WRITE WRONG PARITY IN*  
* EACH BANK OF MEMORY *  
* *  
*****  
      I  
      V  
*****  
* FIND WHICH REGISTER *  
*CONTROLS WHICH BANK. *  
* *  
*****  
      I  
TMAP  V  
*****  
*TYPE PARITY REGISTER *  
* ADDRESS *  
* *  
*****  
      I  
      V      TYPMAP(62)  
*****  
** TYPE MAP OF MEMORY **  
** CONTROLLED BY EACH **  
** REGISTER **  
*****  
      I  
      I  
      I  
      I  
      I  
      I  
      I  
      I  
      V
```



CZQMCFO 0-124K MEM EXER 16K
START1: START OF PASS

DECFL0 VER 00.07 20-FEB-78 07:58 PAGE 16

```
MANUL2
*****
* MAKE NECESSARY *
* ADJUSTMENTS TO FIRST *
* AND LAST ADDRESSES *
* *****
  I
  I
  V
  *****
  **START1 **
  *****
    I
    I
    I
START1 V
*****
* INITIALIZE EVERYTING *
*   FOR A NEW PASS   *
*                   *
*****
  I
  I
  I
  I
  V
```

```

TST1          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I             V
I             *****
I             * WRITE PHYSICAL *
I             * ADDRESS VALUE IN EACH *
I             *   WORD LOCATION   *
I             *****
I             I
I             V      MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS   **
**                   **
**                   **
IDONE
V      INITDN(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I             V
I             /-----\
I             / DOES EACH \ NO
I             / LOCATION HAVE \
I             / ADDRESS VALUE? \
I             /-----\
I             I YES
I             I<-----I
I             V      MMDOWN(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS   **
**                   **
**                   **
*****
IDONE
I
I
I
V

```

\$ERROR(63)

```

*****
**ERROR: ADDRESS VALUE **
** NOT IN LOCATION   **
**                   **
*****

```



```
TST3          INITDN(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
** *****
----->I
I             V
I             *****
I             * WRITE ONE'S *
I             * COMPLEMENT OF ADR *
I             * INTO WORD LOCATIO: *
I             *****
I             I
I             V          MMDOWN(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS         **
** *****
I             IDONE
I             V          INITMM(55)
I             *****
I             ** INITIALIZE ADDRESS **
I             **   POINTERS         **
I             ** *****
----->I
I             V
I             *****
I             / DOES EACH WORD \ NO
I             / HAVE COMPLEMENT OF \-----> **ERROR: COMPLEMENT OF **
I             / ADR. VALUE? \         **ADR. NOT IN WORD LOC.**
I             /-----\
I             I YES
I             I <----- I
I             V          MMUP(50)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS         **
** *****
I             IDONE
I             I
I             I
I             V
```

```

TST4          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I             V
I             *****
I             * WRITE BANK # VALUE *
I             * INTO EACH BYTE   *
I             * LOCATION         *
I             *****
I             I
I             V MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
----->I             POINTERS **
**                   **
*****
IDONE
V INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I             V
I             /-----/
I             / DOES EACH BYTE \ NO
I             / HAVE ITS BANK # \
I             / VALUE?         \
I             /-----/
I             I YES
I             I <----- I
I             V MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
----->I             POINTERS **
**                   **
*****
IDONE
I
I
I
V

```

\$ERROR(63)

```

*****
** ERROR: BANK # VALUE **
** NOT IN LOCATION **
**                   **
*****

```

```

TST5          INITDN(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I              V
I              *****
I              *WRITE 1'S COMPLEMENT *
I              * OF BANK NUMBER INTO *
I              *   BYTE LOCATION   *
I              *****
I              I
I              V      MMDOWN(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
I              **   POINTERS     **
I              **                   **
I              *****
I              IDONE
I              V      INITDN(55)
I              *****
I              ** INITIALIZE ADDRESS **
I              **   POINTERS         **
I              **                   **
I              *****
----->I
I              V
I              /-----/
I              / DOES EACH BYTE \ NO
I              / HAVE COMPLEMENT OF \
I              /   BANK VALUE?   \
I              /-----/
I              I YES
I              I <----- I
I              V      MMDOWN(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
I              **   POINTERS     **
I              **                   **
I              *****
I              IDONE
I              I
I              I
I              V

```

\$ERROR(63)

**ERROR: COMPLEMENT OF **
** BANK # NOT IN BYTE **
** LOC. **

```
TST10
*****
* SET UP DATA TABLE *
*   POINTER           *
*****
>I
V   CKPMER(61)
*****
**CHECK FOR ANY MEMORY **
**   PARITY ERRORS   **
**                   **
*****
I
V
** *****
*GET DATA PATTERN FROM*   END OF TABLE
*   TABLE             *
*                   *
*****
I
V   INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
>I
V
*****
*PUT DATA INTO MEMORY *
*   LOCATION           *
*                   *
*****
I
V
*****
/ DATA JUST WRITTEN \ NO
/   OK?               \
\                       /
\                       /
I YES
I <----- I
V   MMUP(56)
*****
MORE MEMORY ** UPDATE ADDRESS **
**   POINTERS   **
**                   **
*****
DONE I
*****
$ERROR(63)
** *****
**ERROR: DATA CHANGED. **
** *****
```



```

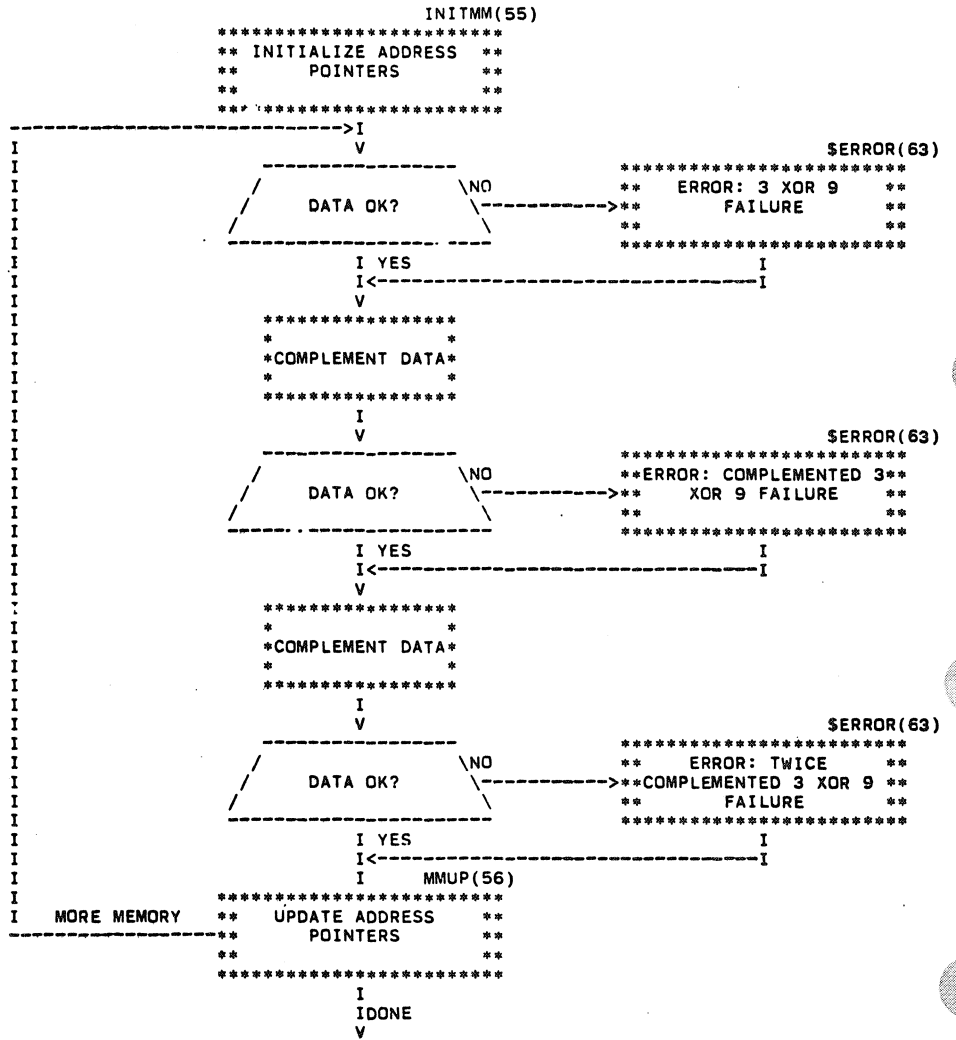
TST12          SETCON(57)
*****
**PUT ALL ZEROS IN ALL **
**      MEMORY          **
**                      **
*****
          I
          V          INITMM(55)
*****
** INITIALIZE ADDRESS **
**      POINTERS      **
**                      **
*****
----->I
I          I          ROTATE(57)
I          V          *****
I          *****
I          **SET C-BIT AND ROTATE **
I          **IT THROUGH TWO BYTES **
I          **                      **
I          *****
I          I
I          V
I          -----
I          / C-BIT SET AND 0 \ NO
I          / IN MEMORY LOCATION? \ ----->
I          -----
I          I YES
I          I<-----I
I          V          MMUP(56)
I          *****
I          MORE MEMORY ** UPDATE ADDRESS **
I          **          POINTERS          **
I          **                      **
I          *****
I          IDONE
I          I
I          I
I          V

```

\$ERROR(63)

** ERROR: ROTATING 1 **
** FAILED **
** **

```
TST13          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I              V      W3X9(57)
I              *****
I              ** WRITE 256. WORD **
I              **   BLOCKS WITH   **
I              ** 0,0,0,0,-1,-1,-1 **
I              *****
I              I
I              V      MMUP(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
**                   **
**                   **
IDONE
V      INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I              V
I              *****
I              /256. WORD BLOCKS \NO
I              WRITTEN WITH  \-----> ** ERROR: 3 XOR 9 **
I              / 0,0,0,0,-1,-1,-1 \ **   PATTERN FAILURE **
I              *****
I              I YES
I              I<-----I
I              V      MMUP(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
**                   **
**                   **
IDONE
I
I
I
I
V
```



```
TST14          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I             V      W3X9(57)
I             *****
I             ** WRITE 256. WORD **
I             **   BLOCKS WITH   **
I             ** -1,-1,-1,-1,0,0,0 **
I             *****
I             I
I             V      MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
I             **   POINTERS     **
I             **                   **
I             *****
I             IDONE
I             V      INITMM(55)
I             *****
I             ** INITIALIZE ADDRESS **
I             **   POINTERS         **
I             **                   **
I             *****
----->I
I             V
I             -----
I             /256. WORD BLOCKS \NO
I             / WRITTEN WITH \----->
I             / -1,-1,-1,-1,0,0,0 \
I             -----
I             I YES
I             I <-----I
I             V      MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
I             **   POINTERS     **
I             **                   **
I             *****
I             IDONE
I             I
I             I
I             I
I             I
I             V
```

```
*****
** ERROR(63) **
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **
**                   **
*****
```



```

TST15          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
** *****
----->I
I              V      W3X9(57)
I              *****
I              ** WRITE 256. WORD **
I              ** BLOCKS WITH 401 AND **
I              **      -1          **
I              *****
I              I
I              V      MMUP(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS         **
** *****
              IDONE
              V      INITMM(55)
              *****
              ** INITIALIZE ADDRESS **
              **   POINTERS         **
              ** *****
----->I
I              V
I              /256. WORD BLOCKS \NO
I              / WRITTEN WITH 401 \
I              / AND -1?          \
I              /----->
I              I YES
I              I<-----I
I              V      MMUP(5C)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS         **
** *****
              IDONE
              I
              I
              I
              I
              V

```

\$ERROR(63)

```

*****
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **
** *****

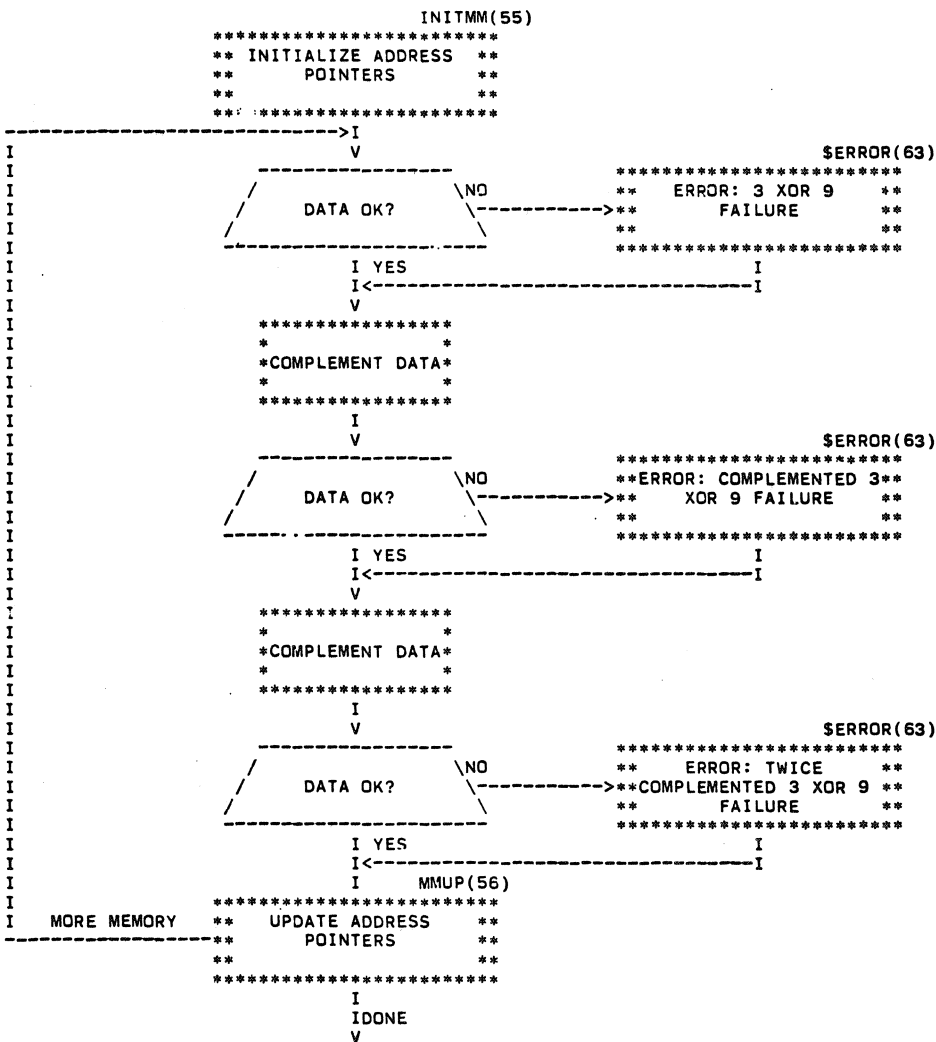
```



```
TST16          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I              V      W3X9(57)
I              *****
I              ** WRITE 256. WORD **
I              ** BLOCKS WITH -1 AND **
I              **      401          **
I              *****
I              I
I              V      MMUP(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS         **
**                   **
*****
IDONE
V      INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I              V
I              -----
I              /256. WORD BLOCKS \NO
I              /WRITTEN WITH -1 AND\----->
I              /      401?          \
I              -----
I              I YES
I              I<-----I
I              V      MMUP(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
-----**   POINTERS         **
**                   **
*****
IDONE
I
I
I
I
V
```

\$ERROR(63)

```
*****
** ERROR: 3 XOR 9 **
**   PATTERN FAILURE **
**                   **
*****
```

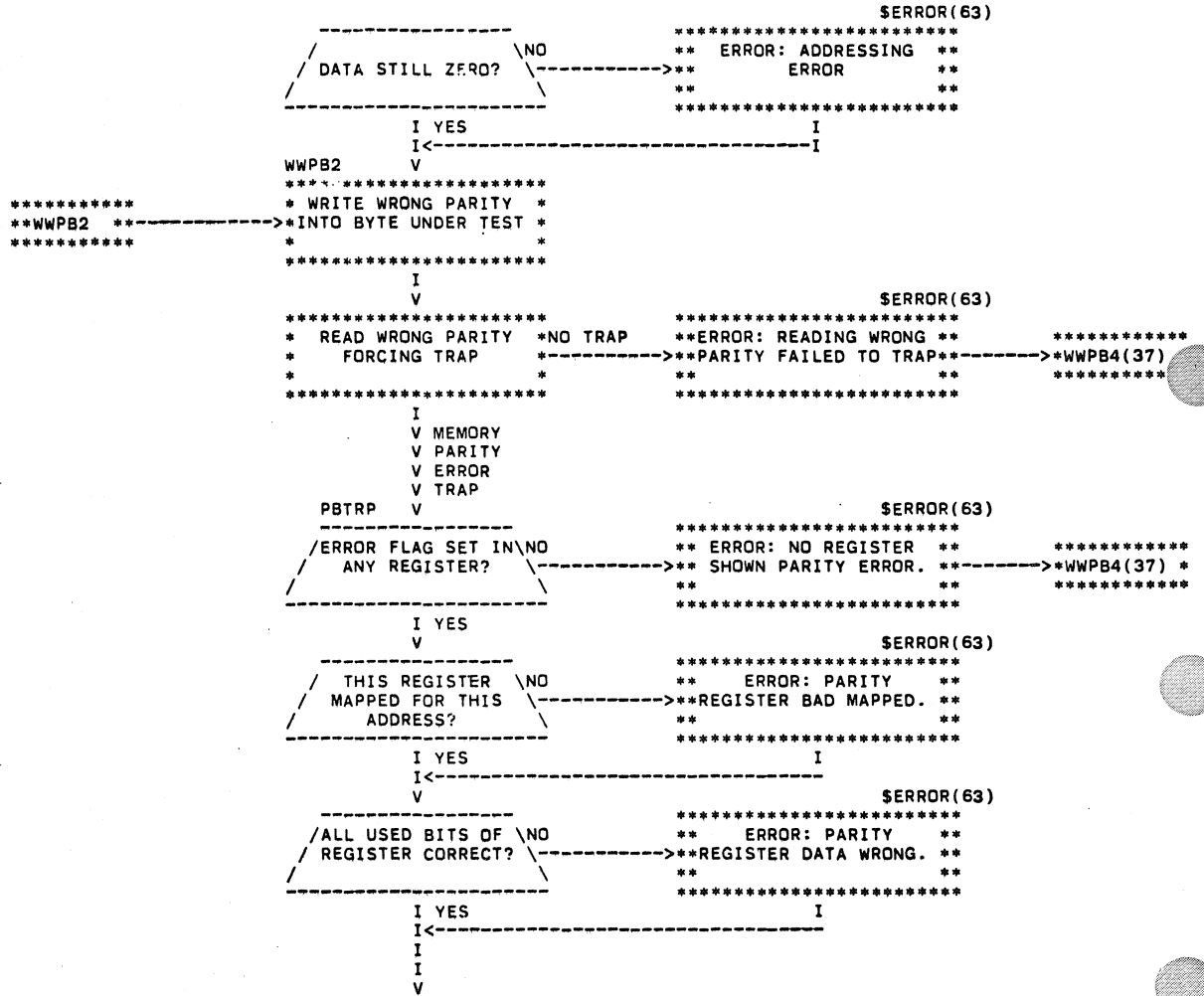


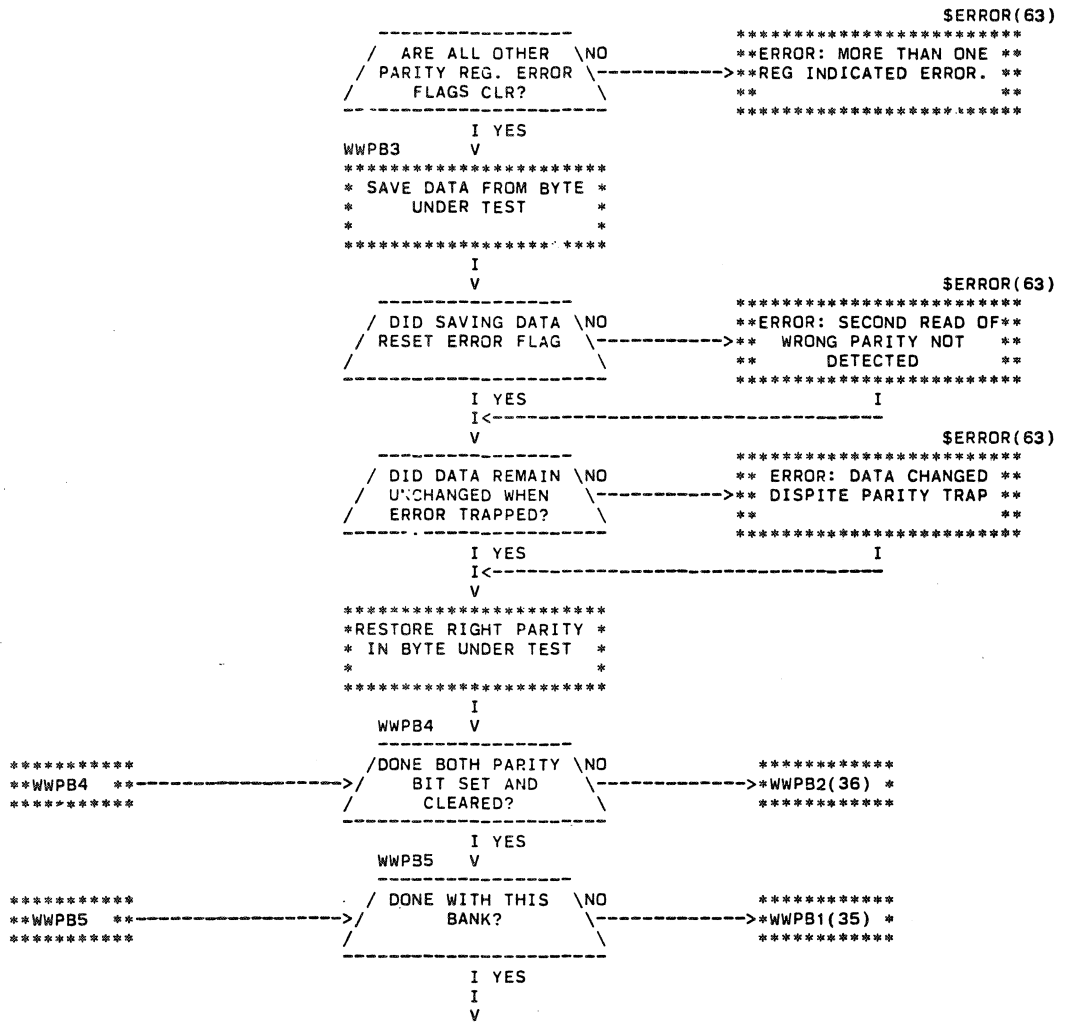
```
TST17
-----
/ANY MEMORY PARITY\NO
  REGISTERS?      ----->*TST20(39) *
-----
                      *****
                      *TST20(39) *
                      *****

      I YES
      V   SETCON(57)
*****
** FILL MEMORY WITH **
**     ZEROS         **
**                   **
*****
      I
      V   INITMM(55)
*****
** IN.TIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
      I
      V   WWPBT
-----
*****
**WWPBT **>/ DOES THIS BANK \NO
*****> HAVE PARITY? ----->*WWPB5(37) *
*****>
                      *****
                      *WWPB5(37) *
                      *****

      I YES
      V   SETAE(60)
*****
** SET MEMORY PARITY **
** ACTION ENABLE ALL **
**   REGISTERS       **
*****
      I
      V   CKPMER(61)
*****
** CHECK FOR NON-TRAP **
** MEMORY PARITY ERRORS **
**                   **
*****
      I
      V   WWPB1
-----
*****
**WWPB1 **>/ POINTING TO \YES
*****> PARITY VECTOR ----->* +4 TO ADDRESS *
-----> (114)? ----->*   POINTER   *----->*WWPB5(37) *
*****>
                      *****
                      *WWPB5(37) *
                      *****

      I NO
      I
      I
      V
```





```
MMUP(56)
*****
***** MORE MEMORY ** UPDATE ADDRESS **
*WWPBT(35) *<-----**   POINTERS   **
*****
*****
IDONE
V MAMF(60)
*****
** RESET ALL PARITY **
**   REGISTERS   **
**               **
*****
I
I
I
V
```

```
*****  
**TST20 **  
*****  
TST20      I      INITMM(55)  
*****  
** INITIALIZE ADDRESS **  
**   POINTERS          **  
**                    **  
*****  
----->I  
I      V  
I      *****  
I      * COPY 2K BLOCK OF *  
I      * PROGRAM CODE INTO *  
I      * MEMORY UNDER TEST *  
I      *****  
I      I  
I      V  
I      /DID "RANDOM" DATA\NO  
I      COPY OK?-----> ** ERROR: PROGRAM CODE **  
I      ** COPIED CHANGE.  **  
I      *****  
I      I YES I  
I      I<-----  
I      V      MMUP(56)  
I      *****  
I      MORE MEMORY ** UPDATE ADDRESS **  
I      **   POINTERS          **  
I      **                    **  
I      *****  
I      IDONE  
I      I  
I      I  
I      I  
I      V
```



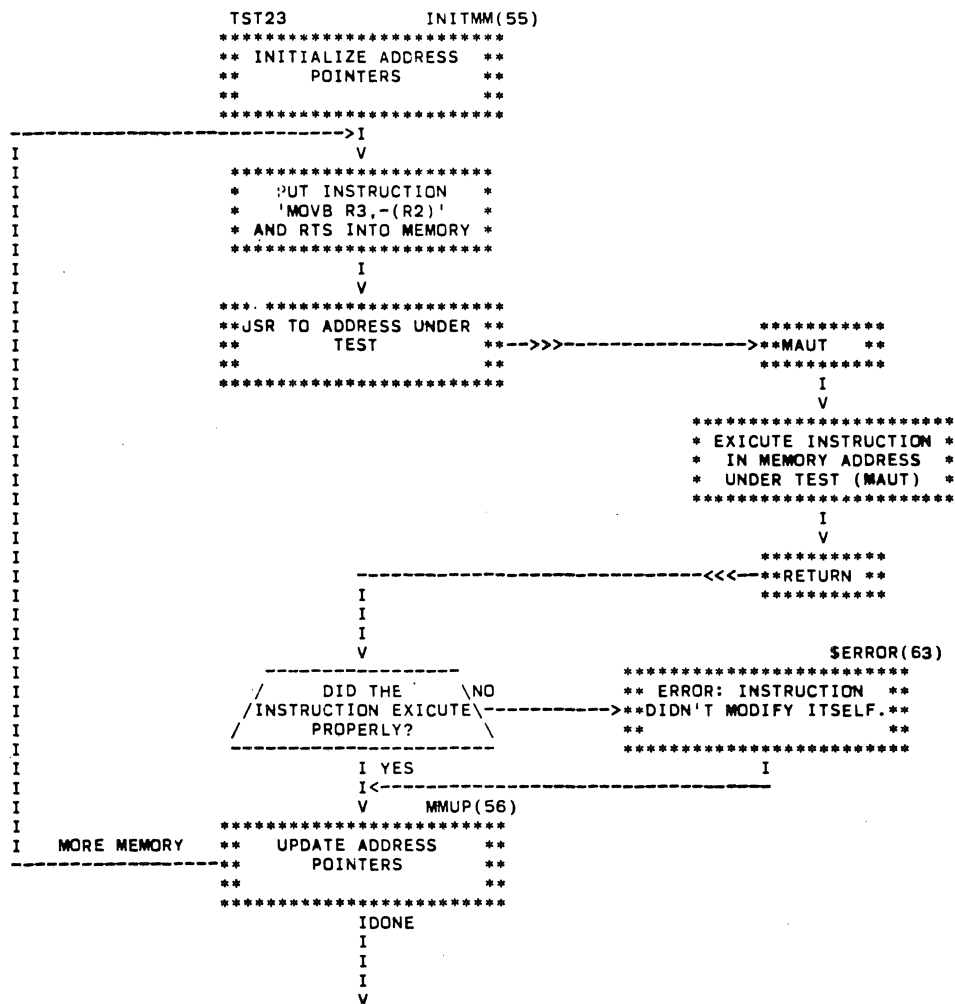
```

TST21          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
** :*****
----->I
I              V
I              *****
I              * PUT INSTRUCTION *
I              * 'MOV R4,(R2)' *
I              * AND RTS INTO MEMORY *
I              *****
I              I
I              V
I              *****
I              **JSR TO ADDRESS UNDER **
I              **   TEST           **-->>>-----> **MAUT **
I              **                   **
I              *****
I              I
I              V
I              *****
I              * EXICUTE INSTRUCTION *
I              * IN MEMORY ADDRESS *
I              * UNDER TEST (MAUT) *
I              *****
I              I
I              V
I              *****
I              <<<--**RETURN **
I              *****
I              I
I              I
I              V
I              *****
I              / DID THE \ NO
I              / INSTRUCTION EXICUTE \
I              / PROPERLY? \-----> ** ERROR: INSTRUCTION **
I              \             \-----> ** DIDN'T MODIFY ITSELF. **
I              \             \
I              I YES           I
I              I<-----I
I              V          MMUP(56)
I              *****
I              MORE MEMORY ** UPDATE ADDRFS **
I              **   POINTERS   **
I              **                   **
I              *****
I              IDONE
I              I
I              I
I              V

```

```

        TST22          INITMM(55)
        *****
        ** INITIALIZE ADDRESS **
        **     POINTERS      **
        **                   **
        *****
        ----->I
        V
        *****
        *   PUT INSTRUCTION   *
        * 'MOVB R4,(R2)'     *
        * AND RTS INTO MEMORY *
        *****
        I
        V
        *****
        **JSR TO ADDRESS UNDER TEST **
        *****
        ----->***MAUT ***
        *****
        I
        V
        *****
        * EXICUTE INSTRUCTION *
        * IN MEMORY ADDRESS *
        * UNDER TEST (MAUT) *
        *****
        I
        V
        *****
        <<<---**RETURN **
        *****
        I
        I
        I
        V
        *****
        / DID THE \ \ NO
        / INSTRUCTION EXICUTE \
        \ PROPERLY? \ ----->***DIDN'T MODIFY ITSELF.**
        *****
        I YES
        I <----- I
        V      MMUP(56)
        *****
        I MORE MEMORY ** UPDATE ADDRESS **
        *****
        **     POINTERS      **
        **                   **
        *****
        IDONE
        I
        I
        I
        V
  
```

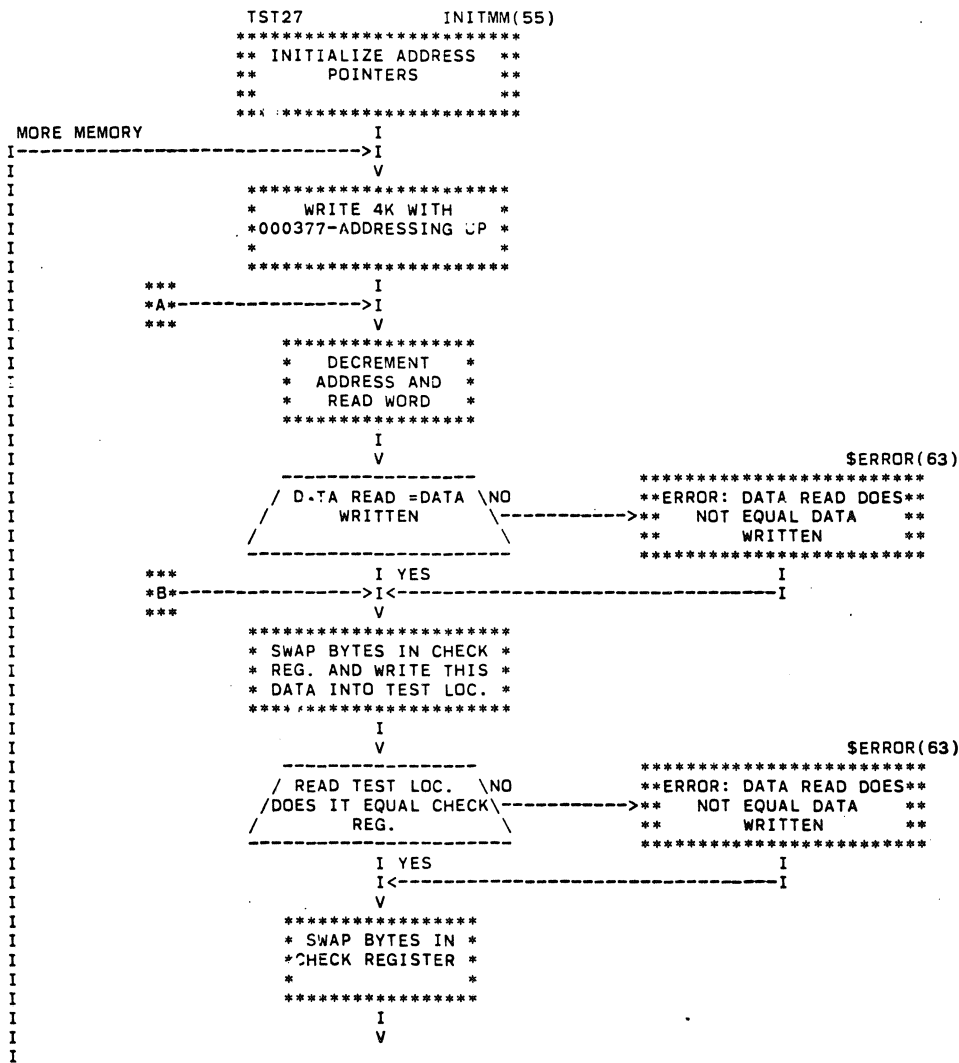


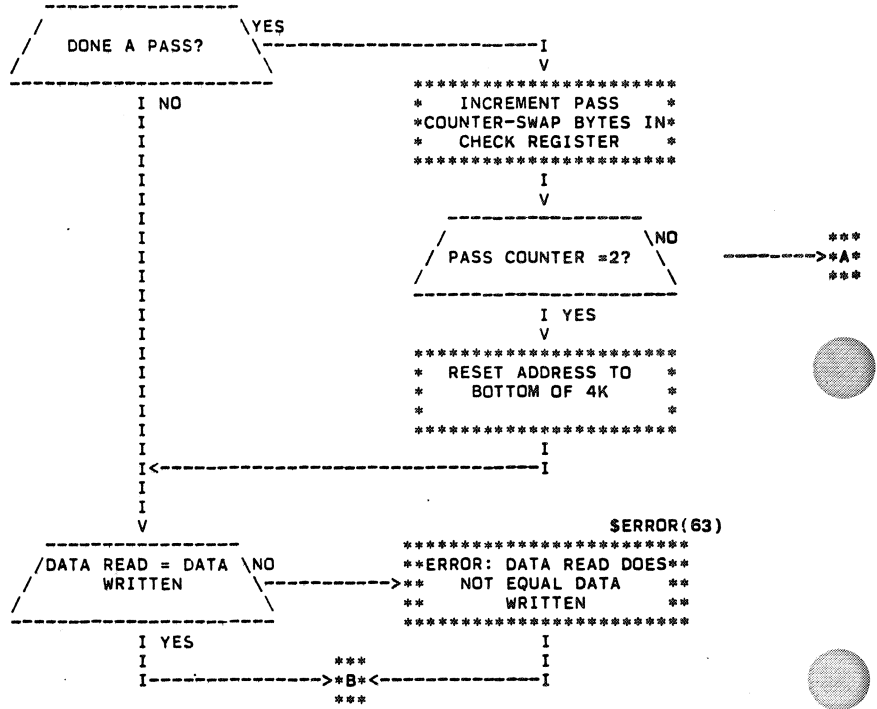
```

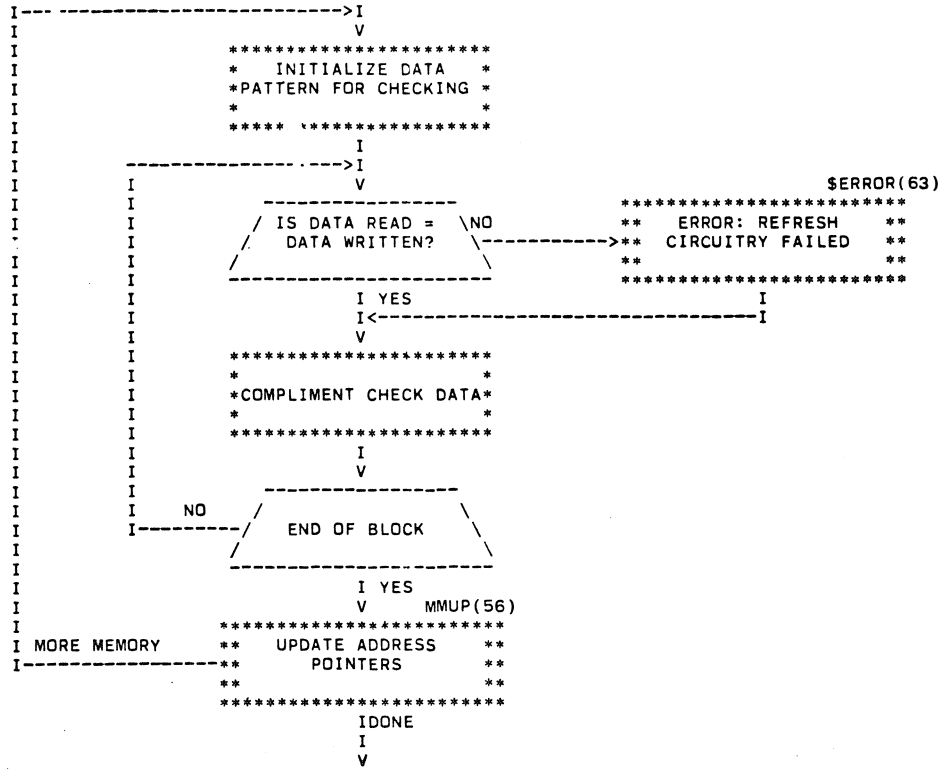
TST24          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
** *****
>I
V
*****
* PUT INSTRUCTION *
* 'NEG (R2)' *
* AND RTS INTO MEMORY *
*****
I
V
*****
**JSR TO ADDRESS UNDER **
**   TEST          **-->>>----->***MAUT ***
** *****
** *****
I
V
*****
* EXICUTE INSTRUCTION *
* IN MEMORY ADDRESS *
* UNDER TEST (MAUT) *
*****
I
V
*****
**RETURN **
** *****
I
I
I
V
*****
$ERROR(63)
/ DID THE \ NO
/ INSTRUCTION EXICUTE \
/ PROPERLY? \ ----->***ERROR: INSTRUCTION **
** *****
** *****
I
I
I
V
MMUP(56)
I MORE MEMORY ** UPDATE ADDRFS **
**   POINTERS   **
** *****
IDONE
I
I
I
V
```

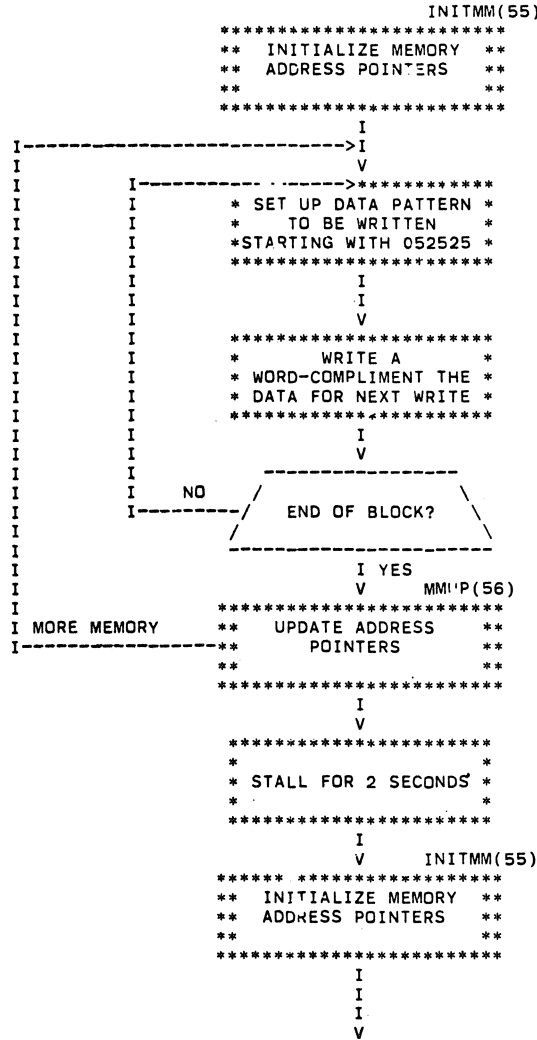
```
TST25          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS       **
**                 **
*****
----->I
I
I      V
I      *****
I      *   PUT INSTRUCTION *
I      * 'BICB (R2)+,-(R2)' *
I      * AND RTS INTO MEMORY *
I      *****
I      I
I      V
I      *****
I      **JSR TO ADDRESS UNDER **
I      **   TEST                **-->>>----->***MAUT ***
I      **                 **
I      *****
I
I
I      I
I      V
I      *****
I      **EXECUTE INSTRUCTION **
I      ** IN MEMORY ADDRESS  **
I      ** UNDER TEST (MAUT) **
I      *****
I      I
I      V
I      *****
I      **RETURN **
I      *****
I
I      I
I      I
I      V
I      *****
I      **ERROR(63) **
I      ** ERROR: INSTRUCTION **
I      **DIDN'T MODIFY ITSELF.**
I      **                 **
I      *****
I      I YES
I      I<-----I
I      V      MMUP(56)
I      *****
I      MORE MEMORY ** UPDATE ADDRESS **
I      **   POINTERS   **
I      **                 **
I      *****
I      IDONE
I      I
I      I
I      V
```

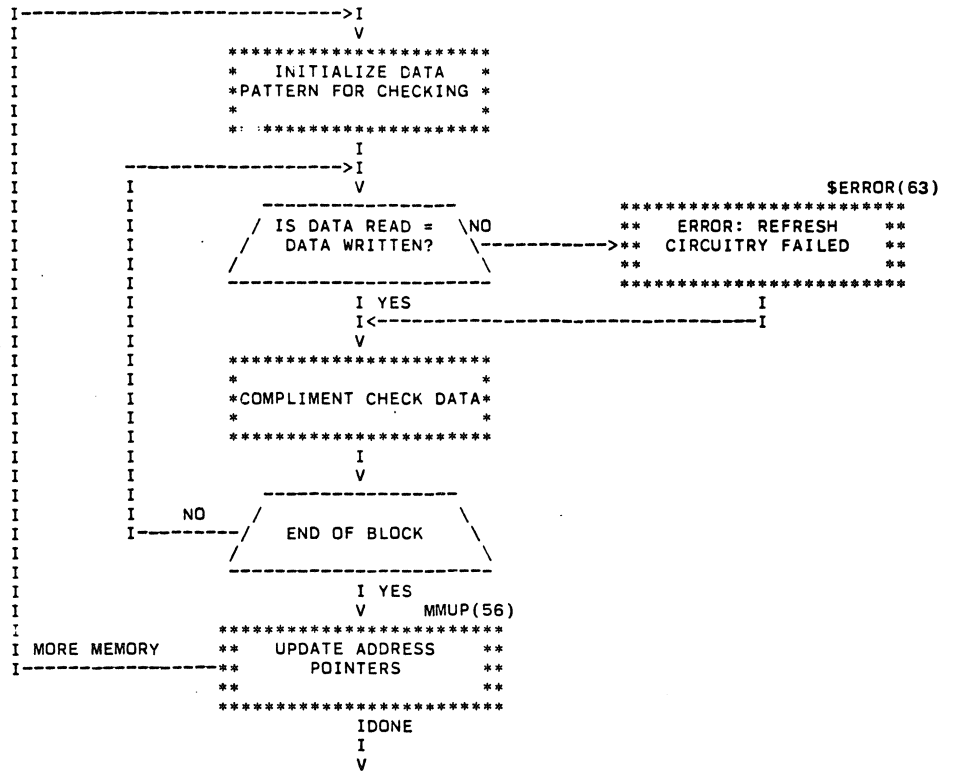
```
TST26          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
** *****
----->I
I              V
I              *
I              * PUT INSTRUCTION *
I              * 'BISB (R2)+,(R2)' *
I              * AND RTS INTO MEMORY *
I              * *****
I              I
I              V
I              *****
I              **JSR TO ADDRESS UNDER **
I              ** TEST **----->>>----->*****
I              ** *****
I              ** *****
I              ** EXICUTE INSTRUCTION **
I              ** IN MEMORY ADDRESS **
I              ** UNDER TEST (MAUT) **
I              ** *****
I              I
I              V
I              *****
I              **RETURN **
I              ** *****
I              I
I              I
I              V
I              *****
I              / DID THE \ \ NO
I              / INSTRUCTION EXICUTE \
I              / PROPERLY? \
I              *****
I              I YES
I              I<-----I
I              V          MMUP(56)
I              *****
I              ** UPDATE ADDRESS **
I              **   POINTERS     **
I              ** *****
I              ** *****
I              IDONE
I              I
I              I
I              V
```

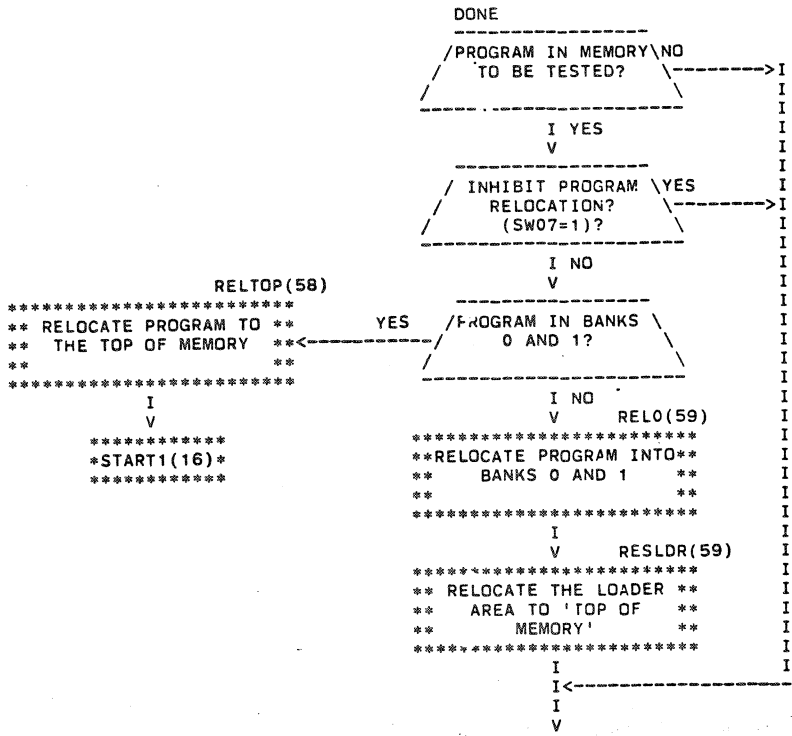






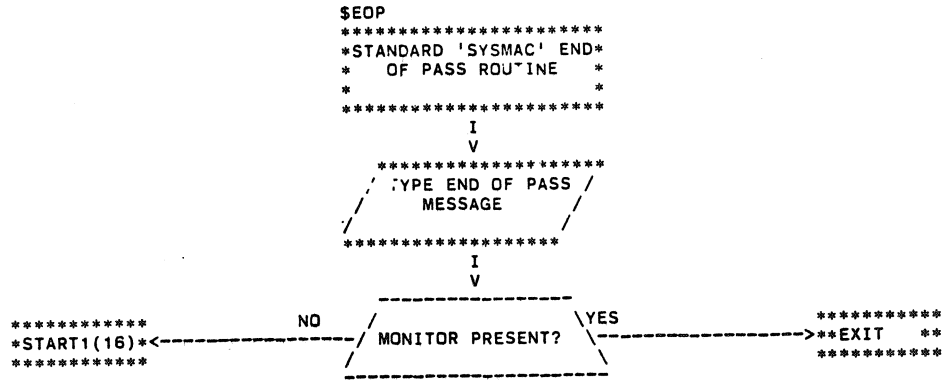






CZQMCFO 0-124K MEM EXER 16K
END OF PASS

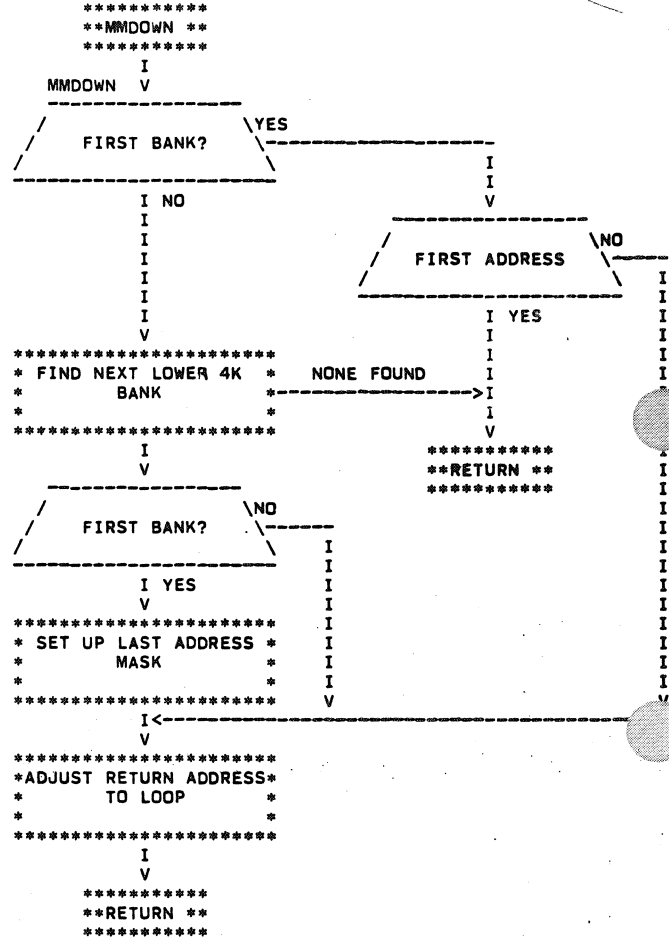
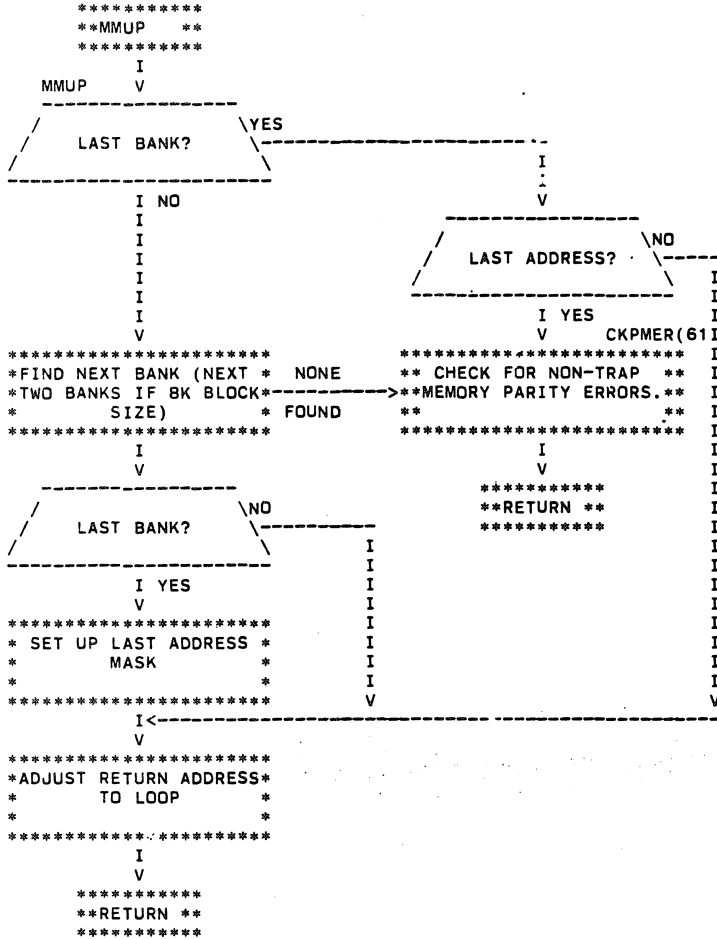
DECFLD VER 00.07 20-FEB-78 07:58



```
*****
**MMINIT **
*****
I
V
MMINIT
*****
*SET UP ALL THE MEMORY*
*MANAGEMENT REGISTERS.*
*
*****
I
V
*****
**RETURN **
*****
```

```
*****
**INITMM **
*****
I
V
INITMM
*****
*LOCATE FIRST BANK IN *
* TEST MAP *
*
*****
I
V
-----
/ LAST BANK AS \ NO
| WELL? |-----
|-----
I YES
V
*****
* SET UP LAST ADR MASK *
*
*****
I<-----
V
-----
/ BK BLOCK SIZE? \ NO
|-----
|-----
I YES
V
*****
* SET UP SECOND BANK *
* POINTERS *
*
*****
I<-----
V
-----
/ FIRST BANK FLAG \ NO
| SET? |-----
|-----
I YES
V
*****
* PUT FIRST ADDRESS *
* INTO ADDRESS POINTER *
*
*****
I
I
I
----->I
```

```
*****
**INITDN **
*****
I
V
INITDN
*****
* SET UP ADDRESS *
* POINTERS TO LAST *
* POSSIBLE ADDRESS *
*****
I
V
MMDOWN(56)
*****
**SEARCH DOWNWARDS FOR **
**TOP BANK IN TEST MAP **
**
*****
I
V
-----
/ LAST BANK FLAG \ NO
| SET? |-----
|-----
I YES
V
*****
* PUT LAST ADDRESS INTO *
* ADDRESS POINTER *
*
*****
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
----->I<-----
INITEX
V
*****
* SAVE RETURN PC FOR *
* LOOP ADDRESS *
*
*****
I
V
*****
**RETURN **
*****
```


```
*****  
**RELOC **  
*****  
I  
RELOC V  
*****  
* MOVE 8K BLOCK OF *  
* MEMORY FROM SRC TO *  
* DST *  
*****  
I  
V  
-----  
/ DATA OK AFTER \ NO  
/ MOVE? \----->  
-----  
I YES  
V  
*****  
/ TYPE PROGRAM \  
/ RELOCATION MESSAGE \  
-----  
I  
V  
*****  
**RETURN **  
*****
```

```
*****  
$ERROR(63)  
*****  
** ERROR: RELOCATION **  
** FAILURE **  
*****  
I  
V  
*****  
**HALT **  
*****
```

```
*****  
**RELTOP **  
*****  
I  
RELTOP V  
-----  
NO / MEMORY \ YES  
/ MANAGEMENT? \-----  
I / I  
I / I  
V / V  
*****  
* SET UP DESTINATION * * SET UP MEM MGMT *  
* PART OF 'RELOC' TO * * REGISTERS TO POINT TO *  
* POINT TO LAST 2 BANK * * LAST 2 BANKS *  
*****  
I I  
V V RELOC(58) RELOC(58)  
*****  
** RELOCATE PROGRAM TO ** * RELOCATE PROGRAM TO *  
** LAST 2 BANKS ** * LAST 2 BANKS *  
** ** *  
*****  
I I  
V V  
*****  
*ADJUST ALL PERTINENT * I  
* ADDRESS POINTERS * I  
* * I  
***** I  
I I  
----->I<-----  
V  
*****  
**RETURN **  
*****
```

```
*****  
**RELO **  
*****  
      I  
      RELO  V  
-----  
NO /  MEMORY MANAGEMENT? \ YES  
-----  
I /  \ I  
I /  \ I  
V /  \ V  
      RELOC(58)  
*****  
**RELOCATE PROGRAM BACK**  
**  TO BANKS 0+1  **  
**                **  
*****  
      I  
      V  
*****  
*ADJUST ALL PERTINENT *  
* ADDRESS POINTERS *  
*                *  
*****  
      I  
----->I<-----  
      I  
      V  
*****  
**RETURN **  
*****
```

```
*****  
**RESLDR **  
*****  
      I  
      RESLDR  V  
*****  
* MOVE "LOADER" FROM *  
* END OF 8K TO TOP OF *  
*   MEMORY   *  
*****  
      I  
      V  
*****  
**RETURN **  
*****
```

```

*****
**PESRV **
*****
I
PESRV V
*****
/TYPE UNEXPECTED
TRAP MESSAGE \
*****
I
V
-----
/ERROR FLAG SET IN NO
ANY PARITY REGISTER? \
-----
I YES
V
*****
** REPORT TRAP PC AND **
** REGISTER DATA **
** **
*****
I
V PSCAN(61)
*****
** SCAN ME. DRY FOR ALL **
**BAD PARITY LOCATIONS **
** **
*****
I<-----
V
*****
**RETURN **
*****

*****
**SETAE ** **MAMF **
*****
I
I MAMF V
I
I /PARITY REGISTER \ NO
I /EXIST AND NOT /
I /INHIBITED? \
I
I I YES
I
I V
I
I *****
I *SET UP PARITY VECTOR.*
I *SET 'ACTION ENABLE' *
I * IN ALL REGISTERS *
I *****
I
I<-----
V
*****
**RETURN **
*****

*****
**CLRPAR **
*****
I
CLRPAR V
*****
**CLEAR OUT ALL MEMORY **
** PARITY REGISTERS **
** **
*****
I
V
*****
**RETURN **
*****

          ERROR(63)
*****
** ERROR: TRAP BUT NO **
** FLAG **
** **
*****
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I
I

```

```
*****  
**CKPMER **  
*****  
      I  
      CKPMER V  
NO /-----/ PARITY REGISTER  
I<---/ EXIST AND NOT  
I / INHIBITED?  
I /-----/ V  
I /-----/ ANY ERROR FLAGS  
IND / SET?  
I<---/-----/ V  
I /-----/ ERROR SHOULD HAVE  
I / TRAPPED? YES  
I /-----/ I NO  
I / V $ERROR(63)  
I *****  
I ** ERROR: PARITY ERROR **  
I ** ERROR: PARITY ERROR **  
I *****  
I /-----/ I  
I / V PSCAN(61)  
I *****  
I ** SCAN MEMORY FOR ALL **  
I **BAD PARITY LOCATIONS.**  
I ** **  
I *****  
I >I<-----/ V  
I *****  
I **RETURN **  
I *****
```

\$ERROR(63)

```
*****  
**PSCAN **  
*****  
      I  
      PSCAN V  
*****  
      /-----/ I  
      / TYPE SCANNING  
      / MESSAGE  
      /-----/ V  
I >I  
I /-----/ I  
I *****  
I *  
I *READ MEMORY LOCATION *  
I *  
I *****  
I /-----/ I  
I / V  
I /-----/ ANY PARITY ERROR  
I / FLAGS? YES  
I /-----/ I NO  
I / I<-----/ I  
I / V  
I *****  
I * UPDATE ADDRESS *  
I * POINTERS *  
I *  
I *****  
I /-----/ I  
I / V IDONE  
I /-----/ I  
I / ANY PARITY ERRORS NO  
I / FOUND? YES  
I /-----/ I  
I / I YES  
I / I<-----/ I  
I / V  
I *****  
I **RETURN **  
I *****
```

\$ERROR(63)

\$ERROR(63)

```

*****
**SPRNT **--->I
*****
I
I
*****
**SPRNTQ **--->I
*****
I
I
*****
**SPRNTNR **--->I
*****
I
I
*****
**SPRNT0 **--->I
*****
I
I
*****
**SPRNT1 **--->I
*****
I
I
*****
**SPRNT3 **--->I
*****
I
I
*****
**SPRNT2 **--->I
*****
I
I
V

```

```

*****
* ROUTINES TO SET UP *
* DATA FOR EPROR *
* TYPEOUTS. *
*****

```

```

I
V
*****
**RETURN **
*****

```

```

*****
**TYPMAP **
*****
I
TYPMAP V
-----
MAP CONTAIN \NO / TYPE EMPTY MAP
  FLAGS? /-----> / MESSAGE /
-----
I YES I
V I
***** I
/TYPE FIRST + LAST I
ADDRESS OF BANKS I
FOUND I
***** I
I<-----
V
*****
**RETURN **
*****

```

```
$SCOPE
*****
* CONTROLS LOOPING, * *****
**$SCOPE ***->* INTERATIONS, ETC. ***->***RETURN **
*****
* BETWEEN SUBTESTS * *****
*****
```

```
$ERROR
*****
*COUNTS ERRORS, LOOPS.* *****
**$ERROR ***->*PASS DATA TO $ERRTYP ***->***RETURN **
*****
* * *****
*****
```

```
$ERRTYP
*****
* TYPEOUT ERROR * *****
**$ERRTYP ***->*MESSAGE, HEADER, AND ***->***RETURN **
*****
* DATA * *****
*****
```

```
$RDCHR
*****
*INPUTS CHARACTER FROM* *****
**$RDCHR ***->* TTY * ***->***RETURN **
*****
* * *****
*****
```

```
$ROLIN
*****
* INPUTS STRING OF * *****
**$ROLIN ***->* CHARACTERS FROM TTY ***->***RETURN **
*****
* * *****
*****
```

```
$RDOCT
*****
*CONVERTS ASCII OCTAL * *****
**$RDOCT ***->* NUMBER TO MACHINE ***->***RETURN **
*****
* NARY * *****
*****
```

```
$PRINT
*****
* RELOCATES MESSAGE * *****
**$PRINT ***->* ADDRESS FOR $TYPE ***->***RETURN **
*****
* * *****
*****
```

```
$TYPE
*****
* TYPES OUT A MESSAGE * *****
**$TYPE ***->* ON TTY. ***->***RETURN **
*****
* * *****
*****
```

```
$TYPDS
*****
* * *****
**$TYPDS ***->*TYPE A DECIMAL NUMBER***->***RETURN **
*****
* * *****
*****
```

```
$TYPOC
*****
* * *****
**$TYPOC ***->*TYPE AN OCTAL NUMBER ***->***RETURN **
*****
* * *****
*****
```

```
$ERRTRP
*****
* UNEXPECTED TIMEOUT * *****
**$ERRTRP ***->* TRAP (TO 4) ROUTINE ***->***HALT **
*****
* * *****
*****
```

```
$TYPAD
*****
* TYPE AN 18-BIT * *****
**$TYPAD ***->* ADDRESS (OCTAL) ***->***RETURN **
*****
* * *****
*****
```

```
*****
* * *****
* ASCII MESSAGES * *****
* * *****
*****
```

```
*****
* ERROR DATA FORMAT * *****
* TABLE * *****
* * *****
*****
```

```
*****
** .END **
*****
```


CZQMCFO 0-124K MEM EXER 16K
FLOW CHART CROSS REFERENCE LIST

DEC FLO VER 00.07 20-FEB-78 07:58 PAGE 66

	63#			
\$SCOPE	63	63#		
\$TYPAD	10	10	63	63#
\$TYPDS	63	63#		
\$TYPE	63	63#		
\$TYPOC	63	63#		
.END	63			

```

TITLE CZQMCF0 0-124K MEMORY EXERCISER, 16K VER
;COPYRIGHT (C) 1975,1978
;DIGITAL EQUIPMENT CORP.
;MAYNARD, MASS. 01754
;*
;*PROGRAM BY BRUCE BURGESS/KEN CHAPMAN
;*
;THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
;*
```

```

13          .SBTTL OPERATIONAL SWITCH SETTINGS
14          ;*
15          ;* SWITCH USE
16          ;* -----
17          ;* 15 HALT ON ERROR
18          ;* 14 LOOP ON TEST
19          ;* 11 INHIBIT ERROR TYPEDOUTS
20          ;* 12 INHIBIT KT11 (AT START TIME ONLY)
21          ;* 11 INHIBIT ITERATIONS
22          ;* 10 BELL ON ERROR
23          ;* 9 LDDP ON ERROR
24          ;* 8 LOOP ON TEST IN SWR<4:0>
25          ;* 7 INHIBIT PROGRAM RELOCATION
26          ;* 6 INHIBIT PARITY ERROR DETECTION
27          ;* 5 INHIBIT EXERCISING VECTOR AREA.
28          .SBTTL BASIC DEFINITIONS
29
30          ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
31          001100 STACK= 1100
32          .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
33          .EQUIV IO1,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
34
35          ;*MISCELLANEOUS DEFINITIONS
36          000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
37          000012 LF= 12 ;;CODE FOR LINE FEED
38          000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
39          000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
40          177776 PS= 177776 ;;PROCESSOR STATUS WORD
41          .EQUIV PS,PSW
42          177774 STKLM= 177774 ;;STACK LIMIT REGISTER
43          177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
44          177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
45          177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
46
47          ;*GENERAL PURPOSE REGISTER DEFINITIONS
48          000000 R0= %0 ;;GENERAL REGISTER
49          000001 R1= %1 ;;GENERAL REGISTER
50          000002 R2= %2 ;;GENERAL REGISTER
51          000003 R3= %3 ;;GENERAL REGISTER
52          000004 R4= %4 ;;GENERAL REGISTER
53          000005 R5= %5 ;;GENERAL REGISTER
54          000006 R6= %6 ;;GENERAL REGISTER
55          000007 R7= %7 ;;GENERAL REGISTER
56          000006 SP= %6 ;;STACK POINTER
```

```

57          000007 PC= %7 ;;PROGRAM COUNTER
58
59          ;*PRIORITY LEVEL DEFINITIONS
60          000000 PRO= 0 ;;PRIORITY LEVEL 0
61          000040 PR1= 40 ;;PRIORITY LEVEL 1
62          000100 PF2= 100 ;;PRIORITY LEVEL 2
63          000140 PR3= 140 ;;PRIORITY LEVEL 3
64          000200 PR4= 200 ;;PRIORITY LEVEL 4
65          000240 PR5= 240 ;;PRIORITY LEVEL 5
66          000300 PR6= 300 ;;PRIORITY LEVEL 6
67          000340 PR7= 340 ;;PRIORITY LEVEL 7
68
69          ;*SWITCH REGISTER SWITCH DEFINITIONS
70          100000 SW15= 100000
71          000000 SW14= 40000
72          020000 SW13= 20000
73          010000 SW12= 10000
74          004000 SW11= 4000
75          002000 SW10= 2000
76          001000 SW09= 1000
77          000400 SW08= 400
78          000200 SW07= 200
79          000100 SW06= 100
80          000040 SW05= 40
81          000020 SW04= 20
82          000010 SW03= 10
83          000004 SW02= 4
84          000002 SW01= 2
85          000001 SW00= 1
86          .EQUIV SW09,SW9
87          .EQUIV SW08,SW8
88          .EQUIV SW07,SW7
89          .EQUIV SW06,SW6
90          .EQUIV SW05,SW5
91          .EQUIV SW04,SW4
92          .EQUIV SW03,SW3
93          .EQUIV SW02,SW2
94          .EQUIV SW01,SW1
95          .EQUIV SW00,SW0
96
97          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
98          100000 BIT15= 100000
99          000000 BIT14= 40000
100         020000 BIT13= 20000
101         010000 BIT12= 10000
102         004000 BIT11= 4000
103         002000 BIT10= 2000
104         001000 BIT09= 1000
105         000400 BIT08= 400
106         000200 BIT07= 200
107         000100 BIT06= 100
108         000040 BIT05= 40
109         000020 BIT04= 20
110         000010 BIT03= 10
111         000004 BIT02= 4
112         000002 BIT01= 2
```

```

113          000001          BIT00= 1
114          .EQUIV BIT09,BIT9
115          .EQUIV BIT08,BIT8
116          .EQUIV BIT07,BIT7
117          .EQUIV BIT06,BIT6
118          .EQUIV BIT05,BIT5
119          .EQUIV BIT04,BIT4
120          .EQUIV BIT03,BIT3
121          .EQUIV BIT02,BIT2
122          .EQUIV BIT01,BIT1
123          .EQUIV BIT00,BIT0
124
125          ;;BASIC "CPU" TRAP VECTOR ADDRESSES
126          000004          ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
127          000010          RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
128          000014          TBITVEC=14        ;;"T" BIT
129          000014          TRIVEC= 14        ;;TRACE TRAP
130          000020          BPTVEC= 14        ;;BREAKPOINT TRAP (BPT)
131          000020          IOTVEC= 20        ;;INPUT,OUTPUT TRAP (IOT) **SCOPE**
132          000024          PWRVEC= 24        ;;POWER FAIL
133          000030          EMIVVEC= 30       ;;EMULATOR TRAP (EMT) **ERROR**
134          000034          TRAPVEC=34        ;;"TRAP" TRAP
135          000060          TKVEC= 60         ;;TTY KEYBOARD VECTOR
136          000064          TPVEC= 64        ;;TTY PRINTER VECTOR
137          000240          PTRQVEC=240      ;;PROGRAM INTERRUPT REQUEST VECTOR
138
139
140          .SBTTL MEMORY MANAGEMENT DEFINITIONS
141
142          ;*KT11 VECTOR ADDRESS
143
144          000250          MMVEC= 250
145
146          ;*KT11 STATUS REGISTER ADDRESSES
147
148          177572          SR0= 177_72
149          177574          SR1= 177574
150          177576          SR2= 177576
151          172516          SR3= 172516
152
153          ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
154
155          172300          KIPDR0= 172300
156          172302          KIPDR1= 172302
157          172304          KIPDR2= 172304
158          172306          KIPDR3= 172306
159          172310          KIPDR4= 172310
160          172312          KIPDR5= 172312
161          172314          KIPDR6= 172314
162          172316          KIPDR7= 172316
163
164          ;*KERNEL "I" PAGE ADDRESS REGISTERS
165
166          172340          KIPAR0= 172340
167          172342          KIPAR1= 172342
168          172344          KIPAR2= 172344
    
```

```

169          172346          KIPAR3= 172346
170          172350          KIPAR4= 172350
171          172352          KIPAR5= 172352
172          172354          KIPAR6= 172354
173          172356          KIPAR7= 172356
174
175          000000          UP = 0          ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
176          000006          RW = 6          ;CODE FOR READ/WRITE IN MEM MGMT PDR'S
177
178          ;* PARITY MEMORY DEFINITIONS.
179          000001          AE=1          ;PARITY ACTION ENABLE
180          000114          PARVEC=114      ;PARITY TRAP VECTOR
181
182          ;* MISCELLANEOUS ASSIGNMENTS
183          017777          MASK4K= 17777    ;MASK FOR 4K ADDRESS BANK BOUNDARY.
184
185          ;* CACHE REGISTER DEFINITIONS.
186          177746          IMPCHE= 177746
187
188          .SBTTL TRAP CATCHER
189
190          000000          .=0
191          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
192          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
193          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
194          000174          .=174
195          000174          000000          DISPREG= .WORD 0          ;;SOFTWARE DISPLAY REGISTER
196          000176          000000          SWREG= .WORD 0          ;;SOFTWARE SWITCH REGISTER
197          .SBTTL STARTING ADDRESS(ES)
198          000200          000137          002640          JMP @START ;;JUMP TO STARTING ADDRESS OF PROGRAM
199          000204          0 0167          002436          JMP SELECT ;STARTING ADDRESS TO ALLOW THE OPERATOR TO
200          ;SELECT VARIOUS PARAMETERS.
201          000210          000167          000064          JMP RESTART ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
202          000214          000167          000064          JMP RESTOR ;RESTORE LOADERS TO END OF MEMORY AND HALT.
203          000220          000167          003376          JMP TIMOUT ;TYPE OUT MEMORY MAP, BYTE BY BYTE.
204
205          .ERRVEC
206          000004          025114          .WORD ERRTRP
207          000006          000000          .WORD 0
208
209          .SBTTL ACT11 HOOKS
210
211          ;*****
212          ;HOOKS REQUIRED BY ACT11
213          000010          $SVPC= .SAVE PC
214          000046          .=46
215          000046          014222          $ENDAD ;(1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
216          000052          .=52
217          000052          040000          .WORD BIT14 ;(2)SET LOC.52 TO BIT14
218          000010          .=$SVPC ;RESTORE PC
    
```

```

219      000300      .,=300
220
221      ;*****
222      ;* THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
223      ;* THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
224      ;* THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
225      ;* MANAGEMENT IS DISABLED BY "CONSOLE START".
226      ;*****
226      000300 005005 RESTAR: CI ? R5 ;CLEAR FLAG TO INDICATE RESTA?T.
227      000302 000401 BR REST1 ;GO RESTORE PROGRAM BEFORE RESTARTING.
228      000304 010705 RESTDR: MOV PC, R5 ;PUT DATA INTO FLAG FOR RESTORE.
229      000306 012706 001100 REST : MOV #STACK, SP ;SET UP THE STACK POINTER.
230      000312 005767 001206 TST MEMMAP ;CHECK IF THE MEMORY HAS BEEN MAPPED.
231      000316 001002 BNE REST2 ;BR IF MEMORY MAPPED.
232      000320 000167 002330 JMP STARTA ;GO START
233      000324 005767 000256 REST2: TST MMAVA ;CHECK IF MEM MGMT AVAILABLE.
234      000330 001470 BEQ 10$ ;BR IF NO MEM MGMT.
235      000332 022737 000001 177572 BIT #BIT0, @#SRO ;CHECK IF MEM MGMT ACTIVE.
236      000340 001034 BNE 2$ ;BR IF MEM MGMT ALREADY SET UP.
237      000342 012700 172300 MOV #KIPDR0,R0 ;POINT TO FIRST MEM MGMT DDATA REG.
238      000346 012701 000010 #B., R1 ;SET UP COUNTER.
239      000352 012720 077406 1$: MOV #077406,(R0)+ ;MAP FIRST 28K 1-FOR-1.
240      000356 005301 DEC R1 ;COUNT REGISTERS.
241      000360 001374 BNE 1$ ;BR IF MORE REG.
242      000362 012700 172340 MOV #KIPAR0,R0 ;POINT TO FIRST MEM MGMT ADDRESS REG.
243      000366 005020 CLR (R0)+ ;PAR0 MAPPED INTO BANK0.
244      000370 012720 000200 MOV #200, (R0)+ ;PAR1 MAPPED INTO BANK1.
245      000374 012720 000400 MOV #400, (R0)+ ;PAR2 MAPPED INTO BANK2.
246      000400 012720 000600 MOV #600, (R0)+ ;PAR3 MAPPED INTO BANK3.
247      000404 012720 001000 MOV #1000, (R0)+ ;PAR4 MAPPED INTO BANK4.
248      000410 012720 001200 MOV #1200, (R0)+ ;PAR5 MAPPED INTO BANK5.
249      000414 012720 001400 MOV #1400, (R0)+ ;PAR6 MAPPED INTO BANK6.
250      000420 012720 007600 MOV #7600, (R0)+ ;PAR7 MAPPED INTO BANK37.
251      000424 012737 000001 177572 MOV #BIT0, @#SRO ;ENABLE MEM MGMT.
252      000432 005000 2$: CLR R0 ;INIT TEMP PAR REG.
253      000434 016701 000142 MOV PRGMAP, R1 ;GET THE PROGRAM MAP...LO 64K.
254      000440 016702 000140 MOV PRGMAP+2,R2 ;...HI 64K.
255      000444 006202 3$: ASR R2 ;SHIFT THE MAP POINTER...HI
256      000446 006001 ROR R1 ;...LO.
257      000450 103404 BCS 4$ ;BR WHEN FIRST BANK FOUND.
258      000452 062700 000200 ADD #200, R0 ;UPDATE TMP PAR TO NEXT BANK.
259      000456 100372 BPL 3$ ;BR IF MORE.
260      000460 000000 HALT ;FATAL ERROR!!! MAP EMPTY?
261      000462 010037 172340 4$: MOV R0, @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
262      000466 000137 000472 JMP @#5$ ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
263      000472 062700 000200 5$: ADD #200, R0 ;KEEP UPDATING TEMP PAR REG.
264      000476 006202 ASR R2 ;SHIFT POINTER...HI
265      000500 006001 ROR R1 ;...LO
266      000502 103373 BCC 5$ ;BR IF TOP BANK NOT YET FOUND.
267      000504 010037 172342 MOV R0, @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
268      000510 000410 BR 20$ ;BR TO RELOCATE SECTION.
269      000512 016700 000062 10$: MOV RELOCF, R0 ;GET RELOCATION FACTOR.
270      000516 062700 001100 ADD #STACK, R0 ;SET UP STACK POINTER.
271      000522 010006 MOV R0, SP ;SET STACK TO RELOCATE PROGRAM.
272      000524 062700 177432 ADD #20$-STACK,R0 ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
273      000530 000110 JNP (R0) ;GO TO "20$" (RELOCATED).
274      000532 022767 000003 000042 20$: CMP #3, PRGMAP ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.
    
```

```

275      000540 001402 BEQ 21$ ;BR IF IN BANKS 0 AND 1.
276      000542 004767 016?14 JSR PC, RELO ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
277      000546 005705 21$: TST R5 ;CHECK RESTART/RESTORE FLAG.
278      000550 001006 BNE 22$ ;BR IF RESTORE.
279      000552 005067 000412 CLR $TIMES ;CLEAN UP BEFORE STARTING.
280      000556 105067 000320 CLR $STINM
281      000562 000167 005316 JNP START1 ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
282      000564 004767 016476 22$: JSR PC, RESLDR ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
283      000572 000000 HALT ;HALT AFTER RESTORING THE LOADERS.
284      000574 000167 002054 JMP STARTA ;CONTINUE WILL RESTART THE PROGRAM.
285
286      ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
287      ;* BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
288      ;* CIRCUMSTANCES.
288      000600 000000 RELOCF: .WORD 0 ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
289      000602 000000 000000 PRGMAP: .WORD 0,0 ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
290      000606 000000 MMAVA: .WORD 0 ;MEMORY MANAGEMENT AVAILABLE FLAG.
    
```

```

291 .SBTTL POWER DOWN AND UP ROUTINES
292
293 ;*****
294 ;POWER DOWN ROUTINE
295 000610 012737 000756 000024 $PWRDN: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST UP
296 000616 012737 000340 000726 MOV #340,@#PWRVEC+2 ;;PRIO:7
297 000624 010046 MOV RC,-(SP) ;;PUSH R0 ON STACK
298 000625 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
299 000630 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
300 000632 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
301 000634 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
302 000636 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
303 000640 017746 000274 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
304 000644 010667 000112 MOV SP,$SAVR6 ;;SAVE SP
305 000650 012737 000662 000024 MOV #SPWRUP,@#PWRVEC ;;SET UP VECTOR
306 000656 000000 HALT
307 000660 000776 BR -2 ;;HANG UP
308
309 ;*****
310 ;POWER UP ROUTINE
311 000662 012737 000756 000024 $PWRUP: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
312 000670 016706 000066 MOV $SAVR6,SP ;;GET SP
313 000674 005067 000062 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
314 000700 005277 000056 1$: INC $SAVR6 ;;WAIT FOR THE INC
315 000704 001375 BNE 1$ ;;OF WORD
316 000706 012677 000226 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
317 000712 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
318 000714 012604 MCL R4 ;;POP STACK INTO R4
319 000716 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
320 000720 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
321 000722 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
322 000724 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
323 000726 012737 000610 000024 MOV #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
324 000734 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
325 000742 004567 022544 JSR R5, SP,INT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
326 000746 025641 $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
327 000750 012716 MOV (PC)+,(SP) ;;RESTART AT RESTART
328 000752 000360 $PWADR: .WORD RESTART ;;RESTART ADDRESS
329 000754 000002 RTI
330 000756 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
331 000760 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
332 000762 000000 $SAVR6: 0 ;;PUT THE SP HERE
    
```

```

333 .SBTTL COMMON TAGS
334
335 ;*****
336 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
337 ;*USED IN THE PROGRAM.
338
339 .=1100
340 001100 $CMTAG: ;;START OF COMMON TAGS
341 001100 .WORD 0
342 001102 000 $STNM: .BYTE 0 ;;CONTAINS THE TEST NUMBER
343 001103 000 $SERF: .BYTE 0 ;;CONTAINS ERROR FLAG
344 001104 000000 $SIGN: .WORD 0 ;;CONTAINS SUBTEST ITERATION COUNT
345 001106 000000 $SLPADR: .WORD 0 ;;CONTAINS SCOPE LOOP ADDRESS
346 001110 000000 $SLPERR: .WORD 0 ;;CONTAINS SCOPE RETURN FOR ERRORS
347 001112 000000 $SERTTL: .WORD 0 ;;CONTAINS TOTAL ERRORS DETECTED
348 001114 000 $ITEMB: .BYTE 0 ;;CONTAINS ITEM CONTROL BYTE
349 001115 001 $SERMAX: .BYTE 1 ;;CONTAINS MAX. ERRORS PER TEST
350 001116 000000 $SERPC: .WORD 0 ;;CONTAINS PC OF LAST ERROR INSTRUCTION
351 001120 000000 $SGADR: .WORD 0 ;;CONTAINS ADDRESS OF 'GOOD' DATA
352 001122 000000 $SBDADR: .WORD 0 ;;CONTAINS ADDRESS OF 'BAD' DATA
353 001124 000000 $SGDAT: .WORD 0 ;;CONTAINS 'GOOD' DATA
354 001126 000000 $SBDAT: .WORD 0 ;;CONTAINS 'BAD' DATA
355 001130 000000 .WORD 0 ;;RESERVED--NOT TO BE USED
356 001132 000000 .WORD 0
357 001134 000 $SAUTOB: .BYTE 0 ;;AUTOMATIC MODE INDICATOR
358 001135 000 $SINTAG: .BYTE 0 ;;INTERRUPT MODE INDICATOR
359 001136 000000 .WORD 0
360 001140 177570 $SWR: .WORD DSWR ;;ADDRESS OF SWITCH REGISTER
361 001142 177570 $DISPLAY: .WORD DDISP ;;ADDRESS OF DISPLAY REGISTER
362 001144 177560 $TKS: 17756C ;;TTY KBD STATUS
363 001146 177562 $TKB: 177562 ;;TTY KBD BUFFER
364 001150 177564 $STPS: 177564 ;;TTY PRINTER STATUS REG. ADDRESS
365 001152 177566 $STPB: 177566 ;;TTY PRINTER BUFFER REG. ADDRESS
366 001154 000 $NULL: .BYTE 0 ;;CONTAINS NULL CHARACTER FOR FILLS
367 001155 002 $FILLS: .BYTE 2 ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
368 001156 012 $FILLC: .BYTE 12 ;;INSERT FILL CHARS. AFTER A "LINE FEED"
369 001157 000 $STPFLG: .BYTE 0 ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
370 001160 000000 $TMP0: .WORD 0 ;;USER DEFINED
371 001162 000000 $TMP1: .WORD 0 ;;USER DEFINED
372 001164 000000 $TMP2: .WORD 0 ;;USER DEFINED
373 001166 000000 $TMP3: .WORD 0 ;;USER DEFINED
374 001170 000000 $STIMES: 0 ;;MAX. NUMBER OF ITERATIONS
375 001172 000000 $SESCAPE: 0 ;;ESCAPE ON ERROR ADDRESS
376 001174 177607 000377 $SBELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
377 001200 000 $SQUES: .ASCIZ /? ;;QUESTION MARK
378 001201 015 $SCLRF: .ASCIZ <15> ;;CARRIAGE RETURN
379 001202 000012 $SLF: .ASCIZ <12> ;;LINE FEED
380
381 .SBTTL APT MAILBOX-ETABLE
382
383 ;*****
384 .EVEN
385 $MAIL: ;;APT MAILBOX
386 001204 000000 $MSGTY: .WORD MSGTY ;;MESSAGE TYPE CODE
387 001206 000000 $FATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
388 001210 000000 $TESTN: .WORD ATESTN ;;TEST NUMBER
    
```

```
389 001212 000000 SFASS: .WORD APASS ;;PASS COUNT
390 001214 000000 SDEVCT: .WORD ADEVCT ;;DEVICE COUNT
391 001216 000000 SUNIT: .WORD AUNIT ;;I/O UNIT NUMBER
392 001220 000000 SMSGAD: .WORD AMSGAD ;;MESSAGE ADDRESS
393 001222 000000 SMSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
394 001224 SETABLE: .WORD ASETAB ;;APT ENVIRONMENT TABLE
395 001224 000 SENV: .BYTE AENV ;;ENVIRONMENT BYTE
396 001225 000 SENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
397 001226 000000 SSAREG: .WORD ASWREG ;;APT SWITCH REGISTER
398 001230 000000 SUSWR: .WORD AUSWR ;;USER SWITCHES
399 001232 000000 SCPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
400 ;*
401 ;* BITS 15-11=CPU TYPE
402 ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
403 ;* 11/70=06,PD0=07,Q=10
404 ;*
405 ;* BIT 10=REAL TIME CLOCK
406 ;* BIT 9=FLOATING POINT PROCESSOR
407 ;* BIT 8=MEMORY MANAGEMENT
408 ;*
409 SMAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS, M.S. BYTE
410 SMTYP1: .BYTE AMTYP1 ;;MEM. TYPE, BLK#1
411 ;* MEM. TYPE BYTE -- (HIGH BYTE)
412 ;* 900 NSEC CORE=001
413 ;* 300 NSEC BIPOLAR=002
414 ;* 500 NSEC MDS=003
415 001236 000000 SMADR1: .WORD AMADR1 ;;HIGH ADDRESS, BLK#1
416 ;* MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
417 SMAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS, M.S. BYTE
418 SMTYP2: .BYTE AMTYP2 ;;MEM. TYPE, BLK#2
419 SMADR2: .WORD AMADR2 ;;MEM. LAST ADDRESS, BLK#2
420 SMAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS, M.S. BYTE
421 SMTYP3: .BYTE AMTYP3 ;;MEM. TYPE, BLK#3
422 SMADR3: .WORD AMADR3 ;;MEM. LAST ADDRESS, BLK#3
423 SMAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS, M.S. BYTE
424 SMTYP4: .BYTE AMTYP4 ;;MEM. TYPE, BLK#4
425 SMADR4: .WORD AMADR4 ;;MEM. LAST ADDRESS, BLK#4
426 SVECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1, BUS PRIORITY#1
427 SVECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2, BUS PRIORITY#2
428 SBASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
429 SDEVM: .WORD ADEVMM ;;DEVICE MAP
430 SCCW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
431 SCCW2: .WORD ACDW2 ;;CONTROLLER DESCRIPTION WORD#2
432 SDDW0: .WORD ADDW0 ;;DEVICE DESCRIPTOR WORD#0
433 SDDW1: .WORD ADDW1 ;;DEVICE DESCRIPTOR WORD#1
434 SDDW2: .WORD ADDW2 ;;DEVICE DESCRIPTOR WORD#2
435 SDDW3: .WORD ADDW3 ;;DEVICE DESCRIPTOR WORD#3
436 SDDW4: .WORD ADDW4 ;;DEVICE DESCRIPTOR WORD#4
437 SDDW5: .WORD ADDW5 ;;DEVICE DESCRIPTOR WORD#5
438 SDDW6: .WORD ADDW6 ;;DEVICE DESCRIPTOR WORD#6
439 SDDW7: .WORD ADDW7 ;;DEVICE DESCRIPTOR WORD#7
440 SDDW8: .WORD ADDW8 ;;DEVICE DESCRIPTOR WORD#8
441 SDDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9
442 SDDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10
443 SDDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11
444 SDDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
445 SDDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
446 SDDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
447 SDDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
```

```
445 001330 SETEND:
446 .MEXIT
447 .SBTTL APT PARAMETER BLOCK
448
449 ;*****
450 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
451 ;*****
452 .SX= . ;SAVE CURRENT LOCATION
453 .=24 . ;SET POWER FAIL TO POINT TO START OF PROGRAM
454 000024 200 . ;FOR APT START UP
455 .=44 . ;POINT TO APT INDIRECT ADDRESS PNTR.
456 000044 $APT:HDR . ;POINT TO APT HEADER BLOCK
457 .=SX . ;RESET LOCATION COUNTER
458 ;*****
459 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
460 ;INTERFACE SPEC.
461
462 001330 $APT:O:
463 001330 SHI815: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
464 001330 SMADR: .WORD SMAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
465 001330 STIM: .WORD 2400 . ;RUN TIME OF LONGEST TEST
466 001330 SPASTM: .WORD 120 . ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
467 001340 SUNITM: .WORD 240 . ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
468 001342 .WORD SETEND-SMAIL/2 . ;LENGTH MAILBOX-ETABLE(WORDS)
469 .SBTTL APT STATISTICS TABLE
470
471 ;*****
472 001344 SASTAT:
473 001344 .WORD -1,0
474 001350 .WORD -1,0
475 001354 .WORD -1,0
476 001360 .WORD -1,0
477 001364 .WORD -1,0
478 001370 .WORD -1,0
479 001374 .WORD -1,0
480 001400 .WORD -1,0
481 001404 .WORD -1,0
482 001410 .WORD -1,0
483 001414 .WORD -1,0
484 001420 .WORD -1,0
485 001424 .WORD -1,0
486 001430 .WORD -1,0
487 001434 .WORD -1,0
488 001440 .WORD -1,0
489 001444 .WORD -1,0
490 001450 .WORD -1,0
491 001454 .WORD -1,0
492 001460 .WORD -1,0
493 001464 .WORD -1,0
494 001470 .WORD -1,0
495 001474 .WORD -1,0
496 001500 .WORD -1,0
497 001504 .WORD -1,0
498 001510 .WORD -1
499 001512 001344 SASTEND:
500 SAPTR: SASTAT
```



```

501 ;*-----
502 ;*THE FOLLOWING TAGS ARE USER DEFINED
503 ;*-----
504 001514 000000 $VERPC: .WORD 0 ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
505 001516 070032 RESRVD: .WORD 070032 ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
506 ;*-----
507 001520 000000 LhAD: .WORD 0 ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
508 001522 000000 LDDISP: .WORD 0 ;CONTAINS DISPLAY REGISTER IMAGE
509 001524 MEMMAP: .WORD 0 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
510 001526 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
511 001530 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
512 001532 000000 TSTMAP: .WORD 0 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
513 001534 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
514 001536 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
515 001538 000000 SAVTST: .WORD 0 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
516 ;*-----
517 001540 000000 .WORD 0 ;TEST EACH BANK ONCE.
518 001542 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
519 001544 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
520 001546 000000 PMEMAP: .WORD 0 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
521 001548 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
522 001550 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
523 001552 000000 BITPT: .WORD 0 ;POINTER TO CURRENT 4K BANK OF MEMORY
524 001554 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
525 001556 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
526 001558 000000 TMPPT: .WORD 0 ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
527 001560 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
528 001562 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
529 001564 000000 MMORE: .WORD 0 ;LCOP ADDRESS FOR MULTIPLE BLOCK TESTING.
530 ;*-----
531 001566 000000 SELF:G: .BYTE 0 ;OPERATOR SELECTED PARAMETERS FLAG. (SA=204)
532 001568 000000 FLAG8K: .BYTE 0 ;BK BLOCK INDICATOR. USED IN "INITMM" AND "MMUP".
533 001570 000000 OEFLG: .BYTE 0 ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
534 001572 001562 .EVEN
535 001574 000000 .EVEN
536 FSTADR: .WORD 0 ;FIRST VIRTUAL ADDRESS TO BE TESTED.
537 001576 000000 .WORD 0 ;FIRST ADDRESS IS USER SELECTABLE.
538 001578 000000 TMPFAD: .WORD 0 ;ADJUSTED FIRST ADDRESS.
539 001580 000000 FADMSK: .WORD 0 ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
540 001582 000000 000000 FADMAP: .WORD 0,0 ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
541 001584 000000 LSTADR: .WORD 0 ;TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
542 ;*-----
543 001586 000000 .WORD 0 ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
544 001588 000000 .WORD 0 ;LAST ADDRESS IS USER SELECTABLE.
545 001590 000000 .WORD 0 ;ADJUSTED LAST ADDRESS.
546 001602 000000 000000 LADMSK: .WORD 0 ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
547 001604 000000 .WORD 0 ;TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
548 001606 000000 LADMAP: .WORD 0,0 ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
549 001608 000000 .WORD 0 ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
550 001610 000000 .CONST: .WORD 0 ;USER SELECTABLE CONSTANT DATA.
551 001612 000004 WWP: .WORD 4 ;WRITE WRONG PARITY COMMAND
552 001614 000000 TEMP: .WORD 0 ;TEMPORARY STORAGE
553 001616 000000 CASFLG: .WORD 0 ;CACHE PRESENT FLAG
554 001618 177746 CASREG: .WORD 177746 ;CACHE CONTROL REGISTER
555 ;*-----
556 ;* RELATIVE ADDRESSING TABLE.
557 ;* THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
    
```

```

557 ;* RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUMENT TAGS.
558 ;*-----
559 001622 RADTAB:
560 001624 001100 .STACK: STACK ;STACK POINTER INITIAL ADDRESS.
561 001626 001516 .RESRV: RESRVD ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
562 001628 002076 .MPRO: MPRO ;MEMORY PARITY REGISTER TABLE ADDRESS.
563 001630 002276 .MPRX: MPRX ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
564 001632 012052 .PBRP: PBRP ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
565 001634 002050 .MPPAT: MPPATS ;MEMORY PARITY PATTERN TABLE ADDRESS.
566 001636 017430 .PESRV: PESRV ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
567 001638 002340 .ERRTB: ERRTB ;ERROR TYPEOUT TABLE POINTER.
568 001640 000010 .EIGHT: 8 ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
569 001642 014004 .TST32: TST32 ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
570 ;*-----
571 ;* DATA CONTAINERS FOR ERROR PRINTOUT.
572 ;*-----
573 001646 001116 001120 001124 DT1: $ERRPC,$GDADR,$GDDAT,$BDDAT,0
574 001648 001126 000000
575 001650 001514 001116 001120 DT2: $VERPC,$ERRPC,$GDADR,$GLJAT,$BDDAT,0
576 001652 001124 001126 000000
577 001654 001514 001116 001120 DT12: $VERPC,$ERRPC,$GDADR,$GDDAT,0
578 001656 001124 000000
579 001658 001514 001116 001160 DT14: $VERPC,$ERRPC,$TMP0,$GDADR,0
580 001660 001120 000000
581 001662 001514 001116 001120 DT15: $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
582 001664 001160 001124 001126
583 001666 000000
584 001668 001514 001116 001160 DT21: $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
585 001670 001124 001124 001126
586 001672 000000
587 001674 001514 001116 001120 DT23: $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
588 001676 001122 001124 001126
589 001678 000000
590 001680 001514 001116 001122 DT24: $VERPC,$ERRPC,$BDADR,0
591 002000 000000
592 002002 001514 001116 001122 DT25: $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
593 002004 001160 001162 000000
594 002006 001514 001116 001160 DT26: $VERPC,$ERRPC,$TMP0,$TMP1,0
595 002008 001162 000000
596 002010 001160 001162 001120 DT30: $TMP0,$TMP1,$GDADR,$BDDAT,0
597 002012 001126 000000
598 002014 001166 000000
599 002016 177777 DT31: $TMP3,0
600 .WORD -1 ;TABLE TERMINATOR.
601 ;*-----
602 ;*SBTTL MEMORY PARITY PATTERNS TABLE
603 ;*-----
604 ;*THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
605 ;*-----
606 002050 125325 MPPATS: 125325 ;EVEN,ODD
607 002052 152652 152652 ;ODD,EVEN
608 002054 052452 052452 ;EVEN,ODD
609 002056 025125 025125 ;ODD,EVEN
610 002058 102070 102070 ;EVEN,EVEN
611 002062 072527 072527 ;ODD,ODD
612 002064 177777 177777 ;EVEN,EVEN
    
```

```

613 002065 107030          107030          ;ODD,ODD
614 002070 152525          152525          ;ODD,EVEN
615 002072 000000          0              ;EXTRA PATTERN HOLDER FOR
616                                     ;FUTURE USE
617 002074 000000          MPEND: 0        ;TABLE TERMINATOR
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632 002076 172101          MPR0: 172100 +1 ;PARITY STATUS REGISTER
633 002100 000000          0              ;CONTROL MAP (LOW 64K)
634 002102 000000          0              ;CONTROL MAP (HIGH 64K)
635 002104 000000          0              ;MASK FOR MOS,CORE,MS11-K
636 002106 172103          MPR1: 172102 +1 ;PARITY STATUS REGISTER
637 002110 000000          0              ;CONTROL MAP (LOW 64K)
638 002112 000000          0              ;CONTROL MAP (HIGH 64K)
639 002114 000000          0              ;MASK FOR MOS,CORE,MS11-K
640 002116 172105          MPR2: 172104 +1 ;PARITY STATUS REGISTER
641 002120 000000          0              ;CONTROL MAP (LOW 64K)
642 002122 000000          0              ;CONTROL MAP (HIGH 64K)
643 002124 000000          0              ;MASK FOR MOS,CORE,MS11-K
644 002126 172107          MPR3: 172106 +1 ;PARITY STATUS REGISTER
645 002130 000000          0              ;CONTROL MAP (LOW 64K)
646 002132 000000          0              ;CONTROL MAP (HIGH 64K)
647 002134 000000          0              ;MASK FOR MOS,CORE,MS11-K
648 002136 172111          MPR4: 172110 +1 ;PARITY STATUS REGISTER
649 002140 000000          0              ;CONTROL MAP (LOW 64K)
650 002142 000000          0              ;CONTROL MAP (HIGH 64K)
651 002144 000000          0              ;MASK FOR MOS,CORE,MS11-K
652 002146 172113          MPR5: 172112 +1 ;PARITY STATUS REGISTER
653 002150 000000          0              ;CONTROL MAP (LOW 64K)
654 002152 000000          0              ;CONTROL MAP (HIGH 64K)
655 002154 000000          0              ;MASK FOR MOS,CORE,MS11-K
656 002156 172115          MPR6: 172114 +1 ;PARITY STATUS REGISTER
657 002160 000000          0              ;CONTROL MAP (LOW 64K)
658 002162 000000          0              ;CONTROL MAP (HIGH 64K)
659 002164 000000          0              ;MASK FOR MOS,CORE,MS11-K
660 002166 172117          MPR7: 172116 +1 ;PARITY STATUS REGISTER
661 002170 000000          0              ;CONTROL MAP (LOW 64K)
662 002172 000000          0              ;CONTROL MAP (HIGH 64K)
663 002174 000000          0              ;MASK FOR MOS,CORE,MS11-K
664 002176 172121          MPR8: 172120 +1 ;PARITY STATUS REGISTER
665 002200 000000          0              ;CONTROL MAP (LOW 64K)
666 002202 000000          0              ;CONTROL MAP (HIGH 64K)
667 002204 000000          0              ;MASK FOR MOS,CORE,MS11-K
668 002206 172123          MPR9: 172122 +1 ;PARITY STATUS REGISTER
    
```

```

669 002210 000000          0              ;CONTROL MAP (LOW 64K)
670 002212 000000          0              ;CONTROL MAP (HIGH 64K)
671 002214 000000          0              ;MASK FOR MOS,CORE,MS11-K
672 002216 172115          MPR10: 172124 +1 ;PARITY STATUS REGISTER
673 002220 000000          0              ;CONTROL MAP (LOW 64K)
674 002222 000000          0              ;CONTROL MAP (HIGH 64K)
675 002224 000000          0              ;MASK FOR MOS,CORE,MS11-K
676 002226 172127          MPR11: 172126 +1 ;PARITY STATUS REGISTER
677 002230 000000          0              ;CONTROL MAP (LOW 64K)
678 002232 000000          0              ;CONTROL MAP (HIGH 64K)
679 002234 000000          0              ;MASK FOR MOS,CORE,MS11-K
680 002236 172131          MPR12: 172130 +1 ;PARITY STATUS REGISTER
681 002240 000000          0              ;CONTROL MAP (LOW 64K)
682 002242 000000          0              ;CONTROL MAP (HIGH 64K)
683 002244 000000          0              ;MASK FOR MOS,CORE,MS11-K
684 002246 172133          MPR13: 172132 +1 ;PARITY STATUS REGISTER
685 002250 000000          0              ;CONTROL MAP (LOW 64K)
686 002252 000000          0              ;CONTROL MAP (HIGH 64K)
687 002254 000000          0              ;MASK FOR MOS,CORE,MS11-K
688 002256 172135          MPR14: 172134 +1 ;PARITY STATUS REGISTER
689 002260 000000          0              ;CONTROL MAP (LOW 64K)
690 002262 000000          0              ;CONTROL MAP (HIGH 64K)
691 002264 000000          0              ;MASK FOR MOS,CORE,MS11-K
692 002266 172137          MPR15: 172136 +1 ;PARITY STATUS REGISTER
693 002270 000000          0              ;CONTROL MAP (LOW 64K)
694 002272 000000          0              ;CONTROL MAP (HIGH 64K)
695 002274 000000          0              ;MASK FOR MOS,CORE,MS11-K
696
697 002276 000021          MPRX: .BLKW 17, ;THIS IS THE END OF THE TABLE !
698                                     ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
699                                     ; (THE EXTRA WORD IS FOR A TERMINATOR.)
    
```

```

700 .SBTTL ERROR POINTER TABLE
701
702 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
703 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
704 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
705 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
706 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
707
708 ;* EM ;:POINTS TO THE ERROR MESSAGE
709 ;* DH ;:POINTS TO THE DATA HEADER
710 ;* DT ;:POINTS TO THE DATA
711 ;* DF ;:POINTS TO THE DATA FORMAT
712
713
714 002340 $ERRTB:
715 ;* ITEM 1
716 002340 027010 DM1 ;:PARITY REGISTER DATA ERROR.
717 002342 030367 DH1 ;:PC,REG,S/B,WAS
718 002344 001646 DT1 ;:SERRPC,$GDADR,$GDDAT,$BDDAT
719 002346 030734 DF1 ;:16,18,16,16
720
721 ;* ITEM 2
722 002350 027044 DM2 ;:ADDRESS TEST ERROR(TST1-5).
723 002352 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
724 002354 0C1660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
725 002356 030740 DF2 ;:16,18,18,16,16
726
727 ;* ITEM 3
728 002360 027044 DM2 ;:ADDRESS TEST ERROR(TST1-5).
729 002362 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
730 002364 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
731 002366 030745 DF3 ;:16,18,18,8,8
732
733 ;* ITEM 4
734 002370 027100 DM4 ;:CONSTANT DATA ERROR(TST6-10).
735 002372 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
736 002374 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
737 002376 030740 DF2 ;:16,18,18,16,16
738
739 ;* ITEM 5
740 002400 027126 DM5 ;:ROTATING BIT ERROR(TST11-12).
741 002402 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
742 002404 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
743 002406 030740 DF2 ;:16,18,18,16,16
744
745 ;* ITEM 6
746 002410 027174 DM6 ;:MDS REFRESH TEST ERROR (TST30-31).
747 002412 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
748 002414 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
749 002416 030740 DF2 ;:16,18,18,16,16
750
751 ;* ITEM 7
752 002420 027240 DM7 ;:3 XOR 9 PATTERN ERROR(TST13-16).
753 002422 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
754 002424 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
755 002426 030740 DF2 ;:16,18,18,16,16
756
757 ;* ITEM 10
758 002430 027301 DM10 ;:MARCHING 1'S AND 0'S ERROR(TST27).
759 002432 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
760 002434 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
761 002436 030740 DF2 ;:16,18,18,16,16
762
763 ;* ITEM 11
    
```

```

756 002440 027345 DM11 ;:PARITY MEMORY ADDRESS ERROR(TST17).
757 002442 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
758 002444 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
759 002446 030745 DF3 ;:16,18,18,8,8
760
761 ;* ITEM 12
762 002450 027411 DM12 ;:DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).
763 002452 030433 DH12 ;:V/PC,P/PC,MA,S/B,WAS
764 002454 001674 DT12 ;:SVERPC,$ERRPC,$GDADR,$GDDAT
765 002456 030745 DF3 ;:16,18,18,8
766
767 ;* ITEM 13
768 002460 027465 DM13 ;:WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
769 002462 030433 DH13 ;:V/PC,P/PC,MA,S/B,WAS
770 002464 001674 DT12 ;:SVERPC,$ERRPC,$GDADR,$GDDAT
771 002466 030745 DF3 ;:16,18,18,8
772
773 ;* ITEM 14
774 002470 027555 DM14 ;:PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).
775 002472 030454 DH14 ;:V/PC,P/PC,REG,MA
776 002474 001765 DT14 ;:SVERPC,$ERRPC,$TMP0,$GDADR
777 002476 030752 DF14 ;:16,18,18,18
778
779 ;* ITEM 15
780 002500 027010 DM1 ;:PARITY REGISTER DATA ERROR.
781 002502 030475 DH15 ;:V/PC,P/PC,MAUT,REG,S/B,WAS
782 002504 001720 DT15 ;:SVERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT
783 002506 030745 DF14 ;:16,18,18,18,16,16
784
785 ;* ITEM 16
786 002510 027654 DM16 ;:MORE THAN ONE REGISTER INDICATED PARITY ERROR.
787 002512 030454 DH14 ;:V/PC,P/PC,REG,MA
788 002514 001765 DT14 ;:SVERPC,$ERRPC,$TMP0,$GDADR
789 002516 030752 DF14 ;:16,18,18,18
790
791 ;* ITEM 17
792 002520 077733 DM17 ;:DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
793 ;: TRAPPED(TST21).
794 002522 030406 DH2 ;:V/PC,P/PC,MA,S/B,WAS
795 002524 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
796 002526 030745 DF3 ;:16,18,18,8,8
797
798 ;* ITEM 20
799 002530 030031 DM20 ;:RANDOM DATA ERROR(TST20).
800 002532 030454 DH2 ;:V/PC,P/PC,REG,MA,S/B,WAS
801 002534 001660 DT2 ;:SVERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
802 002536 030740 DF2 ;:16,18,18,16,16
803
804 ;* ITEM 21
805 002540 030063 DM21 ;:INSTRUCTION EXECUTION ERROR(TST21-26).
806 002542 030530 DH21 ;:V/PC,P/PC,IUT,MA,S/B,WAS
807 002544 001736 DT21 ;:SVERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
808 002546 030760 DF21 ;:16,18,16,18,16,16
809
810 ;* ITEM 22
811 ;:NOT USED
812
813 ;* ITEM 23
814 002550 030132 DM23 ;:PROGRAM CODE CHANGED WHEN RELOCATED.
815 002552 030561 DH23 ;:V/PC,P/PC,SRC MA,DST MA,S/B,WAS
816 002554 001754 DT23 ;:SVERPC,$ERRPC,$GDADR,$BDDADR,$GDDAT,$BDDAT
817 002556 030752 DF14 ;:16,18,18,18,16,16
818
819 ;* ITEM 24
820 002560 030177 DM24 ;:TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
821 002562 030621 DH24 ;:V/PC,P/PC,TRP/PC
822 002564 001772 DT24 ;:SVERPC,$ERRPC,$BDDADR
823 002566 030752 DF14 ;:16,18,18
    
```

```

812
813          002570 030253      ;* ITEM 25
814          002572 030542          DM25          ;TRAPPED TO 114.
815          002574 002072          DH25          ;V/PC,P/PC,TRP/PC,REG,WAS
816          002576 030752          DT25          ;SVRIPC,$ERRPC,$BDADR,$TMP0,$TMP1
      ;16,18,18,18,18
817          ;* ITEM 26
818          002600 030273          DM26          ;FAILED TO TRAP.
819          002602 030673          DF76          ;V/PC,P/PC,REG,WAS
820          002604 002016          DT26          ;SVRIPC,$ERRPC,$TMP0,$TMP1
821          002606 030740          DF2           ;16,18,18,18
822          ;* ITEM 27
823          002610 030313          DM27          ;(ACTION ENABLE WASN'T SET).
824          002612 030873          DH26          ;V/PC,P/PC,REG,WAS
825          002614 002018          DT26          ;SVRIPC,$ERRPC,$TMP0,$BDADR
826          002616 030740          DF2           ;16,18,18,18
827          ;* ITEM 30
828          002620 000000          0             ;NO MESSAGE.
829          002622 030715          DH30          ;REG,WAS,MA,WAS
830          002624 002030          DT30          ;$TMP0,$TMP1,$GDADR,$BDADR
831          002626 030766          DF30          ;18,18,18,B
832          ;* ITEM 31
833          002630 030347          DM31          ;TRAPPED TO 4
834          002632 000000          0             ;NO HEADER
835          002634 002042          DT31          ;$TMP3
836          002636 030766          DF30          ;18
  
```

```

837          .SBTTL START: SETUP AND MAP MEMORY
      ;/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:
      ;* THIS IS THE NORMAL (CA = 200) BEGINNING OF THE PROGRAM.
      ;* NOTE: THIS CODE IS NOT POSITION INDEPENDENT.
      ;/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:
844          002640 105067 176712      START: CLR      SELFLG ;CLEAR SELECT PARAMETER FLAG.
845          002644 000403          BR      STARTA ;GO DO SETUP AND MEMORY MAP.
846          002646 112767 177777 176702 SELECT: MOV     #1,   SELFLG ;SET THE SELECT PARAMETERS FLAG.
847          002654          STARTA:
848          ;SBTTL INITIALIZE THE COMMON TAGS
849          ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
850          002654 012706 0011C )     MOV     #SCMTAG,R6 ;FIRST LOCATION TO BE CLEARED
851          002660 005026          CLR     (R6)+ ;:CLEAR MEMORY LOCATION
852          002662 022706 001140      CMP     #SWR,R6 ;;DONE?
853          002666 001374          BNE     #-8 ;:LOOP BACK IF NO
854          002670 012706 001100      MOV     #STACK,SP ;:SETUP THE STACK POINTER
855          ;;INITIALIZE A FEW VECTORS
856          002674 012737 000610 000024 MOV     #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
857          002702 012737 000340 000026 MOV     #340,@#PWRVEC+2 ;:LEVEL 7
858          002710 016767 011242 011232 MOV     SENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
859          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
860          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
861          002716 013746 000004      MOV     @ERRVEC,-(SP) ;:SAVE ERROR VECTOR
862          002722 012737 002756 000004 MOV     #64$,@ERRVEC ;:SET UP ERROR VECTOR
863          002730 012767 177570 176702 MOV     #DSWR,SWR ;:SETUP FOR A HARDWARE SWITCH REGISTER
864          002736 012767 177570 176176 MOV     #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
865          002744 022777 177777 176166 CMP     #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
866          002752 001012          BNE     66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
867          ;:AND THE HARDWARE SWR IS NOT = -1
868          002754 000403          BR      65$ ;:BRANCH IF NO TIMEOUT
869          002756 012716 002764      64$: MOV     #65$(,SP) ;:SET UP FOR TRAP RETURN
870          002762 000002          RTI
871          002764 012767 000176 176146 65$: MOV     #SWREG,SWR ;:POINT TO SOFTWARE SWR
872          002772 012767 000174 176142 MOV     #DISPREG,DISPLAY
873          003000 012637 000004      66$: MOV     (SP)+,@ERRVEC ;:RESTORE ERROR VECTOR
874
875          003004 005067 176202      CLR     $PASS ;:CLEAR PASS COUNT
876          003010 132767 000200 176207 BIT     #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
877          003016 001403          BEQ     67$ ;:YES,USE NON-APT SWITCH
878          003020 012767 001226 176112 MOV     #SSWREG,SWR ;:NO,USE APT SWITCH REGISTER
879          003026          67$:
880          003026 005077 176470      CLR     LDDISP ;:CLEAR DISPLAY REGISTER STORAGE LOCN
881          003032 005077 176104      CLR     @DISPLAY ;:CLEAR DISPLAY REGISTER
882          ;SBTTL TYPE PROGRAM NAME
883          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
884          I:J #1 ;:FIRST TIME?
885          BNE     68$ ;:BRANCH IF NO
886          003044 001040          CMP     #SENDADR,@#42 ;:ACT-11?
887          BEQ     68$ ;:BRANCH IF YES
888          003052 01434          JSR     R5, SPRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
889          003060 003132          .WORD 69$ ;:ADDRESS OF MESSAGE TO BE TYPED
890          ;SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
891          003062 005737 000042      TST     @#42 ;:ARE WE RUNNING UNDER XXDP/ACT?
892          003066 001015          BNE     70$ ;:BRANCH IF YES
  
```

```
893 003070 126727 176130 000001 CMPB SENV,#1 ;ARE WE RUNNING UNDER APT?
894 003076 001411 BEQ 70$ ;BRANCH IF YES
895 003100 026727 176034 000176 CMP SWR,#SWREG ;SOFTWARE SWITCH REG SELECTED?
896 003106 001010 BNE 71$ ;BRANCH IF NO
897
898 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SGTSWR ROUTINE
899 ;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
900 003110 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
901 003120 000403 JSR PC, SGTSWR ;GO TO THE SUBROUTINE
902 003122 112767 000001 176004 70$: MOVB #1,$AUTOB ;SET AUTO-MODE INDICATOR
903 003130 71$: BR 71$
904 003130 000405 BR 68$ ;GET OVER THE ASCIZ
905 ;:69$: .ASCIZ <CRLF>'CZQMCF0'<CRLF>
906 003144 68$:
907 003144 010700 MOV PC, R0 ;GET CURRENT PROGRAM COUNTER.
908 003146 022700 003146 CMP #, PO ;CHECK IF THE PROGRAM IS RELOCATED.
909 003152 001402 BEQ 10$ ;BR IF PROGAM NOT RELOCATED.
910 003154 000167 175120 JN? RESTAR ;GO TRY TO RELOCATED BEFORE CONTINUING.
911 003160 012767 000003 175414 10$: MOV #3, PRGMAP ;INITIALIZE PROGRAM MAP...LO 64K.
912 003166 005067 175412 CLR PRGMAP+2 ;...HI 64K.
913 003172 005067 175402 CLR RELOCF ;INIT THE RELOCATION FACTOR.
914 003176 105737 001224 TSTB @#SENV ;CHECK FOR APT11
915 003202 001011 BNE 13$ ;BR IF APT11
916 003204 005737 000042 TST @#42 ;CHECK FOR STANDALONE
917 003210 001406 BEQ 13$ ;BR IF STANDALONE
918 003212 023737 000042 000046 CMP @#42,@#46 ;CHECK FOR ACT11
919 003220 001402 BEQ 13$ ;BR IF ACT11
920 ;MUST BE XXDP
921 003222 004767 014122 JSR PC,SAVLDR ;GO SAVE LOADERS
922
923 ;* CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
924 003226 005067 175354 13$: CLR MMAVA ;CLEAR MEM MGMT AVAILABLE FLAG
925 003232 032777 010000 175700 BIT #SW12, @SWP ;CHECK FOR INHIBIT KT11 SWITCH
926 003240 001014 BNE IMPCK ;BRANCH IF SET
927 003242 012737 003272 000004 MOV #IMPCK,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
928 003250 005037 177572 CLR @#SRO ;CLEAR MEM MGMT STATUS REG
929 003254 004767 011020 JSR PC, MMINIT ;MEM MGMT INITIALIZATION ROUTINE.
930 003260 005267 175322 INC MMAVA ;SET MEM MGMT AVAILABLE FLAG
931 003264 004567 020222 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
932 003270 025354 .WORD MMAMES ;ADDRESS OF MESSAGE TO BE TYPED
933 ;"KT11 AVAILABLE"
934
935 ;* CHECK IF CACHE PRESENT, IF SO TURN IT OFF!!!
936 003272 012706 001100 ;IMPCK: MOV #STACK, SP
937 003276 005067 176314 CLR CASFLG ;CLEAR CACHE PRESENT FLAG
938 003302 012737 003324 000004 MOV #MAPMEM,@#ERRVEC
939 003310 052767 000014 174430 BIS #14, IMPCKE
940 003316 012767 000001 176272 MOV #1, CASFLG ;SET CACHE PRESENT FLAG
941
942 ;*****
943 ;* ROUTINE TO MAP ALL OF MEMORY.
944 ;* ONLY FULL 4K BANKS WILL BE RECOGNIZED.
945 ;* R0 = MEMMAP POINTER...LO 64K.
946 ;* R1 = MEMMAP POINTER...HI 64K.
947 ;* R2 = ADDRESS POINTER
948 ;* R3 = BANK POINTER...LO 64K.
```

```
949 ;* R4 = BANK POINTER...HI 64K.
950 ;* R5 = SCRATCH REGISTER.
951 ;*****
952 003324 012706 001100 MAPMEM: MOV #STACK, SP ;RESET THE STACK
953 003330 012700 001524 MOV #MEMMAP,R0 ;SET UP MEMORY MAP POINTER...LO 64K.
954 003334 012701 001526 MOV #MEMMAP+2,R1 ;...HI 64K.
955 003340 005010 CLR (R0) ;CLR MEMORY MAP...LO 64K.
956 003342 005011 CLR (R1) ;...HI 64K.
957 003344 005002 CLR R2 ;CL ADDRESS POINTER TO 0
958 003346 012703 000001 MOV #1, R3 ;SETUP 4K BANK POINTER...LO 64K.
959 003352 005004 CLR R4 ;...HI 64K.
960 003354 005067 175606 CLR STMP3 ;INIT TEMPORARY HIGH ADDRESS BITS.
961 003360 004567 020126 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
962 003364 025421 .WORD MEMMES ;ADDRESS OF MESSAGE TO BE TYPED
963 ;"MEMORY MAP:"
964 003366 012737 003502 000004 MOV #2$, @#ERRVEC ;SET UP TIMEOUT VECTOR
965 003374 011222 1$: MOV (R2), (R2)+ ;READ+WRITE ALL MEMORY
966 003376 022702 017777 BIT #MASK4K,R2 ;CHECK FOR 4K BOUNDARY
967 003402 013744 BNE 1$ ;BRANCH IF MORE IN BANK
968 003404 000310 BIS R3, (R0) ;SET FLAG FOR BANK...LO 64K.
969 003406 000411 BIS R4, (R1) ;...HI 64K.
970 003410 010267 175550 MOV R2, STMP2 ;SAVE ADDRESS POINTER.
971 003414 005367 175544 DEC STMP2 ;ADJUST TO LAST ADR, LAST BANK.
972 003420 005767 175162 TST MMAVA ;CHECK FOR MEM MGMT.
973 003424 001432 BEQ 3$ ;BR IF NO MEM MGMT.
974 003426 0-2767 160000 175530 BIC #160000,STMP2 ;CLEAR BANK BITS ON RELATIVE ADDRESS.
975 003434 013705 172344 MOV @#KIPAR2,R5 ;SAVE KIPAR2.
976 003440 005067 175522 CLR STMP3 ;MAKE SURE HI BITS ARE INIT.
977 003444 006305 ASL R5 ;SHIFT IT 6 PLACES.
978 003446 006305 ASL R5
979 003450 006305 ASL R5
980 003452 006305 ASL R5
981 003454 006305 ASL R5
982 003456 006167 175504 ROL STMP3
983 003462 006305 ASL R5
984 003464 006167 175476 ROL STMP3
985 003470 006067 175470 ADD R5, STMP2 ;MAKE LAST ADR PHYSICAL.
986 003474 005567 175466 ADC STMP3
987 003500 000404 BR 3$ ;GO TO UPDATE POINTERS.
988
989 ;* TIMEOUT TRAPS TO HERE
990 003502 022626 2$: CMP (SP)+, (SP)+ ;RESTORE THE STACK POINTER
991 003504 052702 017777 BIS #MASK4K,R2 ;LAST ADDRESS OF 4K BANK
992 003510 005272 INC R2 ;FIRST ADDRESS OF NEXT BANK.
993 003512 005767 175070 3$: TST MMAVA ;CHECK FOR MEM MGMT
994 003516 001411 BEQ 4$ ;BRANCH IF NO MEM MGMT
995 003520 002737 000200 172344 ADD #200, @#KIPAR2 ;UPDATE THIRD PAR
996 003526 012702 040000 M-/ ASL R2 ;POINT TO START OF THIRD PAR
997 003532 006303 ASL R3 ;UPDATE LO BANK POINTER.
998 003534 006104 ROL R4 ;UPDATE HI BANK POINTER
999 003536 1^0316 BPL 1$ ;BRANCH IF MORE MEMORY TO MAP.
1000 003540 000402 BR 5$ ;EXIT WHEN DONE.
1001
1002 003542 106303 4$: ASLB R3 ;UPDATE MAP POINTER
1003 003544 100313 BPL 1$ ;BRANCH IF NOT YET DONE
1004 003546 012737 025114 000004 5$: MOV #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR
```

```
1005 003554 004767 014632 JSR PC, TYPMAP ;GO TYPE THE MAP.
1006 003560 004567 017726 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1007 003564 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
1008 003566 011067 175742 MOV (R0), SAVTST ;SET UP TEST MAP...LO 64K.
1009 003572 011167 175740 MOV (R1), SAVTST+2 ;...HI 64K.
1010 003576 011000 MOV (R0), R0 ;GET LOW MEM MAP
1011 003600 042760 177760 BIC #177760, R0 ;MASK ALL BUT BOTTOM 4 BANKS
1012 003604 020027 000017 CMP R0, #17 ;CHECK THAT BOTTOM 16K IS ALL THERE!
1013 003610 001530 BEQ GMPR ;BRANCH IF BOTTOM 16K EXISTS
1014 003612 004567 017674 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1015 003616 025524 .WORD INSUFF ;ADDRESS OF MESSAGE TO BE TYPED
1016 ;"FIRST 16K OF MEMORY NOT ALL THERE!"
1017 003620 000000 6S: HALT ;FATAL ERROR HALT...
1018 ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
1019 ;*****
1020 ;* SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
1021 ;* DATI, DATIP, DATO, AND DATOB.
1022 ;*****
1023 003622 012708 001100 TIMEOUT: MOV #STACK, SP ;SET UP THE STACK POINTER.
1024 003626 005067 174754 CLR MMAVA ;CLEAR MEM MGMT AVAILABLE FLAG.
1025 003632 032777 010000 175300 BIT #SW12, @SWR ;CHECK IF MEM MGMT TO BE INHIBITED.
1026 003640 001011 BNE 1S ;BR IF NO MEM MGMT.
1027 003642 012737 003664 000004 MOV #1S, @#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
1028 003650 0050J7 177572 CLR @#SR0 ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
1029 003654 004767 010420 JSR PC, MMINIT ;INIT ALL MEM MGMT REGISTERS.
1030 003660 005267 174722 INC MMAVA ;SET MEM MGMT AVAILABLE FLAG.
1031 003664 1S:
1032 003664 004567 017622 JLN R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1033 003670 025437 .WORD BYTMES ;ADDRESS OF MESSAGE TO BE TYPED
1034 ;"BYTE MEMORY MAP:"
1035 003672 015000 CLR R0 ;SET UP TYPE OUT FLAG.
1036 003674 005002 CLR R2 ;SET ADDRESS POINTER TO ZERO.
1037 003676 012737 003742 000004 MOV #20S, @#ERRVEC ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
1038 003704 105712 10S: TSTB (R2) ;DO DATI ONLY.
1039 003706 032702 000001 BIT #BIT0, R2 ;CHECK FOR WORD ADDRESS.
1040 003712 001001 BNE 11S ;BR IF ODD BYTE ADDRESS.
1041 003714 011212 MCV (R2), (R2) ;DO DATI, DATO...NDP FOR READ ONLY MEM.
1042 003716 151212 11S: BLSB (R2), (R2) ;DO DATI, DATIP, DATOB... NDP FOR READ ONLY MAP.
1043 003720 005700 TST R0 ;CHECK FOR PREVIOUS TYPED.
1044 003722 001023 BNE 30S ;BR IF ALREADY TYPED "FROM".
1045 003724 004567 017562 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1046 003730 025507 .WORD FROM ;ADDRESS OF MESSAGE TO BE TYPED
1047 ;"FROM"
1048 003732 010246 MOV R2, -(SP) ;PUT THE DATA ON THE STACK.
1049 003734 004767 021212 JSR PC, $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1050 003740 000413 BR 29S ;GO TO ADDRESS POINTER UPDATE.
1051 ;* TIME OUTS COME HERE.
1052 003742 022626 20S: CMP (SP)+, (SP)+ ;POP TWO OFF STACK.
1053 003744 005700 TST R0 ;CHECK FOR PREVIOUS TYPED.
1054 003746 001411 BEQ 30S ;BR IF ALREADY TYPED "TO".
1055 003750 004567 017536 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1056 003754 025517 .WORD TO ;ADDRESS OF MESSAGE TO BE TYPED
1057 ;"TO"
1058 003756 005302 DEC R2 ;BACK UP ONE BYTE.
1059 003760 010246 MOV R2, -(SP) ;PUT THE DATA ON THE STACK.
1060 003762 004767 021164 JSR PC, $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
```

```
1061 003766 005202 INC R2 ;RESET ADDRESS POINTER.
1062 003770 005100 COM R0 ;RESET PREVIOUS TYPED FLAG.
1063 003772 005202 29S: INC R2 ;UPDATE ADDRESS POINTER TO NEXT BYTE.
1064 003774 001423 BEQ 31S ;EXIT IF ZERO REACHED.
1065 003776 032702 017777 BIT #MASK4K, R2 ;CHECK FOR 4K BANK BOUNDARY.
1066 004002 001340 BNE 10S ;BR IF MORE THIS 4K BANK.
1067 004004 005767 174576 TST MMAVA ;CHECK IF MEM MGMT IS AVAILABLE.
1068 004010 001735 BFL 10S ;BR IF NO MEM MGMT.
1069 004012 022737 007600 172346 CMP #7600, @#KIPAR3 ;CHECK FOR END OF LAST 4K BANK.
1070 004020 001411 BEQ 31S ;EXIT WHEN ALL DONE.
1071 004022 012702 006000 MOV #60000, R2 ;RESET VIRTUAL ADDRESS POINTER.
1072 004026 013737 172346 172344 MOV @#KIPAR3, @#KIPAR2 ;SAVE MEM MGMT REG FOR TYPED.
1073 004034 052737 000200 172346 ADD #200, @#KIPAR3 ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
1074 004042 000720 BR 10S ;BR BACK TO DO NEXT BANK.
1075 004044 005700 31S: TST R0 ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
1076 004046 001407 BEQ 32S ;BR TO EXIT IF TYPING ALL DONE.
1077 004050 004567 017436 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1078 004054 025517 .WORD TO ;ADDRESS OF MESSAGE TO BE TYPED
1079 ;"TO"
1080 004056 005302 DEC R2 ;BACK ADDRESS POINTER UP ONE BYTE.
1081 004060 010246 MOV R2, -(SP) ;PUT THE DATA ON THE STACK.
1082 004062 004767 021064 JSR PC, $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1083 004066 000000 32S: HALT ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
1084 ;* TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
1085 004070 000654 BR TIMEOUT ;LOOP BACK AND DO AGAIN UPON CONTINUE.
1086
1087 .SBTTL MAP PARITY REGISTERS
1088 ;*****
1089 ;* SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
1090 ;* THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
1091 ;*****
1092
1093 004072 012764 002276 GMPR: MOV #MPRX, R4 ;SET UP POINTER TO PARITY REG EXIST TABLE.
1094 004076 032777 000177 175034 BIT #SW06, @SWR ;CHECK FOR INHIBIT PARITY SWITCH.
1095 004104 001056 BNE GMPRD ;BR IF INHIBIT PARITY.
1096 004106 012703 002076 MCV #MPRO, R3 ;SET UP TABLE POINTER
1097 004112 012737 004134 000004 MOV #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1098 004120 042713 000001 GMPRA: BIC #1, (R3) ;CLEAR FLAG BIT IN TABLE
1099 004124 005773 000000 TST @ (R3) ;DOES THIS MEMORY PARITY REGISTER EXIST.
1100 ;* IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
1101 004130 012324 MOV (R3)+, (R4)+ ;SAVE IT IN THE PARITY REG EXIST TABLE.
1102 004132 000403 BR GMPRC ;SKIP TIMEOUT SERVICE CODE
1103 ;* TIMEOUT COMES HERE
1104 GMPRB: CMP (SP)+, (SP)+ ;RESTORE STACK POINTER
1105 004136 052723 000001 BIS #1, (R3)+ ;SET FLAG TO INDICATE REGISTER NOT PRESENT
1106 004142 005023 GMPRC: CLR (R3)+ ;CLEAR THE MAP...LO 64K.
1107 004144 005023 CLR (R3)+ ;...HI 64K.
1108 004146 005023 CLR (R3)+ ;...AND THE MASK.
1109 004150 020327 002276 CMP R3, #MPRX ;HAVE WE CHECKED ALL REGISTERS?
1110 004154 103761 BLD GMPRA ;NO - GO BACK TO CHECK NEXT ONE
1111 004156 005014 CLR (R4) ;SET TERMINATOR IN PARITY REG EXIST TABLE.
1112 004160 012737 025114 0000J4 MOV #ERRTRP, @#ERRVEC ;RESTORE TRAPCATCHER
1113 004166 005767 176104 TST MPRX ;ANY PARITY REGISTERS PRESENT?
1114 004172 001066 BNE MPAMEM ;YES - GO TEST CONTROLS PRESENT
1115 004174 005677 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1116 004200 025605 .WORD MTR ;ADDRESS OF MESSAGE TO BE TYPED
```

1117
1118 004202 005014 GMPRD: CLR (R4) ;"NO MEMORY PARITY REGISTERS FOUND"
1119 004204 000167 001156 JMP MANUAL ;MAKE SURE TABLE IS CLEAR.
1120 ;AND SKIP ALL CONTROLS TESTING

1121
1122 ;SBTTL MAP PARITY MEMORY
1123 ;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
1124 ;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
1125 ;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
1126 ; - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
1127 ; - PARITY GENERATE OR DETECT LOGIC FAILED
1128 ; - PARITY ERROR BIT FAILED TO SET
1129 ; - PARITY BITS IN MEMORY LOCATION FAILED
1130 ; - I.E. BIT STUCK AT GOOD PARITY VALUE
1131 ;
1132
1133 004210 004767 014054 MPAMEM: JSR PC, CLRPAR ;INITIALIZE ALL PARITY REGISTERS
1134 004214 012767 000001 175322 MOV #1, BITPT ;INITIALIZE 4K POINTER
1135 004222 005067 175320 CLR BITPT+2 ;CLEAR HI 64K POINTER
1136 004226 012702 014000 MOV #14000, R2 ;SET ADR POINTER TO 14000.
1137 004232 005767 174350 TST MMAPA ;CHECK FOR MEM MGMT
1138 004236 001404 BEQ MAPRB ;BRANCH IF NO MEM MGMT
1139 004240 012702 054000 MOV #54000, R2 ;SET ADR POINTER TO PAR2
1140 004244 004767 010030 JSR PC, MMINIT ;SET UP ALL MEMORY MGMT REGISTERS.
1141 ;
1142 ;*****
1143 ;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
1144 ;* THEN WRITE TEST LOCATION VIA DAT0 & READ TEST LOCATION VIA DAT1
1145 ;* THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
1146 ;*****
1147
1148 004250 005067 175264 MAPRB: CLR PMEMAP ;CLEAR THE PARITY MEMORY MAP
1149 004254 005067 175262 CLR PMEMAP+2
1150 004260 012703 002076 15: MOV #MPRO, R3 ;INITIALIZE TABLE ADDRESS
1151 004264 002713 000001 25: BIT #1, (R3) ;IS THIS REGISTER PRESENT?
1152 004270 001052 BNE 35 ;NO - GET THE NEXT ONE
1153 004272 013773 001612 000000 MOV @WWP, @(R3) ;YES - SET WRITE WRONG PARITY
1154 ;AND CLEAR REST OF REGISTER
1155 004300 011212 MOV (R2), (R2) ;WRITE WRONG PARITY
1156 004302 005712 TST (R2) ;READ WRONG PARITY
1157 004304 043773 001612 000000 BIC @WWP, @(R3) ;CLEAR WRITE WRONG PARITY
1158 004312 005773 000000 TST @(R3) ;OTHERWISE, CHECK TO SEE IF THIS
1159 ;CONTROL REGISTER GOT A PARITY
1160 ;ERROR
1161 004316 100014 BPL 65 ;BRANCH IF IT DIDN'T AND CHECK
1162 004320 032773 007740 000000 BIT #7740, @(R3) ;IS IT A CORE PAR. REG.
1163 004326 001404 BEQ 55 ;BRANCH IF NOT.
1164 004330 012763 070032 000006 6: MOV #70032, 6(R3) ;IF IT IS SET UP MASK
1165 004336 000413 BR 75 ;AND BRANCH TO SET BITS.
1166 004340 012763 077772 000006 55: MOV #77772, 6(R3) ;IF MOS SET UP MASK
1167 004346 000407 BR 75 ;AND BRANCH TO SET BIT.
1168 004350 032773 007740 000000 65: BIT #7740, @(R3) ;IF ANY BITS ARE SET
1169 004356 001417 BEQ 35 ;THEN CSR IS MS11-K.
1170 004360 012763 070000 000006 MOV #70000, 6(R3) ;IF MS11-K SET MASK.
1171 004366 056763 175152 000002 75: BIS BITPT, 2(R3) ;SET FLAG IN MAP FOR THIS PARITY REGISTER
1172 004374 056763 175146 000004 BIS BITPT+2, 4(R3)
1173 004402 056767 175136 175130 BIS BITPT, PMEMAP ;SET FLAG IN PARITY MAP
1174 004410 056767 175132 175124 BIS BITPT+2, PMEMAP+2
1175 004416 062703 000010 35: ADD #10, R3 ;STEP UP TO NEXT REGISTER
1176 004422 020327 002276 CMP R3, #MPRX ;ARE WE DONE WITH TABLE?

```

1177 004426 103716          BLD 2$          ;GO BACK TO CHECK FOR ANY MORE!
1178 004430 011212          MOV (R2), (R2) ;CLEAR BAD PARITY
1179 004432 005767 174150   TST MMAPA      ;CHECK FOR MEM MGMT
1180 004436 001414          BEQ 10$        ;BR IF NO MEM MGMT
1181 004440 002737 000200 172344 4$: ADD #200, @#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
1182 004446 006367 175072   ASL BITPT      ;UPDATE BANK POINTER...LO 64K.
1183 004452 006167 175070   ROL BITPT+2    ;...HI 64K.
1184 004456 100441          BR. TMAP      ;BR IF ALL DONE.
1185 004460 023727 172344 001000 CMP @#KIPAR2,#1000 ;THIS CODE TESTS IF MS11-K IS
1186 004466 001013          BNE 12$        ;PRESENT AND IF IT IS I SET
1187 004470 02737 000003 002260 BIT #3,@#MPR14+2 ;THE BIT TO DISABLE ECC IN
1188 004476 001004          BNE 13$        ;THE LOCATION W/P THAT IS
1189 004500 002737 000003 002270 BIT #3,@#MPR15+2 ;USED AS THE COMMAND TO
1190 004506 001400          BEQ 13$        ;WRITE WRONG PARITY.
1191 004510 012737 020004 001612 13$: MOV #20004,@#W.P ;
1192 004516 006767 175022 175000 12$: BIT BITPT, MEMMAP ;CHECK IF BANK EXISTS...LO 64K.
1193 004524 001255          BNE 1$         ;BR IF BANK EXISTS.
1194 004526 006767 175014 174772 BIT BITPT+2, MEMMAP+2 ;...HI 64K.
1195 004534 001251          BNE 1$         ;BR IF BANK EXISTS.
1196 004536 000740          BR 4$         ;BR IF BANK DOESN'T EXIST.
1197 004540 006767 175000 174756 11$: BIT BITPT, MEMMAP ;CHECK IF BANK EXISTS.
1198 004546 001244          BNE 1$         ;BR IF BANK EXISTS.
1199 004550 002702 020000 10$: ADD #20000, R2 ;UPDATE ADDRESS POINTER TO NEXT BANK.
1200 004554 106367 174764   ASLB BITPT    ;MOVE POINTER TO NEXT BANK.
1201 004560 100367          BPL 11$       ;BR IF MORE TO LOOK FOR.
1202
1203 ;*****
1204 ;> ROUTINE T TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
1205 ;* AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
1206 ;*****
1207
1208 004562 004767 013502      TMAP: JSR PC, CLRPAR ;INITIALIZE ALL PARITY REGISTERS PRESENT
1209 004566 004567 016720     JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1210 004572 025462          .WORD MTMAP ;ADDRESS OF MESSAGE TO BE TYPED
1211 ;"PARITY MEMORY MAP:"
1212 004574 012703 002076     MOV #MPRO, R3 ;INITIALIZE TABLE POINTER
1213 004600 032713 000001 1$: BIT #BIT0, (R3) ;CHECK IF THIS REGISTER IS PRESENT.
1214 004604 001046          BNE 2$         ;BR IF NOT PRESENT.
1215 004606 022763 070032 000006 CMP #70032, 6(R3)
1216 004614 001004          BNE 3$         ;
1217 004616 004567 016670     JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1218 004622 026123          .WORD MX3 ;ADDRESS OF MESSAGE TO BE TYPED
1219 ;"CORE PARITY"
1220 004624 000417          BR 5$         ;
1221 004626 022763 077772 000006 3$: CMP #77772, 6(R3)
1222 004634 001004          BNE 4$         ;
1223 004636 004567 016650     JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1224 004642 026142          .WORD MX4 ;ADDRESS OF MESSAGE TO BE TYPED
1225 ;"MOS PARITY"
1226 004644 000407          BR 5$         ;
1227 004646 022763 070000 000006 4$: CMP #70000, 6(R3)
1228 004654 001003          BNF 5$         ;
1229 004656 004567 016630     JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1230 004662 026160          .WORD MX5 ;ADDRESS OF MESSAGE TO BE TYPED
1231 ;"MS11-K CSR"
1232 004664          5$:

```

```

1233 004664 004567 016622     JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1234 004670 026071          .WORD MX1 ;ADDRESS OF MESSAGE TO BE TYPED
1235 ;"REGISTER AT"
1236 004672 011346          MOV (R3),-(SP) ;SAVE (R3) FOR TYPEDOUT
1237 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOC ROUTINE
1238 ;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1239 004674 013746 177776     MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
1240 004700 004767 020004     JSR PC, STYPOC ;GO TO THE SUBROUTINE
1241 004704 004567 016602     JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1242 004710 026110          .WORD MX2 ;ADDRESS OF MESSAGE TO BE TYPED
1243 ;"CONTROLS"
1244 004712 010300          MOV R3, R0 ;SET UP R0 FOR TYPMAP ROUTINE.
1245 004714 005720          TST (R0)+ ;UPDATE POINTER TO MAP.
1246 004716 004767 01341 J JSR PC, TYPMAP ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1247 004722 002703 000010 2$: ADD #10, R3 ;UPDATE TO NEXT REGISTER IN TABLE.
1248 004726 020327 002276     CMP R3, #MPRX ;ARE WE ALL DONE WITH TABLE?
1249 004732 103722          BLO 1$        ;BRANCH IF MORE REGISTERS
1250 004734 004567 016552     JSR R5, SPRINT ;THE REASON I'M OUTPUTTING THIS CRLF
1251 004740 001201          $CRLF ;IS TO GIVE THE PRINTER ENOUGH TIME TO
1252 ;FINISH PRINTING THE MEMORY MAP BEFORE THE RESET OCCURS.
1253 004742 022737 070000 002264 CMP #70000,@#MPR14+6 ;DO WE HAVE MS11-K AT THIS ADDRESS
1254 004750 001006          BNE 7$        ;IF NO BRANCH
1255 004752 043727 002260 001540 BIC @#MPR14+2,@#PMEMAP ;IF YES THEN CLEAR THE BITS IN
1256 004760 043737 002262 001540 BIC @#MPR14+4,@#PMEMAP ;THE PARITY MEMORY MAP.
1257 004766 022737 070000 002274 7$: CMP #70000, @#MPR15+6 ;DO WE HAVE A MS11-K
1258 004774 001031          BNE 9$        ;IF NO GO TO TESTS NOW.
1259 004776 043737 002270 001540 BIC @#MPR15+2,@#PMEMAP ;IF YES I AM GOING TO
1260 005004 043737 002272 001542 BIC @#MPR15+4,@#PMEMAP+2 ;CLEAR THE PARITY INDICATORS
1261 005012 012705 002276     MOV #MPRX, R5 ;FOR THAT PORTION OF MEMORY.
1262 005016 021537 002256 6$: CMP (R5),@#MPR14 ;SEARCH FOR THIS MS11-K CSR IN
1263 005022 001004          BNE 8$        ;AND IF ITS THERE DELETE IT
1264 005024 005015          CLP (R5)
1265 005026 052737 000001 002256 BIS #1,@#MPR14
1266 005034 022537 002266 8$: CMP (R5)+, @#MPR15 ;SEARCH FOR MS11-K CSR IN
1267 005040 001366          BNE 6$        ;THE AVAILABILITY TABLE,
1268 005042 005045          CLR -(R5) ;AND CLEAR ITS ADDRESS FROM THE TABLE
1269 005044 052737 000001 002266 BIS #1, @#MPR15 ;SET BIT0 IN ADDRESS IN CSR TABLE
1270 005052 004567 016434     JSR R5, SPRINT ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1271 005056 006176          .WORD MX5
1272 005060 005737 002276 9$: TST @#MPRX ;ARE THERE ANY PARITY REGISTERS TO TEST?
1273 005064 001002          BNE CTRLS ;IF SO TEST THE BITS IN THE REGISTERS,
1274 005066 000167 000274   JMP MANUAL ;IF NO JUMP OVER REGISTER TESTS.
1275
1276 .SBTTL TEST PARITY REGISTERS
1277 ;*****
1278 ;> SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
1279 ;* CAN BE SET AND CLEARED.
1280 ;* THIS IS A ONCE ONLY TEST.
1281 ;*****
1282
1283 005072 022703 002076     CTRL: MOV #MPRO, R3 ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1284 005076 011302 1$: MOV (R3), R2 ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
1285 005100 002703 000010 ADD #10, R3 ;UPDATE POINTER TO NEXT PAR. REG. ACC.
1286 005104 032702 000001 BIT #1, R2 ;IS THIS REGISTER BEING USED?
1287 005110 001372          BNE 1$        ;GO TO NEXT IF NOT
1288 005112 020327 002276     CMP R3, #MPRX ;ARE WE AT END OF TABLE

```



```
1289 005116 003055 BGT RESCHK ;GO TO NEXT TEST IF YES
1290 005120 005762 177776 TST -2(R3) ;TEST MASK FOR PARITY REGISTER
1291 005124 001764 BEQ 1$ ;IF = 0, THEN DO NOT TEST
1292 005126 016367 177776 174362 MOV -2(R3), RESRVD ;GET MASK FOR REGISTER WE ARE WORKING ON
1293 005134 012700 000001 MOV #1, R0 ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
1294 005140 005012 CLR (R2) ;INITIALIZE THE PARITY REGISTER
1295 005142 011201 MOV (R2), R1 ;READ THE CONTENTS OF THE PARITY REGISTER
1296 005144 046701 174346 BIC RESRVD, R1 ;CLEAR BITS WHICH ARE RESERVED
1297 005150 001405 BEQ 2$ ;CHECK OTHER BITS - BRANCH IF OK
1298 005152 004767 013134 64$: JSR PC, SPRNT ;SET UP VALUES FOR ERROR PRINTING.
1299 005156 004767 014456 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1300 005162 000001 .WORD 1 ;ERROR TYPE CODE.
1301 005164 030067 174326 2$: BIT R0, RESRVD ;IS THIS BIT RESERVED?
1302 005170 001025 BNE 3$ ;YES - DON'T TEST IT
1303 005172 010012 MOV R0, (R2) ;NO - SET THIS BIT IN THE PARITY REGISTER
1304 005174 011201 MOV (R2), R1 ;READ & SAVE CONTENTS OF THE PARITY REGISTER
1305 005176 005012 CLR (R2) ;CLEAR THE PARITY REGISTER
1306 005200 046701 174312 BIT RESRVD, R1 ;CLEAR BIT LOCATIONS THAT ARE RESERVED
1307 005204 000001 R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1308 005206 001405 BEQ 66$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1309 005210 004767 013126 65$: JSR PC, SPRNT0 ;SET UP VALUES FOR ERROR PRINTING.
1310 005214 004767 014420 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1311 005220 000001 .WORD 1 ;ERROR TYPE CODE.
1312 005222 66$:
1313 005222 011201 MOV (R2), R1 ;READ THE CONTENTS OF THE PARITY REGISTER
1314 005224 046701 174266 BIC RESRVD, R1 ;CLEAR BITS WHICH ARE RESERVED
1315 005230 001405 BEQ 3$ ;CHECK OTHER BITS - BRANCH IF OK
1316 005232 004767 013054 67$: JSR PC, SPRNT ;SET UP VALUES FOR ERROR PRINTING.
1317 005236 004767 014376 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1318 005242 000001 .WORD 1 ;ERROR TYPE CODE.
1319 005244 0 5300 3$: ASL R0 ;ROTATE TO GET NEXT BIT TO BE TESTED
1320 005246 103346 BCC 2$ ;BRANCH IF NOT DONE WITH ALL BITS
1321 005250 000712 BR 1$ ;AFTER TESTING FOR BIT 15 GO GET NEXT REGISTER.
1322
1323 ;*****
1324 ;* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
1325 ;* THIS IS A ONCE ONLY TEST.
1326 ;*****
1327
1328 005252 012704 002076 RESCHK: MOV #MPRO, R4 ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1329 005256 010403 1$: MOV R4, R3
1330 005260 062704 006010 ADD #10, R4
1331 005264 032713 000001 BIT #1, (R3) ;IS THIS REGISTER BEING USED
1332 005270 001372 BNE 1$ ;BRANCH IF NO
1333 005272 012773 177777 000000 MOV #-1, @R3 ;SET ALL BITS TO A 1
1334 005300 022704 002276 CMP #MPRX, R4 ;ARE WE AT THE END OF THE TABLE
1335 005304 002764 BLT 1$ ;IF YES THEN WE ARE READY TO TEST
1336 005305 000005 RESET ;RESET THE WORD
1337 005310 012703 MOV #MPRO, R3 ;LOAD INITIAL ADDRESS FOR POINTER
1338 005314 011302 2$: MOV (R3), R2 ;STORE PARITY REGISTER ADDRESS
1339 005316 062703 000010 ADD #10, R3
1340 005322 032702 000001 BIT #1, R2
1341 005326 001372 BNE 2$
1342 005330 022703 002276 CMP #MPRX, R3
1343 005334 002014 BGE MANUAL
1344 005336 011201 MOV (R2), R1 ;GET CONTENTS OF REGISTER
```

```
1345 005340 005012 CLR (R2)
1346 005342 042701 077772 BIC #77772, R1 ;CLEAR BITS NOT EFFECTED BY RESET
1347 005346 005701 TST R1 ;CHECK IF REST WERE CLEARED BY RESET
1348 005350 001405 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1349 005352 004767 012734 64$: JSR PC, SPRNT ;SET UP VALUES FOR ERROR PRINTING.
1350 005356 004767 014256 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1351 005362 000001 .WORD 1 ;ERROR TYPE CODE.
1352 005364 65$:
1353 005364 000753 BR 2$ ;BRANCH BACK TO CHECK NEXT REGISTER
1354
1355
1356 005366 012760 000014 MANUAL: MOV #12, R0 ;SET COUNTER TO CLEAR 12 WORDS.
1357 005372 012701 001562 MOV #FSTADR, R1 ;STARTING AT FSTADR.
1358 005376 005201 1$: CLR (R1)+ ;CLEAR THE LOCATIONS.
1359 005400 005300 DEC R0 ;COUNT.
1360 005402 001375 BNE 1$ ;BR IF MORE.
1361 005404 105767 174146 TSTB SELFLG ;CHECK FOR SELECT PARAMETERS STARTUP.
1362 005410 001005 BNE MANUL1 ;BR IF PARAMETERS TO BE SELECTED.
1363 005412 016767 173546 174154 MOV $TMP2, LSTADR ;SET UP VIRTUAL LAST ADDRESS.
1364 005420 000167 000402 JMP MANUL2 ;SKIP PARAMETER SELECTION SECTION.
```

```
1365 .SBTTL USER PARAMETER SELECTION SECTION
1366 ;*****
1367 ;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
1368 ;*****
1369 005424 012700 000001 MANUL1: MOV #BIT0, R0 ;SET UP BANK POINTER.
1370 005430 065001 CLR R1 ;...HI 64K.
1371 005432 065002 CLR R2 ;CLEAR ADDRESS POINTER.
1372 005434 065003 CLR R3 ;...HI ADDRESS BITS.
1373 005435 004567 016050 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1374 005442 026311 .WORD FADMES ;ADDRESS OF MESSAGE TO BE TYPED
1375 ;"FIRST ADDRESS:"
1376 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SRDOCT ROUTINE
1377 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1378 005444 013746 177777 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
1379 005450 004767 015664 JSR PC, SRDOCT ;GO TO THE SUBROUTINE
1380 005454 002716 000001 BIC #BIT0, (SP) ;MAKE SURE ADDRESS IS ON A WORD BOUNDARY.
1381 005460 065067 174030 CLR SAVTST ;INIT TEST MAP...LO 64K.
1382 005464 065067 174048 CLR SAVTST+2 ;...HI 64K.
1383 005470 062702 020000 15: ADD #20000, R2 ;UPDATE ADDRESS POINTER TO NEXT BANK.
1384 005474 065503 ADC R3
1385 005476 002367 016006 CMP R3, SHIOCT ;CHECK HI ADDRESS BITS.
1386 005502 103403 BLO 25 ;BR IF NOT HI ENOUGH YET.
1387 005504 101006 BHI 35 ;BR IF PAST SELECTED ADDRESS.
1388 005506 002016 CMP R2, (SP) ;CHECK THE LO ADDRESS BITS.
1389 005510 101004 BHI 35 ;BR IF PAST SELECTED ADDRESS.
1390 005512 006300 25: ASL R0 ;UPDATE POINTER...LO 64K.
1391 005514 068101 ROL R1 ;...HI 64K.
1392 005516 100364 BPL 15 ;BR BACK TO CHECK NEXT BANK.
1393 005520 000507 BR 175 ;BR IF OVERFLOW.
1394 005522 000067 173776 35: BIT R0, MEMMAP ;CHECK IF BANK EXISTS.
1395 005526 001003 BNE 45 ;BR IF BANK EXISTS.
1396 005530 000167 173772 BIT R1, MEMMAP+2 ;CHECK HI 64K.
1397 005534 001501 BEQ 175 ;BR IF ADDRESS IN UN-MAPPED BANK.
1398 005536 016704 015746 45: MOV SHIOCT, R4 ;SAVE FIRST ADR HI BITS.
1399 005542 105: JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1400 005542 004567 015744 .WORD LADMES ;ADDRESS OF MESSAGE TO BE TYPED
1401 005546 026376 ;"LAST ADDRESS:"
1402 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SRDOCT ROUTINE
1403 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1404 005550 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
1405 005554 004767 015560 JSR PC, SRDOCT ;GO TO THE SUBROUTINE
1406 005560 005716 TST (SP) ;CHECK IF ADR 0 SELECTED (DEFAULT).
1407 005562 001003 BNE 115 ;BR IF NOT 0 (DEFAULT)
1408 005564 005767 015720 TST SHIOCT ;CHECK HI BITS.
1409 005570 001005 BNE 115 ;BR IF NOT 0 (DEFAULT).
1410 005572 006716 173386 MOV $TMP2, (SP) ;SET UP DEFAULT LAST ADR.
1411 005576 006767 173364 015704 115: MVI $TMP3, SHIOCT
1412 005604 002667 173764 MOV (SP)+, LSTADR ;GET THE DATA.
1413 005610 002467 015674 CMP R4, SHIOCT ;CHECK FOR LAST ADR BELOW FIRST ADR.
1414 005614 101352 BHI 105 ;BR IF LAST BELOW FIRST.
1415 005616 103403 BLO 125 ;BR IF LAST ABOVE FIRST.
1416 005620 021667 173750 CMP (SP), LSTADR ;CHECK FOR LAST BELOW FIRST.
1417 005624 101346 BHI 105 ;BR IF LAST BELOW FIRST.
1418 005626 032716 017777 125: BIT #MASK4K,(S) ;CHECK IF FIRST ADR ON BANK BOUNDARY.
1419 005632 001404 BEQ 135 ;BR IF ON BOUNDARY.
```

```
1421 005634 010067 173730 MOV R0, FADMAP ;SET UP FIRST ADDRESS MAP.
1422 005640 010167 173726 MOV R1, FADMAP+2
1423 005644 060067 173664 135: BIS R0, SAVTST ;SET FLAG IN TEST MAP...LO 64K.
1424 005650 060167 173662 BIS R1, SAVTST+2 ;...HI 64K.
1425 005654 020367 015630 145: CMP R3, SHIOCT ;CHECK FOR PAST LAST ADR.
1426 005660 103404 BLO 155 ;BR IF BELOW LAST ADR.
1427 005662 101020 BHI 165 ;BR IF GONE PAST LAST ADR.
1428 005664 020267 173704 CMP R2, LSTADR ;CHECK FOR PAST LAST ADR.
1429 005670 101015 BHI 165 ;BR IF GONE PAST LAST ADR.
1430 005672 062702 020000 155: ADD #20000, R2 ;UPDATE ADDRESS POINTER.
1431 005676 065503 ADC R3 ;...HI BITS.
1432 005700 006300 ASL R0 ;UPDATE BANK POINTER...LO 64K.
1433 005702 068101 ROL R1 ;...HI 64K.
1434 005704 100415 BMI 175 ;BR IF OVERFLOW.
1435 005706 000067 173612 BIT R0, MEMMAP ;CHECK IF THIS BANK EXISTS.
1436 005712 001354 BNE 135 ;BR IF BANK EXISTS.
1437 005714 000167 173606 BIT R1, MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
1438 005720 001351 BNE 135 ;BR IF BANK EXISTS.
1439 005722 000754 BR 145 ;BR IF BANK DOESN'T EXIST.
1440 005724 000067 173574 165: BIT R0, MEMMAP ;CHECK IF THIS BANK EXISTS.
1441 005730 001010 BNE 205 ;BR IF IT EXISTS.
1442 005732 000167 173570 BIT R1, MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
1443 005736 001005 BNE 205 ;BR IF IT EXISTS.
1444 005740 005713 175: TST (SP)+ ;ADJUST THE STACK.
1445 005742 004567 015544 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1446 005746 026421 .WORD BADADR ;ADDRESS OF MESSAGE TO BE TYPED
1447 ;"ADDRESS IN UNMAPPED BANK?"
1448 005750 000606 BR MANUAL ;LOOP BACK TO THE BEGINNING.
1449 005752 010067 173624 205: MOV R0, LADMAP ;SET UP MAP FOR LAST ADDRESS.
1450 005756 010167 173622 MOV R1, LADMAP+2
1451 005762 005767 172920 215: TST MMVA ;CHECK FOR MEMORY MANAGEMENT.
1452 005766 001404 BEQ 225 ;BR IF NO MEM MGMT.
1453 005770 042716 160000 BIC #160000,(SP) ;ADJUST FSTADR TO VIRTUAL BANK 0.
1454 005774 062716 040000 ADD #40000,(SP) ;...TO VIRTUAL BANK 2.
1455 005000 012667 173556 225: MOV (SP)+, FSADR ;SAVE FIRST ADDRESS OFF THE STACK.
1456 006004 305:
1457 006004 004567 015502 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1458 006010 026456 .WORD CONST ;ADDRESS OF MESSAGE TO BE TYPED
1459 ;"SELECT CONSTANT:"
1460 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SRDOCT ROUTINE
1461 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1462 006012 013746 177176 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
1463 006016 004767 015316 JSR PC, SRDOCT ;GO TO THE SUBROUTINE
1464 006022 012667 173582 MOV (SP)+, .CONST ;SAVE THE CONSTANT
1465 006026 005767 172554 MANUL2: TST MMVA ;CHECK IF MEM MGMT IS AVAILABLE.
1466 006032 001406 BEQ 315 ;BR IF NO MEM MGMT.
1467 006034 002767 160000 173532 BIC #160000,LSTADR ;ADJUST LSTADR TO VIRTUAL BANK 0.
1468 006042 002767 040000 173524 ADD #40000, LSTADR ;...VIRTUAL BANK 2.
1469 006050 002767 000002 173516 315: ADD #2, LSTADR ;ADJUST LAST ADDRESS UP ONE WORD.
1470 006056 002767 000001 173510 BIC #BIT0, LSTADR ;MAKE SURE IT IS A WORD ADDRESS.
1471 006064 002767 017777 315: BIT #MASK4K,LSTADR ;CHECK IF LAST ADR IS ON BANK BOUNDARY.
1472 006072 001004 BNE ;BR IF NOT ON BOUNDARY.
1473 006074 005067 173502 CLR LADMAP ;CLEAR OUT THE LAST ADDRESS MAP.
1474 006100 005067 173500 CLR LADMAP+2
1475
```

/* THE REST OF THE PROGRAM IS POSITION INDEPENDENT CODE, SO THAT IT CAN EXECUTE PROPERLY WHEN THE PROGRAM HAS BEEN RELOADED. THIS IS DONE SO THAT THE FIRST TWO BANKS OF MEMORY CAN BE EXERCISED IN EXACTLY THE SAME MANNER AS THE REST OF MEMORY */

```
1482 006104 016706 173512 START1: MOV -STACK, SP ;SET STACK POINTER
1483 006110 005767 173502 TST CASFLG ;CHECK CACHE PRESENT FLAG
1484 006114 001403 BFN 1$ ;BRANCH IF NO CACHE
1485 006116 052777 000014 173474 BIS #14, @CASREG ;TURN OFF CACHE
1486 006124 012767 006104 172754 1$: MOV #START1,$LPADR ;INIT LOOP ADDRESS.
1487 006132 066767 172442 172746 ADD RELOCF, $LPADR
1488 006140 004767 011372 JSR PC, MAMF ;SET UP MEMORY PARITY ERROR VECTOR
1489 006144 005767 172436 TST MMVA ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
1490 006150 001406 BEQ TST1 ;BRANCH IF NO MEM MGMT.
1491 006152 022737 000001 177572 BIT #BIT0, @#C90 ;CHECK IF MEM MGMT ENABLED.
1492 006160 001002 BNE TST1 ;BR IF MEM MGMT ENABLED.
1493 006162 004767 006112 JSR PC, MMINIT ;SET UP MEM MGMT REGISTERS.
```

```
1494 .SBTTL SECTION 1: MEMORY ADDRESS TESTS
1495 *****
1496 *TEST 1 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1497 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1498 ;* R1 = DATA READ FROM MEMORY (WAS)
1499 ;* R2 = VIRTUAL ADDRESS
1500 ;* R3 = NOT USED
1501 ;* R4 = NOT USED
1502 ;* R5 = BLOCK BOUNDARY BIT MASK.
1503 *****
1504 006166 TST1:
1505 006166 004567 012436 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1506 006172 000001 .WORD 1 ;MINIMUM BLOCK SIZE OF 1 WORDS
1507 ; REQUIRED FOR THIS TEST.
1508 006174 000167 005604 JMP TST32 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
1509 ; AVAILABLE FOR TEST.
1510 ;* UPWARDS WORD ADDRESSING.
1511 006200 004467 006222 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1512 006204 004767 007644 1$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
1513 006210 010012 2$: MOV R0, (R2) ;WRITE VALUE OF ADDRESS INTO ADDRESS
1514 006212 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1515 006214 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1516 006216 001405 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1517 006220 004767 012142 64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1518 006224 004767 013410 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1519 006230 000002 .WORD 2 ;ERROR TYPE CODE.
1520 006232 65$:
1521 006232 062700 000002 ADD #2, R0 ;ADD #2 TO PHYSICAL ADDRESS
1522 006236 030502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1523 006240 001363 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
1524 006242 004767 006736 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
1525
1526 ;* CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
1527 ;* DOWNWARDS WORD ADDRESSING.
1528 006246 004467 006612 JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1529 006252 004767 007576 3$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
1530 006256 162700 000002 4$: SUB #2, R0 ;DEC DATA BY 2
1531 006262 014201 MOV -(R2), R1 ;GET THE DATA FROM MEMORY
1532 006264 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1533 006266 001405 BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1534 006270 004767 012046 66$: JSR PC, SPRNT0 ;SET UP VALUES FOR ERROR PRINTING.
1535 006274 004767 013340 JSR PC, $ERRDR ;*** ERROR *** (GO TYPE A MESSAGE)
1536 006300 000002 .WORD 2 ;ERROR TYPE CODE.
1537 67$:
1538 006302 030502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1539 006304 001364 BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.
1540 006306 004767 007362 JSR PC, MMDDN ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
1541 ;*****  
1542 ;*TEST 2 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY  
1543 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)  
1544 ;* R1 = DATA READ FROM MEMORY (WAS)  
1545 ;* R2 = VIRTUAL ADDRESS  
1546 ;* R3 = NOT USED  
1547 ;* R4 = NOT USED  
1548 ;* R5 = BLOCK BOUNDRY BIT MASK.  
1549 ;*****  
1550 TST2:  
1551 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.  
1552 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.  
1553 ;* UPWARDS BYTE ADDRESSING.  
1554 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1555 JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0  
1556 MOV# R0, (R2)+ ;WRITE VALUE OF ADDRESS INTO ADDRESS  
1557 INC R0 ;ADD ONE TO PHYSICAL ADDRESS  
1558 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
1559 BNE 25 ;BRANCH IF MORE IN CURRENT BLOCK.  
1560 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.  
1561 ;*****  
1562 ;* CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY  
1563 ;* DOWNWARDS BYTE ADDRESSING.  
1564 JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1565 JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0  
1566 DEC R0 ;DEC DATA BY 1  
1567 MOV# -(R2), R1 ;GET THE DATA FROM MEMORY  
1568 CMPB R0, R1 ;CHECK THE DATA...LO BYTE ONLY VALID.  
1569 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
1570 JSR PC, SPRNT0 ;SET UP VALUES FOR ERROR PRINTING.  
1571 JSR PC, SERRR ;*** ERROR *** (GO TYPE A MESSAGE)  
1572 .WORD 3 ;ERROR TYPE CODE.  
1573 65$:  
1574 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
1575 BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.  
1576 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO STAG1.  
1577 ;*****  
1578 ;*TEST 3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.  
1579 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)  
1580 ;* R1 = DATA READ FROM MEMORY (WAS)  
1581 ;* R2 = VIRTUAL ADDRESS  
1582 ;* R3 = NOT USED  
1583 ;* R4 = NOT USED  
1584 ;* R5 = BLOCK BOUNDRY BIT MASK.  
1585 ;*****  
1586 TST3:  
1587 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.  
1588 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.  
1589 ;* DOWNWARDS WORD ADDRESSING.  
1590 JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1591 JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0  
1592 CCM R0 ;COMPLEMENT THE ADR  
1593 ADD #2, R0 ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP  
1594 MOV R0, -(R2) ;PUT DATA INTO MEMORY  
1595 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
1596 ;*****
```

```
1597 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.  
1598 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.  
1599 ;*****  
1600 ;* CHECK COMPLEMENT DATA WRITTEN DOWN  
1601 ;* UPWARDS WORD ADDRESSING.  
1602 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1603 JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0  
1604 CCL R0 ;COMPLEMENT IT  
1605 4$:  
1606 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
1607 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
1608 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
1609 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
1610 JSR PC, SERRR ;*** ERROR *** (GO TYPE A MESSAGE)  
1611 .WORD 2 ;ERROR TYPE CODE.  
1612 65$:  
1613 SUB #2, R0 ;COUNT DOWN WITH ADDRESS  
1614 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
1615 BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.  
1616 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 3$.  
1617 ;*****  
1618 ;*TEST 4 WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK  
1619 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)  
1620 ;* R1 = DATA READ FROM MEMORY (WAS)  
1621 ;* R2 = VIRTUAL ADDRESS  
1622 ;* R3 = NOT USED  
1623 ;* R4 = NOT USED  
1624 ;* R5 = BLOCK BOUNDRY BIT MASK.  
1625 ;*****  
1626 TST4:  
1627 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.  
1628 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.  
1629 ;* UPWARDS BYTE ADDRESSING.  
1630 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1631 JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0  
1632 MOV# R0, (R2)+ ;WRITE BANK # INTO ALL ADDRESSES  
1633 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
1634 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.  
1635 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.  
1636 ;*****  
1637 ;* CHECK THAT DATA WRITTEN ABOVE CAN BE READ  
1638 ;* UPWARDS BYTE ADDRESSING.  
1639 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1640 JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0  
1641 MOV# (R2)+, R1 ;READ THE DATA OUT OF MEMORY  
1642 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
1643 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
1644 JSR PC, SPRNT1 ;SET UP VALUES FOR ERROR PRINTING.  
1645 JSR PC, SERRR ;*** ERROR *** (GO TYPE A MESSAGE)  
1646 .WORD 3 ;ERROR TYPE CODE.  
1647 65$:  
1648 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
1649 BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.  
1650 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 3$.  
1651 ;*****  
1652
```

```

1653 ;*TEST 5 WRITE 1'S COMPLEMENT OF BANK #.
1654 ;*
1655 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1656 ;* R1 = DATA READ FROM MEMORY (WAS)
1657 ;* R2 = VIRTUAL ADDRESS
1658 ;* R3 = NOT USED
1659 ;* R4 = NOT USED
1660 ;* R5 = BLOCK BOUNDARY BIT MASK.
1661 ;*
1662 TST5: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1663 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1664 ;*
1665 ;* DOWNWARDS BYTE ADDRESSING.
1666 JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1667 JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0
1668 CDM R0 ;1'S COMPLEMENT OF BANK #
1669 MOV R0, -(R2) ;PUT 1'S COMPLEMENT OF BANK # INTO MEMORY
1670 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1671 BNE ZS ;BRANCH IF MORE IN CURRENT BLOCK.
1672 JSR PC, MMDDWN ;FIND NEXT BLOCK AND LOOP TO 1$.
1673 ;*
1674 ;* CHECK THAT DATA WRITTEN CAN BE READ.
1675 ;* DOWNWARDS BYTE ADDRESSING.
1676 JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1677 JSR PC, BANKNO ;GET THE BANK # INTO R0
1678 CDM R0 ;SET 1'S COMPLEMENT OF BANK #
1679 MOV R0, -(R2), R1 ;READ DATA OUT OF MEMORY
1680 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1681 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1682 JSR PC, SPRNT0 ;SET UP VALUES FOR ERROR PRINTING.
1683 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1684 .WORD 3 ;ERROR TYPE CODE.
1685 ;*
1686 TST6: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1687 BNE ZS ;BRANCH IF MORE IN CURRENT BLOCK.
1688 JSR PC, MMDDWN ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

```

1689 .SBTTL SECTION 2: WORST CASE NOISE TESTS
1690 ;*
1691 ;* THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
1692 ;* MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
1693 ;*
1694 ;*
1695 ;*TEST 6 WRITE A CONSTANT INTO MEMORY.
1696 ;* THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
1697 ;*
1698 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1699 ;* R1 = DATA READ FROM MEMORY (WAS)
1700 ;* R2 = VIRTUAL ADDRESS
1701 ;* R3 = NOT USED
1702 ;* R4 = NOT USED
1703 ;* R5 = BLOCK BOUNDARY BIT MASK.
1704 ;*
1705 TST6: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1706 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1707 ;*
1708 TST6A: MOV .CONST, R0 ;GET USER CONSTANT
1709 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1710 MOV R0, (R2)+ ;WRITE CONSTANT INTO MEMORY.
1711 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1712 BNE ZS ;BRANCH IF MORE IN CURRENT BLOCK.
1713 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
1714 ;*
1715 ;*TEST 7 READ MEMORY AND COMPARE TO CONSTANT.
1716 ;* IMPORTANT: THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST STN.
1717 ;*
1718 TST7: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1719 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1720 ;*
1721 TST7A: MOV .CONST, R0 ;GET USER CONSTANT
1722 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1723 ;*
1724 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1725 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1726 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1727 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1728 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1729 .WORD 4 ;ERROR TYPE CODE.
1730 ;*
1731 TST8: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1732 BNE ZS ;BRANCH IF MORE IN CURRENT BLOCK.
1733 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
1734 ;* SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
1735 ;* ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
1736 ;* BY SIMPLY "TOGGING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.
1737 BIT #SW08, @SWR ;CHECK THAT LOOP ON TEST BIT SET
1738 BEQ TST10 ;BRANCH IF NOT LOOP ON TEST
1739 MOV @SWR, -(SP) ;GET SWITCH REGISTER DATA.
1740 BIC #177740, (SP) ;CLEAR NON-TEST-NUMBER SWITCHS.
1741 BIC #6, (SP)+ ;CHECK IF TEST 6 IN SWITCHES.
1742 CMP #6, (SP)+ ;CHECK IF TEST 6 IN SWITCHES.
1743 BNE TST10 ;BRANCH IF NOT TEST 6
1744 SUB #1, STSTNM ;RESET TEST NUMBER
1745 SUB #TST7-TST6, SLPADR ;RESET LOOP ADDR
    
```

```

1745 007036 000722          BR      TST6A          ;GO TO TEST 6
1746
1747
1748 ;*****
1749 ;*TEST 10  WORSE CASE NOISE (PARITY) WORD TESTING
1750 ;* CHECK MEMORY WITH A SERIES OF PATTERNS
1751 ;*****
1751 007040          TST10:
1752 007040 004567 011564      JC      R5,    $SCOPE ;GO TO SCOPE ROUTINE.
1753 007044 000000          .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1754 007046 016704 172562      MOV     .MPPAT, R4 ;INITIALIZE PATTERN TABLE POINTER
1755 007052 04767 010560      1$:   JSR     PC,    CKPMER ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
1756 007056 012400          MDV     (R4)+, R0 ;GET THE DATA PATTERN.
1757 007060 001420          BEQ     TST11 ;BR IF END OF TABLE.
1758 007062 004467 005340      JSR     R4,    INIMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1759 007066 010012          MOV     R0,    (R2) ;PUT DATA PATTERN INTO MEMORY.
1760 007070 012201          MDV     (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1761 007072 020001          CMP     R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1762 007074 0C1405          BEQ     65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1763 007076 0C4767 011264      64$:   JSR     PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1764 007102 0C4767 012532      JSR     PC,    SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1765 007106 0C0004          .WORD 4 ;ERROR TYPE CODE.
1766 007110
1767 007110 030502          65$:   BIT     R5,    R2 ;CHECK FOR END OF A BLOCK.
1768 007112 001365          BNE    1$,    ;BRANCH IF MORE IN CURRENT BLOCK.
1769 007114 0C4767 006064      JSR     PC,    MMUP ;FIND NEXT BLOCK AND LOOP TO 2$.
1770 007120 000754          BR     1$,    ;BR BACK TO DO NEXT PATTERN
  
```

```

1771 ;*****
1772 ;*TEST 11  ROTATE A "0" BIT THROUGH A FIELD OF ONES.
1773 ;*****
1774 007122          TST11:
1775 007122 0C4587 011502      JSR     R5,    $SCOPE ;GO TO SCOPE ROUTINE.
1776 007126 0C0000          .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1777 007130 012700 177777      MOV     #-1,   R0 ;SET CHECK WORD
1778 007134 0C4767 007030      JSR     PC,    SETCON ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
1779 007140 0C4467 005262      JSR     R4,    INIMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1780 007144 0C0241          1$:   CLC ;CLEAR CARRY BIT IN PSW
1781 007146 0C4767 007036      JSR     PC,    ROTATE ;ROTATE
1782 007152 016201 177776      MOV     -2(R2), R1 ;GET RESULT
1783 007156 1C3402          BCS    63$ ;BRANCH IF 'C' BIT WAS SET
1784 007160 020001          CMP     R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1785 007162 001405          BEQ    64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1786 007164 0C4767 011176      63$:   JSR     PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1787 007170 0C4767 012444      JSR     PC,    SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1788 007174 000005          .WORD 5 ;ERRDR TYPE CODE.
1789 007176
1790 007176 030502          64$:   BIT     R5,    R2 ;CHECK FOR END OF A BLOCK.
1791 007200 001361          BNE    1$,    ;BRANCH IF MORE IN CURRENT BLOCK.
1792 007202 0C4767 005776      JSR     PC,    MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
1793
1794 ;*****
1795 ;*TEST 12  ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
1796 ;*****
1797 007206          TST12:
1798 007206 0C4567 011416      JSR     R5,    $SCOPE ;GO TO SCOPE ROUTINE.
1799 007212 0C0000          .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1800 007214 0C5000          CLR     R0 ;SET CHECK WORD
1801 007216 0C4767 006746      JSR     PC,    SETCON ;PUT THE CONTENTS OF R0 IN ALL MEMORY
1802 007222 0C4467 005200      JSR     R4,    INIMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1803 007226 0C0261          1$:   SEC ;SET 'C' BIT IN PSW
1804 007230 0C4767 006754      JSR     PC,    ROTATE ;GO ROTATE '1' BIT
1805 007234 016201 177776      MDV     -2(R2), R1 ;GET RESULT
1806 007240 1C3002          BCC    63$ ;BRANCH IF 'C' IS CLEAR
1807 007242 020001          CMP     R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1808 007244 0C1405          BEQ    64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1809 007246 0C4767 011114      63$:   JSR     PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1810 007252 0C4767 012362      JSR     PC,    SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1811 007256 000005          .WORD 5 ;ERROR TYPE CODE.
1812 007260
1813 007260 030502          64$:   BIT     R5,    R2 ;CHECK FOR END OF A BLOCK.
1814 007262 001361          BNE    1$,    ;BRANCH IF MORE IN CURRENT BLOCK.
1815 007264 0C4767 005714      JSR     PC,    MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```

```
1816 ;*TEST 13 3 XOR 9 TEST PATTERN.
1817 ;*
1818 ;*
1819 TST13:
1820 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1821 .WORD 777 ;MINIMUM BLOCK SIZE OF 256. WORDS
1822 ; REQUIRED FOR THIS TEST.
1823 JMP TST14 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
1824 ; AVAILABLE FOR TEST.
1825 .3X9: CLR R0 ;SET UP TEST DATA
1826 MOV #-1, R3 ;SET COM DATA REG
1827 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1828 JSR PC, W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
1829 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1830 BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
1831 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
1832
1833 ;*
1834 ;* CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
1835 ;*
1836 CLR R0 ;SET CHECK WORD
1837 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1838 MOV #64, R4 ;SET 256. WORD COUNTER
1839
1840 11$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1841 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1842 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1843 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1844 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1845 .WORD 7 ;ERROR TYPE CODE.
1846
1847 65$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1848 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1849 BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1850 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1851 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1852 .WORD 7 ;ERROR TYPE CODE.
1853
1854 67$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1855 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1856 BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1857 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1858 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1859 .WORD 7 ;ERROR TYPE CODE.
1860
1861 69$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1862 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1863 BEQ 71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1864 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1865 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1866 .WORD 7 ;ERROR TYPE CODE.
1867
1868 71$: COM R0 ;COMPLEMENT CHECK WORD
1869 DEC R4 ;DECREMENT 256. WORD COUNTER
1870 BNE 12$
1871 COM R0 ;COMPLEMENT CHECK WORD
```

```
1872 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1873 BNE 11$ ;BRANCH IF MORE IN CURRENT BLOCK.
1874 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.
1875
1876 ;*
1877 ;* CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.
1878 ;*
1879 CLR R0
1880 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1881 MOV #64, R4 ;SET 256. WORD COUNTER
1882 MOV #4, R3 ;SET 4 WORD COUNTER
1883
1884 21$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1885 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1886 BEQ 73$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1887 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1888 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1889 .WORD 7 ;ERROR TYPE CODE.
1890
1891 73$: COM R0 ;COMPLEMENT CHECK WORD
1892 COM -(R2) ;COMPLEMENT TEST DATA
1893 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1894 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1895 BEQ 75$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1896 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1897 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1898 .WORD 7 ;ERROR TYPE CODE.
1899
1900 75$: COM R0 ;COMPLEMENT CHECK WORD
1901 COM -(R2) ;COMPLEMENT TEST DATA
1902 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1903 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1904 BEQ 77$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1905 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1906 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1907 .WORD 7 ;ERROR TYPE CODE.
1908
1909 77$: DEC R3 ;DECREMENT 4 WORD COUNTER
1910 BNE 23$ ;BR IF NOT DONE.
1911 COM R0 ;COMPLEMENT CHECK WORD
1912 DEC R4 ;DECREMENT 256. WORD COUNTER
1913 BNE 22$ ;BR IF NOT DONE.
1914 COM R0 ;COMPLEMENT CHECK WORD
1915 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1916 BNE 21$ ;BRANCH IF MORE IN CURRENT BLOCK.
1917 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
1918 ;*TEST 14 COMPLEMENT 3 XOR 9 TEST PATTERN
1919 ;*
1920 ;*
1921 007614 ;
1922 007614 0C4567 011010 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1923 007620 0C0777 .WORD 777 ;MINIMUM BLOCK SIZE OF 256. WORDS
1924 ; REQUIRED FOR THIS TEST.
1925 007622 0C0167 000316 JMP TST15 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
1926 ; AVAILABLE FOR TEST.
1927 007626 012700 177777 MOV #-1, R0 ;SET UP TEST DATA
1928 007632 005003 CLR R3 ;SET COM DATA REG
1929 007634 0C4467 004566 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1930 007640 0C4767 006412 1$: JSR PC, W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
1931 007644 0C0502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1932 007646 0C1374 BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
1933 007650 0C4767 005330 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
1934
1935
1936 ;*
1937 ;* CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
1938 ;*
1939 007654 012700 177777 MOV #-1, R0 ;SET CHECK WORD
1940 007660 0C4467 004542 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1941 007664 012704 000100 11$: MOV #64., R4 ;SET 256. WORD COUNTER
1942 007670 12$:
1943 007670 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1944 007672 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1945 007674 0C1405 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1946 007676 0C4767 010464 64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1947 007702 0C4767 011732 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1948 007706 0C0007 .WORD 7 ;ERROR TYPE CODE.
1949 007710 65$:
1950 007710 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1951 007712 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1952 007714 0C1405 BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1953 007716 0C4767 010444 66$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1954 007722 0C4767 011712 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1955 007726 0C0007 .WORD 7 ;ERROR TYPE CODE.
1956 007730 67$:
1957 007730 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1958 007732 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1959 007734 0C1405 BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1960 007736 0C4767 010424 68$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1961 007742 0C4767 011672 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1962 007746 0C0007 .WORD 7 ;ERROR TYPE CODE.
1963 007750 69$:
1964 007750 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1965 007752 020001 C.-> R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1966 007754 0C1405 BEQ 71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1967 007756 0C4767 010404 70$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1968 007762 0C4767 011652 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1969 007766 0C0007 .WORD 7 ;ERROR TYPE CODE.
1970 007770 71$:
1971 007770 0C05100 COM R0 ;COMPLEMENT CHECK WORD
1972 007772 0C5304 DEC R4 ;DECREMENT 256. WORD COUNTER
1973 007774 0C1335 BNE 12$
```

```
1974 007776 0C05100 COM R0 ;COMPLEMENT CHECK WORD
1975 010000 0C0502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
1976 010002 0C1330 BNE 11$ ;BRANCH IF MORE IN CURRENT BLOCK.
1977 010004 0C4767 005174 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.
1978
1979 ;*
1980 ;* CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
1981 ;*
1982 010010 012700 177777 MOV #-1, R0 ;SET UP CHECK WORD.
1983 010014 0C4467 004406 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1984 010020 012704 000100 21$: MOV #64., P4 ;SET 256. WORD COUNTER
1985 010024 012703 000004 22$: MOV #4., R3 ;SET 4 WORD COUNTER
1986 010030 23$:
1987 010030 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1988 010032 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1989 010034 0C1405 BEQ 72$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1990 010036 0C4767 010324 72$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1991 010042 0C4767 011572 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1992 010046 0C0007 .WORD 7 ;ERROR TYPE CODE.
1993 010050 73$:
1994 010050 0C05100 COM R0 ;COMPLEMENT CHECK WORD
1995 010052 0C05142 COM -(R2) ;COMPLEMENT TEST DATA
1996 010054 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1997 010056 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
1998 010060 0C1405 BEQ 75$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
1999 010062 0C4767 010300 74$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2000 010066 0C4767 011546 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2001 010072 0C0007 .WORD 7 ;ERROR TYPE CODE.
2002 010074 75$:
2003 010074 0C05100 COM R0 ;COMPLEMENT CHECK WORD
2004 010076 0C5142 COM -(R2) ;COMPLEMENT TEST DATA
2005 010100 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2006 010102 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2007 010104 0C1405 BEQ 77$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2008 010106 0C4767 010254 76$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2009 010112 0C4767 011522 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2010 010116 0C0007 .WORD 7 ;ERROR TYPE CODE.
2011 010120 77$:
2012 010120 0C05303 DEC R3 ;DECREMENT 4 WORD COUNTER
2013 010122 0C1342 BNE 23$ ;BR IF NOT DONE.
2014 010124 0C05100 COM R0 ;COMPLEMENT CHECK WORD
2015 010126 0C5304 DEC R4 ;DECREMENT 256. WORD COUNTER
2016 010130 0C1335 BNE 22$ ;BR IF NOT DONE.
2017 010132 0C05100 COM R0 ;COMPLEMENT CHECK WORD
2018 010134 0C0502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2019 010136 0C1330 BNE 21$ ;BRANCH IF MORE IN CURRENT BLOCK.
2020 010140 0C4767 005040 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 21$.
```



```
2021 ;*****  
2022 ;* TEST 15 MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY  
2023 ;*****  
2024 TST15:  
2025 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.  
2026 .WORD 777 ;MINIMUM BLOCK SIZE OF 256. WORDS  
2027 ; REQUIRED FOR THIS TEST.  
2028 JMP TST16 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK  
2029 ; AVAILABLE FOR TEST.  
2030 MOV #401, R0 ;SET UP PARITY "ALL ZEROS" PATTERN  
2031 MOV #-1, R3 ;SET COM DATA REG  
2032 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
2033 JSR PC, W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.  
2034 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
2035 BNE 13 ;BRANCH IF MORE IN CURRENT BLOCK.  
2036 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 15.  
2037  
2038 ;*****  
2039 ;* CHECK PARITY 3 XOR 9 PATTERN, WRITTEN ABOVE.  
2040 ;*****  
2041 MOV #401, R0 ;RESET PARITY "ALL ZEROS" PATTERN.  
2042 MOV #-1, R3 ;RESET PARITY ALL ONES PATTERN.  
2043 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
2044 .WORD #64, R4 ;SET 256. WORD COUNTER  
2045  
2046 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2047 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2048 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2049 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2050 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2051 .WORD 7 ;ERROR TYPE CODE.  
2052  
2053 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2054 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2055 BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2056 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2057 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2058 .WORD 7 ;ERROR TYPE CODE.  
2059  
2060 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2061 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2062 BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2063 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2064 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2065 .WORD 7 ;ERROR TYPE CODE.  
2066  
2067 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2068 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2069 BEQ 71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2070 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2071 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2072 .WORD 7 ;ERROR TYPE CODE.  
2073  
2074 MOV R0, -(S) ;SAVE R0  
2075 MOV R3, R0 ;PUT R3 INTO R0  
2076 MOV (SP)+, R3 ;PUT SAVED R0 INTO R3
```

```
2077 010334 005304 DEC R4 ;COUNT 256. WORDS  
2078 010335 001333 BNE 12$ ;BRANCH IF MORE  
2079 010340 010046 MOV R0, -(SP) ;SAVE R0  
2080 010342 010300 MOV R3, R0 ;PUT R3 INTO R0  
2081 010344 012503 MOV (SP)+, R3 ;PUT SAVED R0 INTO R3  
2082 010346 005002 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
2083 010350 001324 BNE 11$ ;BRANCH IF MORE IN CURRENT BLOCK.  
2084 010352 004767 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.  
2085  
2086 ;*****  
2087 ;* CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.  
2088 ;*****  
2089 MOV #401, R0 ;SET UP PARITY "ALL ZEROS" PATTERN.  
2090 MOV #-1, R3 ;SET UP ALL ONES PATTERN.  
2091 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
2092 .WORD #64, R4 ;SET 256. WORD COUNTER  
2093  
2094 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2095 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2096 BEQ 73$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2097 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2098 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2099 .WORD 7 ;ERROR TYPE CODE.  
2100  
2101 COM R0 ;COMPLEMENT CHECK WORD  
2102 COM -(R2) ;COMPLEMENT TEST DATA  
2103 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2104 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2105 BEQ 75$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2106 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2107 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2108 .WORD 7 ;ERROR TYPE CODE.  
2109  
2110 COM R0 ;COMPLEMENT CHECK WORD  
2111 COM -(R2) ;RESTORE DATA  
2112 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2113 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2114 BEQ 77$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2115 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2116 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2117 .WORD 7 ;ERROR TYPE CODE.  
2118  
2119 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2120 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2121 BEQ 79$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2122 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2123 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2124 .WORD 7 ;ERROR TYPE CODE.  
2125  
2126 COM R0 ;COMPLEMENT CHECK WORD  
2127 COM -(R2) ;COMPLEMENT TEST DATA  
2128 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.  
2129 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2130 BEQ 81$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2131 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2132 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
```

```

2133 010530 000007      .WORD 7      ;ERROR TYPE CODE.
2134 010532
2135 010532 005100      COM R0      ;COMPLEMENT CHECK WORD
2136 010534 005100      COM -(R2)   ;RESTORE DATA
2137 010536 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2138 010540 020001      CMP R0, R1  ;COMPARE THE CHECK WORD WITH THE DATA READ.
2139 010542 001405      BEQ 85$    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2140 010544 004767      JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2141 010550 004767 007616 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2142 010554 000007      .WORD 7      ;ERROR TYPE CODE.
2143 010556
2144 010556 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2145 010560 020001      CMP R0, R1  ;COMPARE THE CHECK WORD WITH THE DATA READ.
2146 010562 001405      BEQ 85$    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2147 010564 004767 007576 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2148 010570 004767 011044 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2149 010574 000007      .WORD 7      ;ERROR TYPE CODE.
2150 010576
2151 010576 005100      COM R0      ;COMPLEMENT CHECK WORD
2152 010600 005142      COM -(R2)   ;COMPLEMENT TEST DATA
2153 010602 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2154 010604 020001      CMP R0, R1  ;COMPARE THE CHECK WORD WITH THE DATA READ.
2155 010606 001405      BEQ 87$    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2156 010610 004767 007552 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2157 010614 004767 011020 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2158 010620 000007      .WORD 7      ;ERROR TYPE CODE.
2159 010622
2160 010622 005100      COM R0      ;COMPLEMENT CHECK WORD
2161 010624 005142      COM -(R2)   ;RESTORE DATA
2162 010626 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2163 010630 020001      CMP R0, R1  ;COMPARE THE CHECK WORD WITH THE DATA READ.
2164 010632 001405      BEQ 89$    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2165 010634 004767 007526 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2166 010640 004767 0107.1 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2167 010644 000007      .WORD 7      ;ERROR TYPE CODE.
2168 010646
2169 010646 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2170 010650 020001      CMP R0, R1  ;COMPARE THE CHECK WORD WITH THE DATA READ.
2171 010652 001405      BEQ 91$    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2172 010654 004767 007506 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2173 010660 004767 010754 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2174 010664 000007      .WORD 7      ;ERROR TYPE CODE.
2175 010666
2176 010666 005100      COM R0      ;COMPLEMENT CHECK WORD
2177 010670 005142      COM -(R2)   ;COMPLEMENT TEST DATA
2178 010672 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2179 010674 020001      CMP R0, R1  ;COMPARE THE CHECK WORD WITH THE DATA READ.
2180 010676 001405      BEQ 93$    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2181 010700 004767 007462 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2182 010704 004767 010730 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2183 010710 000007      .WORD 7      ;ERROR TYPE CODE.
2184 010712
2185 010712 005100      COM R0      ;COMPLEMENT CHECK WORD
2186 010714 005142      COM -(R2)   ;RESTORE DATA
2187 010716 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2188 010720 020001      CMP R0, R1  ;COMPARE THE CHECK WORD WITH THE DATA READ.
    
```

```

2189 010722 001405      BEQ 95$    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2190 010724 004767 007436 JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2191 010730 004767 010704 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2192 010734 000007      .WORD 7      ;ERROR TYPE CODE.
2193 010736
2194 010736 010046      MOV R0, -(SP) ;SAVE R0
2195 010740 010300      MOV R3, R0  ;PUT R3 INTO R0
2196 010742 012603      MOV (SP)+, R3 ;PUT SAVED P0 INTO R3
2197 010744 005304      DEC R4     ;DECREMENT 256. WORD COUNTER
2198 010746 001213      BNE 22$   ;BRANCH IF MORE.
2199 010750 010046      MOV R0, -(SP) ;SAVE R0
2200 010752 010300      MOV R3, R0  ;PUT R3 INTO R0
2201 010754 012603      MOV (SP)+, R3 ;PUT SAVED P0 INTO R3
2202 010756 000502      BIT R5, R2  ;CHECK FOR END OF A BLOCK.
2203 010760 001204      BNE 21$   ;BRANCH IF MORE IN CURRENT BLOCK.
2204 010762 004767 004216 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
2205
2206
2207
2208
2209 010766
2210 010766 004567 007636 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
2211 010772 000777      .WORD 777   ;MINIMUM BLOCK SIZE OF 256. WORDS
2212
2213 010774 000167 000610 JMP TST17    ;REQUIRED FOR THIS TEST.
2214
2215 011000 012700 177777 MOV #-1, R0 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2216 011004 012703 000401 MOV #401, R3 ;AVAILABLE FOR TEST.
2217 011010 004467 003412 JSR R4, INITMM ;SET UP ALL ONES PATTERN
2218 011014 004767 005236 JSR R4, INITMM ;SET UP PARITY "ALL ZEROS" PATTERN
2219 011020 000502      BIT R5, R2  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2220 011022 001374      BNE 1$     ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2221 011024 004767 004154 JSR PC, MMUP ;CHECK FOR END OF A BLOCK.
2222
2223
2224
2225
2226 011030 012700 177777 MOV #-1, R0 ;BRANCH IF MORE IN CURRENT BLOCK.
2227 011034 012703 000401 MOV #401, R3 ;FIND NEXT BLOCK AND LOOP TO 1$.
2228 011040 004467 003362 JSR R4, INITMM ;CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2229 011044 012704 000100 MOV #64, R4 ;SET 256. WORD COUNTER
2230 011050
2231 011050 012201      MOV (R2)+, R1 ;SET UP ALL ONES PATTERN
2232 011052 020001      CMP R0, R1  ;SET UP PARITY "ALL ZEROS" PATTERN
2233 011054 001405      BEQ 65$    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2234 011056 004767 007304 JSR PC, SPRNT2 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2235 011062 004767 010552 JSR PC, SERROR ;CHECK FOR END OF A BLOCK.
2236 011066 000007      .WORD 7      ;BRANCH IF MORE IN CURRENT BLOCK.
2237 011070
2238 011070 012201      MOV (R2)+, R1 ;FIND NEXT BLOCK AND LOOP TO 1$.
2239 011072 020001      CMP R0, R1  ;CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2240 011074 001405      BEQ 67$    ;SET 256. WORD COUNTER
2241 011076 004767 007264 JSR PC, SPRNT2 ;SET UP ALL ONES PATTERN
2242 011102 004767 010532 JSR PC, SERROR ;SET UP PARITY "ALL ZEROS" PATTERN
2243 011106 000007      .WORD 7      ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2244 011110      67$      ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
    
```

```
2245 011110 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2246 011112 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2247 011114 001405      BEQ   69$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2248 011116 004767 007244 68$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2249 011122 004767 010512 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2250 011126 000007      .WORD 7           ;ERROR TYPE CODE.
2251 011130
2252 011130 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2253 011132 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2254 011134 001405      BEQ   71$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2255 011136 004767 007224 70$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2256 011142 004767 010472 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2257 011146 000007      .WORD 7           ;ERROR TYPE CODE.
2258 011150
2259 011150 010046      MOV    R0, -(SP)   ;SAVE R0
2260 011152 010300      MOV    R3, P0      ;PUT R3 INTO R0
2261 011154 012503      MOV    (SP)+, R3   ;PUT SAVED P0 INTO R3
2262 011155 005304      DEC   R4           ;COUNT 256. WORDS
2263 011160 001333      BNE   12$         ;BRANCH IF MORE
2264 011162 010046      MOV    R0, -(SP)   ;SAVE R0
2265 011164 010300      MOV    R3, R0      ;PUT R3 INTO R0
2266 011166 012603      MOV    (SP)+, R3   ;PUT SAVED P0 INTO R3
2267 011170 030502      BIT   R5, R2       ;CHECK FOR END OF A BLOCK.
2268 011172 001324      BNE   11$         ;BRANCH IF MORE IN CURRENT BLOCK.
2269 011174 004767 004004 JSR   PC, MMUP     ;FIND NEXT BLOCK AND LOOP TO 11$.
2270
2271 ;*****
2272 ;* CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 6 PATTERN.
2273 ;*****
2274 011200 012700 177777      MOV    #-1, R0     ;SET UP ALL ONES PATTERN
2275 011204 0-27C3 000401      MOV    #401, R3    ;SET UP PARITY "ALL ZEROS" PATTERN
2276 011210 004467 003212 JSR   R4, INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2277 011214 012704 000100      MOV    #64, R4     ;SET 256. WORD COUNTER
2278
2279 011220 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2280 011222 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2281 011224 001405      BEQ   73$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2282 011226 004767 007134 72$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2283 011232 004767 010402 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2284 011236 000007      .WORD 7           ;ERROR TYPE CODE.
2285 011240
2286 011240 005100      COM    R0           ;COMPLEMENT CHECK WORD
2287 011242 005142      COM    -(R2)       ;COMPLEMENT TEST DATA
2288 011244 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2289 011246 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2290 011250 001405      BEQ   75$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2291 011252 004767 007110 74$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2292 011256 004767 010356 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2293 011262 000007      .WORD 7           ;ERROR TYPE CODE.
2294 011264
2295 011264 005100      COM    R0           ;COMPLEMENT CHECK WORD
2296 011266 005142      COM    -(R2)       ;RESTORE DATA
2297 011270 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2298 011272 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2299 011274 001405      BEQ   77$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2300 011276 004767 007064 76$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
2301 011302 004767 010332 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2302 011306 000007      .WORD 7           ;ERROR TYPE CODE.
2303 011310
2304 011310 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2305 011312 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2306 011314 001405      BEQ   79$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2307 011316 004767 007044 78$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2308 011322 004767 010312 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2309 011326 000007      .WORD 7           ;ERROR TYPE CODE.
2310 011330
2311 011330 015100      COM    R0           ;COMPLEMENT CHECK WORD
2312 011332 005142      COM    -(R2)       ;COMPLEMENT TEST DATA
2313 011334 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2314 011336 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2315 011340 001405      BEQ   81$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2316 011342 004767 007020 80$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2317 011346 004767 010266 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2318 011352 000007      .WORD 7           ;ERROR TYPE CODE.
2319 011354
2320 011354 005100      COM    R0           ;COMPLEMENT CHECK WORD
2321 011356 005142      COM    -(R2)       ;RESTORE DATA
2322 011360 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2323 011362 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2324 011364 001405      BEQ   83$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2325 011366 004767 006774 82$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2326 011372 004767 010242 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2327 011376 000007      .WORD 7           ;ERROR TYPE CODE.
2328 011400
2329 011400 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2330 011402 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2331 011404 001405      BEQ   85$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2332 011406 004767 006754 84$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2333 011412 004767 010222 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2334 011416 000007      .WORD 7           ;ERROR TYPE CODE.
2335 011420
2336 011420 005100      COM    R0           ;COMPLEMENT CHECK WORD
2337 011422 005142      COM    -(R2)       ;RESTORE DATA
2338 011424 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2339 011426 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2340 011430 001405      BEQ   87$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2341 011432 004767 006730 86$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2342 011436 004767 010176 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2343 011442 000007      .WORD 7           ;ERROR TYPE CODE.
2344 011444
2345 011444 005100      COM    R0           ;COMPLEMENT CHECK WORD
2346 011446 005142      COM    -(R2)       ;RESTORE DATA
2347 011450 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2348 011452 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2349 011454 001405      BEQ   89$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2350 011456 004767 006704 88$: JSR   PC, SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2351 011462 004767 010152 JSR   PC, SERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2352 011466 000007      .WORD 7           ;ERROR TYPE CODE.
2353 011470
2354 011470 012201      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2355 011472 020001      CMP    R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2356 011474 001405      BEQ   91$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
```

CZQMCF0 0-124K MEMORY EXERCISER, 16K VER MACY11 30A(1052) 20-FEB-78 07:56 PAGE 50
 CZQMCF.P11 14-FEB-78 08:19 T16 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN. SEQ 0132

```

2357 011476 004767 006664 90S: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2358 011502 004767 010132 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2359 011506 000007 .WORD 7 ;ERROR TYPE CODE.
2360 011510
2361 011510 005100 91S: COM R0 ;COMPLEMENT CHECK WORD
2362 011512 005142 COM -(R2) ;COMPLEMENT TEST DATA
2363 011514 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2364 011516 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2365 011520 001405 BEQ 93S ;BRANCH OVER ERROR CALL IF GOOD DATA.
2366 011522 004767 006640 92S: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2367 011526 004767 010106 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2368 011532 000007 .WORD 7 ;ERROR TYPE CODE.
2369 011534
2370 011534 005100 93S: COM R0 ;COMPLEMENT CHECK WORD
2371 011536 005142 COM -(R2) ;RESTORE DATA
2372 011540 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2373 011542 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2374 011544 001405 BEQ 95S ;BRANCH OVER ERROR CALL IF GOOD DATA.
2375 011548 004767 006614 94S: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2376 011552 004767 010062 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2377 011556 000007 .WORD 7 ;ERROR TYPE CODE.
2378 011560
2379 011560 010046 95S: MOV R0, -(SP) ;SAVE R0
2380 011562 010300 MOV R3, R0 ;PUT R3 INTO R0
2381 011564 012603 MOV (SP)+, R3 ;PUT SAVED P0 INTO R3
2382 011566 005304 DEC R4 ;DECREMENT 256. WORD COUNTER
2383 011570 001213 BNE 22S ;BRANCH IF MORE.
2384 011572 010046 MOV R0, -(SP) ;SAVE R0
2385 011574 010300 MOV R3, R0 ;PUT R3 INTO R0
2386 011576 012603 MOV (SP)+, R3 ;PUT SAVED P0 INTO R3
2387 011600 030502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2388 011602 001204 BNE 21S ;BRANCH IF MORE IN CURRENT BLOCK.
2389 011604 004767 003374 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 21S.

```

CZQMCF0 0-124K MEMORY EXERCISER, 16K VER MACY11 30A(1052) 20-FEB-78 07:56 PAGE 51
 CZQMCF.P11 14-FEB-78 08:19 T17 WORSE CASE NOISE PARITY BYTE TESTING SEQ 0133

```

2390 *****
2391 ;* TEST 17 WORSE CASE NOISE PARITY BYTE TESTING
2392 ;* CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
2393 ;* 1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
2394 ;* 2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
2395 ;* 3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
2396 ;* 4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
2397 ;* *****
2398 011610 TST'17:
2399 011610 004567 007014 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
2400 011614 000000 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2401 011616 005787 170454 WWPB0: TST MPRX ;CHECK FOR ANY PARITY MEMORY.
2402 011622 001404 BEQ 1S ;BR IF NO PARITY MEMORY.
2403 011624 002777 000100 167306 BIT #SW06, @SWK ;CHECK FOR INHIBIT PARITY SWITCH.
2404 011632 001402 BEQ 2S ;BR IF NOT SET.
2405 011634 000167 000622 1S: JUMP TST20 ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2406 011640 005000 2S: CLR R0 ;ZERO TO BE PUT IN ALL MEMORY.
2407 011642 004767 004322 JSR PC, SETCON ;ROUTINE TO LOAD ALL MEMORY.
2408 011646 004467 002554 JSR R4, INITWM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2409 011652 006767 167666 WWPBYT: BIT BITPT, PMEMAP ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
2410 011660 001010 BNE 2S ;BR IF PARITY MEM.
2411 011662 006767 167360 167652 BIT BITPT+2, PMEMAP+2 ;...HI 64K.
2412 011670 001004 BNE 2S ;BR IF PARITY MEM.
2413 011672 005002 BIS R5, R2 ;POINT TO END OF BLOCK.
2414 011674 005202 INC R2 ;FIRST ADR OF NEXT BLOCK.
2415 011676 000167 000540 JMP WWPB5 ;BR TO FIND NEXT BLOCK.
2416 011702 004767 006674 2S: JSR PC, SETAE ;SET ACTION ENABLE (EVEN IF BANK0.)
2417 011706 004767 005724 JSR PC, CKPMER ;CHECK FOR ANY NON TRAP PARITY ERRORS.
2418 011712 010227 000114 WWPB1: CMP R2, #114 ;CHECK IF POINTING TO PARITY ERROR VECTOR.
2419 011716 001004 BNE 3S ;BR IF NOT AT VECTOR.
2420 011720 002702 000004 ADD #4, R2 ;SKIP PARITY VECTOR.
2421 011724 000167 000512 JMP WWPB5 ;CHECK FOR BLOCK END.
2422 011730 111201 3S: MOVB (R2), R1 ;CHECK IF BYTE STILL CLEARED.
2423 011732 001405 BEQ 64S ;BRANCH OVER ERROR CALL IF GOOD DATA.
2424 011734 004767 006352 64S: JSR PC, SPRNT ;SET UP VALUES FOR ERROR PRINTING.
2425 011740 004767 007674 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2426 011744 000011 .WORD 11 ;ERROR TYPE CODE.
2427 011746
2428 011746 105067 167606 65S: CLRB DEFLG ;CLEAR ODD/EVEN FLAG.
2429 011752 112700 000252 MOVB #252, R0 ;SET UP DATA...EV N, SETS PARITY BIT.
2430 011756 100012 WWPB2: MCVB R0, (R2) ;MOV DATA INTO TEST LOCATION.
2431 011760 016703 MOV MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
2432 011764 006703 000000 10S: BIS WWP,@(R3) ;SET WRITE WRONG PARITY.
2433 011772 002703 000001 BIT #AE,@(R3)+ ;...HI 64K.
2434 011776 005713 TST (R3) ;CHECK FOR TABLE TERMINATOR.
2435 012000 001371 BNE 10S ;BR IF MORE REGS IN TABLE.
2436 ;* SET WRONG PARITY IN LOCATION UNDER TEST.
2437 012002 110012 MCVB R0, (R2) ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
2438 012004 016703 167620 MCV MPRX, R3 ;GET PARITY REG TABLE POINTER.
2439 012010 006703 167576 11S: BIC WWP, @(R3)+ ;CLEAR WRITE WRONG PARITY.
2440 012014 005713 TST (R3) ;CHECK FOR TABLE TERMINATOR.
2441 012016 001374 BNE 11S ;BR IF MORE PARITY REGISTERS.
2442 012020 016703 167606 000114 MOV #PTRP, @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
2443 ;* DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
2444 NEGB (R2) ;DATIP (DATOB AND COM PARITY BIT.)
2445 ;* SHOULD HAVE TRAPPED TO PTRP.

```

```
2446 012030 016737 167602 000114      MOV    .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
2447 012036 004767 006700      JSR    PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2448 012042 004767 007572      JSR    PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2449 012046 000012      .WORD 12 ;ERROR TYPE CODE.
2450 012050 000562      BR     WWPB4 ;SKIP TRAP SERVICE.
2451
2452
2453 012052 016737 167560 000114      ;* EXPECTED PARITY MEMORY TRAPS COME HERE.
2454 012060 022626      MOV    .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
2455 012062 016703 167540      CMP    (SP)+, (SP)+ ;RESET THE STACK POINTER AFTER TRAP.
2456 012066 032713 000001      MOV    #MPRO, R3 ;GET PARITY REG AND MAP TABLE POINTER.
2457 012072 001003      BIT    #BIT0, (R3) ;CHECK IF THIS REGISTER EXISTS.
2458 012074 017301 000000      BNE    22$ ;BR IF IT DOESN'T EXIST.
2459 012100 100413      MOV    @(R3), R1 ;GET THE CONTENTS.
2460 012102 002703 000010      BMI    23$ ;BR IF ERROR FLAG SET.
2461 012106 020367 167516      ADD    #10, R3 ;MOVE POINTER TO NEXT REG.
2462 012112 103765      CMP    R3, .MPRX ;CHECK FOR END OF TABLE.
2463 012114 004767 006222      BLO    21$ ;BR IF MORE REGISTERS.
2464 012120 004767 007514      JSR    PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2465 012124 000013      JSR    PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2466 012126 000533      .WORD 13 ;ERROR TYPE CODE.
2467 012130 016763 167410 000002 23$: BR     WWPB4 ;EXIT AFTER ERROR.
2468 012136 001011      BIT    BITPT, 2(R3) ;CHECK THE MAP FOR THIS REGISTER.
2469 012140 036763 167402 000004      BNE    24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
2470 012146 001005      BIT    BITPT+2,4(R3) ;CHECK THE HI 64K.
2471 012150 004767 006162      BNE    24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
2472 012154 004767 007460      JSR    PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2473 012160 000014      JSR    PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2474 012162      .WORD 14 ;ERROR TYPE CODE.
2475 012162      MCV    R0, -(SP) ;:PUSH R0 ON STACK
2476 012164 010046      MOV    R2, R0 ;GET THE ADDRESS POINTER.
2477 012166 042700 003777      BIC    #3777, R0 ;CLEAR LOW ADDRESS BITS.
2478 012172 000300      SWAB  R0 ;SHIFT 6 PLACES RIGHT.
2479 012174 006300      ASL   R0
2480 012176 006300      ASL   R0
2481 012200 005767 166402      TST   MMAVA ;CHECK FOR MEM MGMT.
2482 012204 001404      BEQ   25$ ;BR IF NO MEM MGMT.
2483 012206 042700 177600      BIT   #177600, R0 ;CLEAR BANK BITS.
2484 012212 003700      ADD   @#KIPAR2, R0 ;ADD MEM MGMT OFFSET.
2485 012216 002700 100001      BIS   #BIT15-BIT0, R0 ;SET ERROR AND AE BIT IN CHECK WORD.
2486 012222 016367 000006      MOV   6(R3), RESRVD ;GET APPROPRIATE MASK.
2487 012230 046700 167262      BIC   RESRVD, R0 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2488 012234 046701 167256      BIC   RESRVD, R1 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2489 ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2490 012240 020001      CMP   R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2491 012242 001405      BEQ   67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2492 012244 004767 006066      JSR   PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2493 012250 004767 007364      JSR   PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2494 012254 000015      .WORD 15 ;ERROR TYPE CODE.
2495 012256      67$: CLR   @(R3) ;CLEAR REG INCLUDING ACTION ENABLE.
2497 012262 010346      MGV   R3, -(SP) ;:PUSH R3 ON STACK
2498 012264 062703 000010      ADD   #10, R3 ;UPDATE POINTER TO NEXT PARITY REG + MAP.
2499 012270 020367 167334      CMP   R3, .MPX ;CHECK FOR END OF TABLE.
2500 012274 101014      BHI   WWPB3 ;BR IF END OF TABLE REACHED.
2501 012276 032713 000001      RTT   #BIT0, (R3) ;CHECK IF NEXT REG EXISTS.
```

```
2502 012302 001370      BNE   26$ ;BR IF THIS PARITY REG DOESN'T EXIST.
2503 012304 017301 000000      MOV   @(R3), R1 ;SAVE AND CHECK FOR ERROR FLAG.
2504 012310 100365      BPL   26$ ;BR IF NO ERROR FLAG.
2505 012312 004767 006020      JSR   PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2506 012316 004767 007316      JSR   PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2507 012322 000016      .WORD 16 ;ERROR TYPE CODE.
2508 012324 000757      BR    26$ ;BR AFTER ERROR.
2509 012326 111204      WWPB3: MCVB (R2), R4 ;GET THE DATA FOR CHECKING.
2510 ;* READING THE DATA VIA DATA TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
2511 ;* ACTION ENABLE IS NOT SET IN CONTROLLING REG, SO NO TRAP SHOULD OCCURE.
2512 012330 111212      MOVB (R2), (R2) ;RESTORE RIGHT PARITY
2513 ;NOTE: THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
2514 ; WHICH DO ONLY DATOB TO DESTINATION OF MOV'S INSTRUCTIONS.
2515 012332 012603      MOV   (SP)+, R3 ;:POP STACK INTO R3
2516 012334 017301 000000      MOV   @(R3), R1 ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
2517 012340 006701 167152      BIC   RESRVD, R1 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2518 ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2519 012344 042700 000001      BIT   #AE, R0 ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
2520 012350 000001      CMP   R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2521 012352 001405      BEQ   65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2522 012354 004767 005756      JSR   PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2523 012360 004767 007254      JSR   PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2524 012364 000015      .WORD 15 ;ERROR TYPE CODE.
2525 012366      65$:
2526 012366 012773 000001 000000      MOV   #1, @(R3) ;CLEAR ALL BUT ACTION ENABLE.
2527 012374 010401      MOV   R4, R1 ;GET DATA READ FROM MEMORY FOR TESTING.
2528 012376 012600      MOV   (SP)+, R0 ;:POP STACK INTO R0
2529 012400 120001      CMPB  R0, R1 ;CHECK THE DATA.
2530 012402 001405      BEQ   67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2531 012404 004767 005732      JSR   PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2532 012410 004767 007224      JSR   PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2533 012414 000017      .WORD 17 ;ERROR TYPE CODE.
2534 012416      67$:
2535 012416      WWPB4: MCVB R0, (R2) ;RESTORE DATA.
2536 012420 167112      TSTB (R2) ;DO A DAT1 TO BE SURE RIGHT PARITY.
2537 012422 012700 000253      MOV   #253, R0 ;SET ODD PARITY DATA.
2538 012426 105167 167126      CCM3  ODFLG ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
2539 012432 100002      BPL   27$ ;BR IF DONE BOTH EVEN AND ODD.
2540 012434 000167 177316      JMP   WWPB2 ;LOOP BACK AND DO ODD(PARITY BIT CLR).
2541 012440 005202      INC   R2 ;MOVE POINTER TO NEXT MEMORY BYTE.
2542 012442 000502      WWPB5: BIT   R5, R2 ;CHECK FOR END OF BLOCK.
2543 012444 001402      BEQ   30$ ;BR IF END OF BLOCK FOUND.
2544 012446 000167 177240      JMP   WWPB1 ;LOOP BACK TO TEST NEXT BYTE.
2545 012452 004767 002526      JSR   PC, MMUP ;FIND NEXT BLOCK AND LOOP TO WWPBYT
2546 012456 004767 005054      JSR   PC, MAMF ;GO RESET PARITY REGISTERS.
```

```

2547 ;*****
2548 ;*TEST 20 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
2549 ;*****
2550 TST20:
2551 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
2552 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2553 RANTST: MOV PC, R3 ;GET CURRENT PROGRAM COUNTER.
2554 BIT #7777, R3 ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
2555 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2556 1$: MOV R2, -(SP) ;SAVE MEMORY POINTER.
2557 MCV R3, -(SP) ;SAVE "DATA" POINTER.
2558 2$: MOV (R3)+, (R2)+ ;MOV CODE INTO TEST MEMORY.
2559 BIT #7777, R3 ;CHECK FOR END OF "DATA TABLE"
2560 BNE 3$ ;BRANCH IF MORE
2561 SUB #10000, R3 ;RESET POINTER TO START OF "RANDOM DATA"
2562 3$: BIT R5, R2 ;CHECK FOR END OF BLOCK
2563 BNE 2$ ;BRANCH IF MORE
2564 MOV (SP)+, R3 ;RESET "DATA" POINTER.
2565 MOV (SP)+, R2 ;RESET MEMORY POINTER.
2566 4$: MOV (R3)+, R0 ;GET S/B DATA.
2567 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2568 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2569 BEQ 64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2570 64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2571 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2572 .WORD 20 ;ERROR TYPE CODE.
2573 65$:
2574 BIT #7777, R3 ;CHECK FOR END OF "DATA TABLE"
2575 BNE 5$ ;BR IF MORE.
2576 SUB #10000, R3 ;RESET POINTER TO TOP OF "DATA TABLE".
2577 5$:
2578 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2579 BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.
2580 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

```

2581 .SBTTL SECTION 3: INSTRUCTION EXECUTION TESTS.
2582 ;*****
2583 ;*TEST 21 EXECUTE DAT1, DAT0 THRU MEMORY.
2584 ;* EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
2585 ;* AN 'RTS RE' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
2586 ;* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2587 ;* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2588 ;*
2589 ;* MEMORY INSTRUCTION CONTENTS OF MEMORY LOCATION
2590 ;* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
2591 ;*
2592 ;* 1ST PASS / 40000 010412 000205
2593 ;* THRU TEST / 40002 000205 000205
2594 ;*
2595 ;* 2ND PASS / 40002 010412 000205
2596 ;* THRU TEST / 40004 000205 000205
2597 ;*
2598 ;* ETC., ETC., ETC.
2599 ;*
2600 ;* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2601 ;* R1 = DATA READ FROM MEMORY (WAS).
2602 ;* R2 = ADDRESS OF IUT/DATA.
2603 ;* R3 = INSTRUCTION UNDER TEST (IUT).
2604 ;* R4 = RTS R5 (CODE 205).
2605 ;* R5 = BLOCK BOUNDARY BIT MASK.
2606 ;*****
2607 TST21:
2608 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
2609 .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
2610 ; REQUIRED FOR THIS TEST.
2611 JMP TST22 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2612 ; AVAILABLE FOR TEST.
2613 DIDD: MOV #010412,R3 ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
2614 MOV #205, R4 ;GET 'RTS R5'
2615 MOV R4, R0 ;SET UP S/B DATA AFTER EXECUTION.
2616 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2617 1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2618 2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
2619 JSR R5, -(R2) ;GO EXECUTE THE IUT.
2620 MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2621 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2622 BEQ 64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2623 64$: JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
2624 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2625 .WORD 21 ;ERROR TYPE CODE.
2626 65$:
2627 MCV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
2628 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2629 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
2630 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

```

2631 ;*****
2632 ;TEST 22 EXECUTE DAT1, DAT05 (LOW BYTE) THRU MEMORY.
2633 ; EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
2634 ; AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
2635 ; CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2636 ; THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2637 ;
2638 ;
2639 ;          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
2640 ;          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
2641 ;
2642 ; 1ST PASS / 40000          110412          110605
2643 ; THRU TEST / 40002          000205          000205
2644 ;
2645 ; 2ND PASS / 40002          110412          110605
2646 ; THRU TEST / 40004          000205          000205
2647 ;
2648 ;          ETC., ETC., ETC.
2649 ;
2650 ; R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2651 ; R1 = DATA READ FROM MEMORY (WAS).
2652 ; R2 = ADDRESS OF IUT/DATA.
2653 ; R3 = INSTRUCTION UNDER TEST (IUT).
2654 ; R4 = RTS R5 (CODE 205).
2655 ; R5 = BLOCK BOUNDRY BIT MASK.
2656 ;*****
2657 TST22:
2658 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.
2659 LJR3 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
2660 ; REQUIRED FOR THIS TEST.
2661 JMP TST23 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2662 ; AVAILABLE FOR TEST.
2663 DIOB1: MOV #110412,R3 ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
2664 MCV #205, R4 ;GET 'RTS R5'
2665 MCV #110605,R0 ;SET UP S/B DATA AFTER EXECUTION.
2666 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2667 MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2668 MCV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
2669 JSR R5, -(R2) ;GO EXECUTE THE IUT.
2670 MCV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2671 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2672 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2673 JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
2674 JSR PC, SERRCR ;*** ERROR *** (GO TYPE A MESSAGE)
2675 ;.WORD 21 ;ERROR TYPE CODE.
2676 65$:
2677 MCV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
2678 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2679 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
2680 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

```

2680 ;*****
2681 ;TEST 23 EXECUTE DAT1, DAT05 (HIGH BYTE) THRU MEMORY.
2682 ; EXECUTES THE INSTRUCTION 'MOVB R3, -(R2)' THROUGHOUT MEMORY.
2683 ; AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
2684 ; CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2685 ; THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2686 ;
2687 ;
2688 ;          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
2689 ;          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
2690 ;
2691 ; 1ST PASS / 40000          110342          161342
2692 ; THRU TEST / 40002          000205          000205
2693 ;
2694 ; 2ND PASS / 40002          110342          161342
2695 ; THRU TEST / 40004          000205          000205
2696 ;
2697 ;          ETC., ETC., ETC.
2698 ;
2699 ; R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2700 ; R1 = DATA READ FROM MEMORY (WAS).
2701 ; R2 = ADDRESS OF IUT/DATA.
2702 ; R3 = INSTRUCTION UNDER TEST (IUT).
2703 ; R4 = RTS R5 (CODE 205).
2704 ; R5 = BLOCK BOUNDRY BIT MASK.
2705 ;*****
2706 TST23:
2707 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.
2708 ;.WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
2709 ; REQUIRED FOR THIS TEST.
2710 JMP TST24 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2711 ; AVAILABLE FOR TEST.
2712 DIOB3H: MOV #110342,R3 ;GET 'MOVB R3, -(R2)' INSTRUCTION (IUT).
2713 MCV #205, R4 ;GET 'RTS R5'
2714 MOV #161342,R0 ;SET UP S/B DATA AFTER EXECUTION.
2715 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2716 MCV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2717 MCV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
2718 JSR R5, -(R2) ;GO EXECUTE THE IUT.
2719 DEC R2 ;ADJUST R2 TO POINT TO MAUT.
2720 MCV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2721 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2722 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2723 JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
2724 JSR PC, SERRCR ;*** ERROR *** (GO TYPE A MESSAGE)
2725 ;.WORD 21 ;ERROR TYPE CODE.
2726 65$:
2727 MCV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
2728 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2729 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
2730 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

```
2730
2731 :*****
2732 :*TEST 24 EXECUTE DATI, DATIP, DATO THRU MEMORY.
2733 :* EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
2734 :* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
2735 :* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2736 :* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2737 :*
2738 :*          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
2739 :*          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
2740 :*
2741 :* 1ST PASS / 40000          005412          172366
2742 :* THRU TEST / 40002          000205          000205
2743 :*
2744 :* 2ND PASS / 40002          005412          172366
2745 :* THRU TEST / 40004          000205          000205
2746 :*
2747 :*          ETC., ETC., ETC.
2748 :*
2749 :* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2750 :* R1 = DATA READ FROM MEMORY (WAS).
2751 :* R2 = ADDRESS OF IUT/DATA.
2752 :* R3 = INSTRUCTION UNDER TEST (IUT).
2753 :* R4 = RTS R5 (CODE 205).
2754 :* R5 = BLOCK BOUNDRY BIT MASK.
2755 :*****
2756 :*TST24:
2757 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.
2758 .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
2759 JMP TST25 ;REQUIRED FOR THIS TEST.
2760 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2761 ;AVAILABLE FOR TEST.
2762 DIPDO: MCV #005412,R3 ;GET 'NEG (R2)' INSTRUCTION (IUT).
2763 MCV #205, R4 ;GET 'RTS R5'
2764 MCV #172366,R0 ;SET UP S/B DATA AFTER EXECUTION.
2765 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2766 MCV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2767 MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
2768 JSR R5, -(R2) ;GO EXECUTE THE IUT.
2769 MCV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2770 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2771 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2772 JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
2773 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2774 .WORD 21 ;ERROR TYPE CODE.
2775 65$: MCV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
2776 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2777 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
2778 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804 013056
2805 013056 004567 005546
2806 013062 000003
2807
2808 013064 000167 000060
2809
2810 013070 012703 005412
2811 013074 012704 000205
2812 013100 012700 172366
2813 013104 004467 001316
2814 013110 010322
2815 013112 010412
2816 013114 004542
2817 013116 012201
2818 013120 020001
2819 013122 001405
2820 013124 004767 005232
2821 013130 004767 006504
2822 013134 000021
2823 013136
2824 013136 010322
2825 013140 030502
2826 013142 001363
2827 013144 004767 002034
```

```
2779
2780 :*****
2781 :*TEST 25 EXECUTE DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.
2782 :* EXECUTES THE INSTRUCTION 'BICB (R2)+, -(R2)' THROUGHOUT MEMORY.
2783 :* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BICB' INSTRUCTION TO RETURN
2784 :* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2785 :* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2786 :*
2787 :*          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
2788 :*          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
2789 :*
2790 :* 1ST PASS / 40000          142242          142000
2791 :* THRU TEST / 40002          000205          000205
2792 :*
2793 :* 2ND PASS / 40002          142242          142000
2794 :* THRU TEST / 40004          000205          000205
2795 :*
2796 :*          ETC., ETC., ETC.
2797 :*
2798 :* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2799 :* R1 = DATA READ FROM MEMORY (WAS).
2800 :* R2 = ADDRESS OF IUT/DATA.
2801 :* R3 = INSTRUCTION UNDER TEST (IUT).
2802 :* R4 = RTS R5 (CODE 205).
2803 :* R5 = BLOCK BOUNDRY BIT MASK.
2804 :*****
2805 :*TST25:
2806 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.
2807 .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
2808 JMP TST26 ;REQUIRED FOR THIS TEST.
2809 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2810 ;AVAILABLE FOR TEST.
2811 DPDBL: MCV #142242,R3 ;GET 'BICB (R2)+, -(R2)' INSTRUCTION (IUT).
2812 MCV #205, R4 ;GET 'RTS R5'
2813 MCV #142000,R0 ;SET UP S/B DATA AFTER EXECUTION.
2814 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2815 MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2816 MCV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
2817 JSR R5, -(R2) ;GO EXECUTE THE IUT.
2818 MCV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2819 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2820 BEQ 64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2821 JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
2822 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2823 .WORD 21 ;ERROR TYPE CODE.
2824 65$: MCV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
2825 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2826 BNE 2$ ;BRANCH IF MORE IN CURRENT BLCK.
2827 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854 013150
2855 013150 004567 005454
2856 013154 000003
2857
2858 013156 000167 000060
2859
2860 013162 012703 142242
2861 013166 012704 000205
2862 013172 012700 142000
2863 013176 004467 001224
2864 013202 010322
2865 013204 010412
2866 013206 004542
2867 013210 012201
2868 013212 020001
2869 013214 001405
2870 013216 004767 005140
2871 013222 004767 006412
2872 013226 000001
2873 013230
2874 013230 010322
2875 013232 030502
2876 013234 001363
2877 013236 004767 001742
```



```
2828 .....  
2829 :*TEST 26 EXECUTE DAT1, DAT1, DAT1P, DAT0B (HIGH BYTE) THRU MEMORY.  
2830 :* EXECUTES THE INSTRUCTION 'B1SB (R2)+,(R2)' THROUGHOUT MEMORY.  
2831 :* AN 'RTS R5' (CODE 2C5) IS PLACED AFTER THE 'B1SB' INSTRUCTION TO RETURN  
2832 :* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.  
2833 :* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:  
2834 :*  
2835 :*  
2836 :* MEMRY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION  
2837 :* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION  
2838 :*  
2839 :* 1ST PASS / 40000 152212 157212  
2840 :* THRU TEST / 40002 000205 000205  
2841 :*  
2842 :* 2ND PASS / 40002 152212 157212  
2843 :* THRU TEST / 40004 000205 000205  
2844 :*  
2845 :* ETC., ETC., ETC.  
2846 :*  
2847 :* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).  
2848 :* R1 = DATA READ FROM MEMORY (WAS).  
2849 :* R2 = ADDRESS OF IUT/DATA.  
2850 :* R3 = INSTRUCTION UNDER TEST (IUT).  
2851 :* R4 = RTS R5 (CODE 205).  
2852 :* R5 = BLOCK BOUNDARY BIT MASK.  
2853 :*  
TCT26:  
2853 013242 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.  
2854 013242 .WJRD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS  
2855 013246 000003 ;REQUIRED FOR THIS TEST.  
2856 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK  
2857 013250 000167 000062 ;AVAILABLE FOR TEST.  
2858 ;  
2859 013254 012703 152212 DPDBH: MOV #152212,R3 ;GET 'B1SB (R2)+,(R2)' INSTRUCTION (IUT).  
2860 013260 012704 000205 #205, R4 ;GET 'RTS R5'  
2861 013264 012700 157212 MOV #157212,R0 ;SET UP S,B DATA AFTER EXECUTION.  
2862 013270 004467 001132 JSR R4, IN:TM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
2863 013274 010322 15: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.  
2864 013276 010412 25: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.  
2865 013300 004542 JSR R5, -(R2) ;GO EXECUTE THE IUT.  
2866 013302 005302 DEC R2 ;RESET R2 TO POINT TO IUT.  
2867 013304 012201 MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADDR UNDER TEST.  
2868 013306 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ  
2869 013310 001405 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2870 013312 004767 005044 64$: JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.  
2871 013316 004767 006316 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2872 013322 000021 .WORD 21 ;ERROR TYPE CODE.  
2873 013324 65$:  
2874 013324 010322 MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.  
2875 013326 005002 BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
2876 013330 001362 BNE 2$ ;BRANCH IF MDRE IN CURRENT BLOCK.  
2877 013332 004767 001646 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
2878 .....  
2879 :*SBITL SECTION 4:MOS TESTS  
2880 :*  
2881 :*TEST 27 MARCHING 1'S AND 0'S.  
2882 :* THIS TEST IS DESIGNED TO STRESS MOS MEMORIES.  
2883 :* STARTING AT THE BOTTOM ADDRESS AND ADDRESSING UPWARDS A 4K BANK IS  
2884 :* WRITTEN WITH 000377, THEN STARTING AT THE TOP ADDRESS OF THE BANK THE  
2885 :* 000377 IS READ, THE BYTES ARE SWAPPED TO 177400 AND THE LOCATION  
2886 :* REREAD TO CONFIRM THE WRITE. THIS IS REPEATED FOR EVERY LOCATION  
2887 :* ADDRESSED DOWNWARD UNTIL THE BOTTOM IS REACHED. STARTING AT THE  
2888 :* BOTTOM EACH LOCATION IS READ FOR 177400, THE BYTES ARE SWAPPED TO  
2889 :* 000377 AND REREAD TO CONFIRM THE WRITE UNTIL THE TOP ADDRESS OF THE  
2890 :* BANK IS REACHED. AGAIN STARTING AT THE BOTTOM EACH LOCATION IS READ  
2891 :* FOR 000377, THE BYTES SWAPPED TO 177400 AND THE LOCATION REREAD TO  
2892 :* CONFIRM THE WRITE. LASTLY STARTING FROM THE TOP AND ADDRESSING DOWN-  
2893 :* WORD EACH LOCATION IS REREAD, THE BYTES SWAPPED TO 000377 AND THE  
2894 :* LOCATION IS REREAD TO CONFIRM THE WRITE. THIS IS REPEATED FOR EVERY  
2895 :* 4K BANK UNDER TEST.  
2896 :*  
2897 :* R0=DATA WRITTEN INTO MEMORY(SHOULD BE)  
2898 :* R1=DATA READ FROM MEMORY(WAS)  
2899 :* R2=VIRTUAL ADDRESS  
2900 :* R3=TIMES THROUGH COUNTER  
2901 :* R4=NOT USED  
2902 :* R5=BLOCK BOUNDARY BIT MASK.  
2903 :*  
TST27:  
2904 013336 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.  
2905 013342 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.  
2906 013344 004467 001056 JSR R4, IN:TM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
2907 013350 012627 166240 MOV R2,TEMP ;SAVE BANK STARTING ADDRESS  
2908 013354 005003 15: CLR R3 ;CLEAR PASS COUNTER  
2909 013356 012700 000377 MOV #000377,R0 ;SETUP TO WRITE PATTERN  
2910 013362 010022 25: MOV R0,(R2)+ ;WRITE PATTERN  
2911 013364 030502 BIT R5,R2 ;END OF 4K?  
2912 013366 001375 BNE 2$ ;CONTINUE WRITING IF NO.  
2913 013370 014201 35: MOV -(R2),R1 ;GET DATA WRITEN  
2914 013372 020001 CVP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2915 013374 001405 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2916 013376 004767 004764 64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2917 013402 004767 006232 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2918 013406 000010 .WORD 10 ;ERROR TYPE CODE.  
2919 013410 65$:  
2920 013410 000300 45: SWAB R0 ;SWAP BYTES OF DATA  
2921 013412 010012 MOV R0,(R2) ;WRITE SWAPPED WORD  
2922 013414 011201 MOV (R2),R1 ;GET DATA WRITEN  
2923 013416 020001 CVP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
2924 013420 001405 BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
2925 013422 004767 004740 66$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
2926 013426 004767 006206 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
2927 013432 000010 .WORD 10 ;ERROR TYPE CODE.  
2928 013434 67$:  
2929 013434 000300 SWAP R0 ;PUT DATA BACK TO ORIGINAL  
2930 013436 005703 TST R3 ;IF ON PASS 0 OR PASS 3  
2931 013440 001403 BEQ 5$ ;WE ARE ADDRESSING DOWN  
2932 013442 030327 CMP R3,#3 ;IF ON PASS 1 OR 2 GO TO  
2933 013446 001010 BNE 6$ ;UPWARD
```

```
2934 013450 030502 5$: BIT R5,R2 ;DDNE A PASS?
2935 013452 0C1346 BNE 3$ ;IF NO CONTINUE
2936 013454 0C5203 INC R3 ;IF YES INCREMENT PASS COUNTER
2937 013456 022703 CMP #4,R3 ;ARE WE DONE ALL PASSES FOR THIS 4K?
2938 013462 001427 BEQ 9$ ;IF YES BRANCH
2939 013464 000300 SWAB R0 ;ELSE SET UP NEW READ WORD
2940 013466 000404 BR 7$ ;GO TO START OF ADDRESS UP
2941 013470 062702 6$: ADD #2,R2 ;UPDATE TO NEXT ADDRESS
2942 013474 030502 BIT R5,R2 ;DDNE A PASS
2943 013476 0C1411 BEQ 8$ ;IF YES BRANCH
2944 013500 011201 7$: MOV (R2),R1 ;GET DATA WRITTEN
2945 013502 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2946 013504 001405 BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2947 013506 004767 69$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2948 013512 004767 006122 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2949 013516 000010 .WORD 10 ;ERROR TYPE CODE.
2950 013520 69$: BR 4$
2951 013520 000733 BR 4$
2952 013522 0C5203 8$: INC R3 ;INCREMENT PASS COUNTER
2953 013524 000300 SWAB R0 ;SET UP NEW READ WORD
2954 013526 020327 CMP R3,#2 ;ADDRESSING UP?
2955 013532 001316 BNE 3$ ;IF NO GO TO DOWN SEQUENCE
2956 013534 018702 166054 MOV TEMP,R2 ;IF YES RESET ADDRESS TO START
2957 013540 000757 BR 7$ ;GO TO UP SEQUENCE
2958 013542 004467 000660 9$: JSR R4,INITMM ;INITIALIZE MEMORY ADDRESS POINTERS
2959 013546 004767 001432 JSR PC,MMUP ;UPDATE TO NEW BANK IF EXISTS
2960
2961 ;*****
2962 ;*TEST 30 WRITE CHECKERBOARD STARTING WITH '125252' DATA.
2963 ;* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY,STALL
2964 ;* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
2965 ;* DETERIORATE BETWEEN REFRESH CYCLES.
2966 ;*
2967 ;* R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
2968 ;* R1=DATA READ FROM MEMORY(WAS)
2969 ;* R2=VIRTUAL ADDRESS
2970 ;* R3=SMALL LOOP COUNTER FOR STALL
2971 ;* R4=NUMBER OF TIMES SMALL LOOP DONE
2972 ;* R5=BLOCK BOUNDARY BIT MASK.
2973 ;*****
2974 013552 TST30:
2975 013552 004567 005052 JSR R5, SSCOPE ;GO TO SCOPE ROUTINE.
2976 013556 000000 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2977 013560 004477 000642 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2978 013564 012700 125252 MOV #125252,R0 ;SETUP DATA PATTERN
2979 013570 010022 1$: MOV R0 ,(R2)+ ;WRITE A WORD
2980 013572 005100 CCM R0 ;COMPLEMENT DATA
2981 013574 030502 BTT R5, R2 ;CHECK FOR END OF A BLOCK.
2982 013576 001374 BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
2983 013600 004767 001400 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
2984 013604 005003 CLR R3 ;SET UP COUNTER FOR STALL
2985 013606 012764 000046 MOV #46, R4 ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
2986 013612 005303 2$: DEC R3
2987 013614 001376 BNE 2$
2988 013616 005304 DEC R4
2989 013620 001374 BNE 2$
```

```
2990 013622 004467 000600 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2991 013626 012700 125252 MOV #125252,R0 ;INIT DATA FOR CHECKING
2992 013632 3$:
2993 013632 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2994 013634 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2995 013636 001405 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2996 013640 004767 004522 64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2997 013644 004767 005770 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2998 013650 000006 .WORD 6 ;ERROR TYPE CODE.
2999 013652 65$:
3000 013652 005100 CCM R0
3001 013654 030502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
3002 013656 001365 BNE 3$ ;BRANCH IF MORE IN CURRENT BLOCK.
3003 013660 004767 001320 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
3004
3005 ;*****
3006 ;*TEST 31 WRITE CHECKERBOARD STARTING WITH 052525 DATA
3007 ;*****
3008 013664 TST31:
3009 013670 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3010 013672 004467 000530 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3011 013676 012700 052525 MOV #052525,R0 ;SETUP DATA PATTERN
3012 013702 010022 1$: MOV R0 ,(R2)+ ;WRITE A WORD
3013 013704 005100 CCM R0
3014 013706 030502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
3015 013710 001374 BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
3016 013712 004767 001266 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
3017 013716 005003 CLR R3 ;SET COUNTER FOR LOOP
3018 013720 012704 000046 MOV #46, R4 ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
3019 013724 005303 2$: DEC R3
3020 013726 01376 BNE 2$
3021 013730 005304 DEC R4
3022 013732 001374 BNE 2$
3023 013734 004467 000466 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3024 013740 012700 052525 MOV #052525,R0 ;INIT PATTERN FOR CHECKING
3025 013744 3$:
3026 013744 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
3027 013746 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
3028 013750 001405 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
3029 013752 004767 004410 64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
3030 013756 004767 005656 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3031 013762 000006 .WORD 6 ;ERROR TYPE CODE.
3032 013764 65$:
3033 013764 005100 CCM R0
3034 013766 030502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
3035 013770 001365 BNE 3$ ;BRANCH IF MORE IN CURRENT BLOCK.
3036 013772 004767 001206 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```

3037 .SBTTL DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.
3038 DONE: JSR R5, \$SCOPE ;GO TO SCOPE ROUTINE.
3039 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3040 014002 000000 TST32: CLR \$TIMES ;RESET ITERATION COUNTER FOR RESTARTING TEST.
3041 014004 005067 165160 CLR STSTN ;RESET TEST NUMBER.
3042 014010 105067 165066 BIT PRGMAP, SAVTST ;CHECK IF PPROGRAM IS IN TEST AREA.
3043 014014 036767 164562 165512 1\$: BNE 2S ;BR IF IT PRDGM IN MEM TO BE TESTED.
3044 014022 001004 BIT PRGMAP+2,SAVTST+2 ;CHECK HI 64K
3045 014024 036767 164554 165504 BEQ \$EOP ;BR IF PROG NOT IN MEM TO BE TESTED.
3046 014032 061435 #SW07, @SWR ;CHECK FOR INHIBIT RELOCATION SWITCH.
3047 014034 032777 000200 165076 2\$: BNC \$EOP ;SKIP RELOCATION IF SWITCH SET.
3048 014042 001031 CMP #3, PRGMAP ;CHECK IF PROGRAM IN FIRST BK.
3049 014044 022767 000003 164530 BNE 4S ;BR IF NOT IN FIRST BK.
3050 014352 001013 CHP @#42,@#46 ;CHECK FOR ACT11
3051 014054 023737 000042 000046 BEQ 6S ;BR IF ACT11.
3052 014062 001416 TSTB @#SENV ;CHECK FOR APT11
3053 014064 105737 001224 BNE 6S ;IF APT11 DO NOT RELOCATE
3054 014070 001013 JSR PC, RELTOP ;RELOCATE PROGRAM TO TOP OF MEMORY.
3055 JSR PC, RELTOP ;LOOP BACK AND RUN ALL TESTS AGAIN.
3056 014072 004767 002362 3\$: JMP START1
3057 014076 000167 172002 JSR PC, RELO ;RELOCATE PROGRAM BACK TO FIRST BK.
3058 TST @#42 ;TEST FOR
3059 014102 004767 002754 4\$: JSR PC, RELO ;RELOCATE PROGRAM BACK TO FIRST BK.
3060 014106 005737 000042 TST @#42 ;TEST FOR
3061 014110 001402 BEQ 6S ;IF NOT R. NG UNDER MON. DONT
3062 014114 004767 003150 5\$: JSR PC, RESLDR ;RESTORE L ADERS.
3063 014120 JSR PC, RESLDR ;RESTORE L ADERS.
3064 014120 004567 007366 JSR R5, \$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3065 014124 001201 .WORD \$CRLF ;ADDRESS OF MESSAGE TO BE TYPED

3066 ;*****
3067 .SBTTL END OF PASS ROUTINE
3068 ;*INCREMENT THE PASS NUMBER (\$PASS)
3069 ;*TYPE "END PASS #XXXX" (WHERE XXXX IS A DECIMAL NUMBER)
3070 ;*IF THERES A MONITOR GO TO IT
3071 ;*IF THERES ISN'T JUMP TO START1
3072 ;*IF THERES ISN'T JUMP TO START1
3073
3074
3075 014126 \$EOP:
3076 014126 000240 NOP
3077 014130 005067 165034 CLR \$TIMES ;ZERO THE NUMBER OF ITERATIONS
3078 014134 005267 165052 INC \$PASS ;INCREMENT THE PASS NUMBER
3079 014140 002767 100000 165044 BIC #100000,\$PASS ;DON'T ALLOW A NEG. NUMBER
3080 014146 005327 DEC (PC)+ ;LOOP?
3081 014150 000001 \$EOPCT: .WORD 1
3082 014152 003040 BGT \$DOAGN ;YES
3083 014154 012737 MOV (PC)+,@(PC)+ ;RESTORE COUNTER
3084 014156 000001 \$ENDCT: .WORD 1
3085 014160 014150 \$EOPCT
3086 014162 004567 007324 JSR R5, \$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3087 014166 014260 .WORD \$ENDMG ;ADDRESS OF MESSAGE TO BE TYPED
3088 014170 016746 165016 MOV \$PASS,-(SP) ;SAVE \$PASS FOR TYPEOUT
3089 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$TYPDS ROUTINE
3090 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3091 014174 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3092 014200 004767 010226 JSR PC, \$TYPDS ;GO TO THE SUBROUTINE
3093 014204 004567 007302 JSR R5, \$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3094 014210 014275 .WORD \$ENULL ;ADDRESS OF MESSAGE TO BE TYPED
3095 014212 \$GET42:
3096 014212 016700 163624 MOV 42, R0 ;GET MONITOR ADDRESS
3097 014216 001416 BEQ \$DOAGN ;BRANCH IF NO MONITOR
3098 014220 000005 RESET ;CLEAR THE WORLD
3099 014222 004710 \$ENDAD: JSR PC,(R0) ;GO TO MONITOR
3100 014224 000240 NOP ;SAVE R0DM
3101 014226 000240 NOP ;FOR
3102 014230 000240 NOP ;ACT11
3103 014232 023737 000042 000046 CMP @#42,@#46 ;ARE WE UNDER ACT11 OR XXDP
3104 014240 001405 BEQ \$DOAGN ;IF ACT11 THEN RSTART
3105 014242 105737 001224 TSTB @#SENV ;CHECK FOR APT11
3106 014246 001002 BNE \$DOAGN ;IF APT11 THEN RESTART
3107 014250 004767 003074 JSR PC, SAVLDR ;IF XXDP FIRST SAVE MONITOR
3108 014254 \$DOAGN:
3109 014254 JMP START1 ;RETURN*****
3110 014254 000167 171624 \$ENDMG: .ASCIZ <15><12>/END PASS #/
3111 014260 005015 047105 020104
3112 014266 0040520 051523 02.140
3113 014274 000
3114 014275 377 377 000 \$ENULL: .BYTE -1,-1,0 ;NULL CHARACTER STRING
3115 .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
3116 .SBTTL MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
3117 ;*****
3118 ;* SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
3119 ;* THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
3120 ;* THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
3121 ;* THE DEV.ICE ADDRESS AREA IS POINTED TO BY PAR 7.

```
3122 ;* PARS 4, 5, AND 6 ARE UNUSED.
3123 ;*****
3124 014300 MMINIT:
3125 014300 012737 077406 172300 MOV #200-1+400+UP+RW,@#KIPDR0 ;SET KIPDR0 = RW UP 200 BLOCKS
3126 014306 012737 077406 172302 MOV #200-1+400+UP+RW,@#KIPDR1 ;SET KIPDR1 = RW UP 200 BLOCKS
3127 014314 012737 077406 172304 MOV #200-1+400+UP+RW,@#KIPDR2 ;SET KIPDR2 = RW UP 200 BLOCKS
3128 014322 012737 077406 172306 MOV #200-1+400+UP+RW,@#KIPDR3 ;SET KIPDR3 = RW UP 200 BLOCKS
3129 014330 005037 172310 CLR @#KIPDR4
3130 014334 005037 172312 CLR @#KIPDR5
3131 014340 005037 172314 CLR @#KIPDR6
3132 014344 012737 077406 172316 MOV #200-1+400+UP+RW,@#KIPDR7 ;SET KIPDR7 = RW UP 200 BLOCKS
3133 014352 005037 172340 CLR @#KIPAR0 ;MAP PAR0 INTO BANK0
3134 014355 012737 000200 172342 MOV #200, @#KIPAR1 ;MAP PAR1 INTO BANK1
3135 014364 005037 172344 CLR @#KIPAR2 ;MAP PAR2 INTO BANK0
3136 014370 005037 172346 CLR @#KIPAR3
3137 014374 005037 172350 CLR @#KIPAR4
3138 014400 005037 172352 CLR @#KIPAR5
3139 014404 005037 172354 CLR @#KIPAR6
3140 014410 012737 007600 172356 MOV #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
3141 014416 012737 000001 177572 MOV #1, @#SRO ;ENABLE MEMORY MANAGEMENT
3142 014424 000207 RTS PC ;RETURN
3143
3144
3145 ;*****
3146 ;* MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
3147 ;*****
3148 014426 012767 000001 165110 INITMM: MOV #BIT0, BITPT ;SET POINTER TO BANK0
3149 014434 005007 165106 CLR BITPT+2 ;CLEAR HI 64K BANK POINTERS
3150 014440 005002 CLR R2 ;SET ADDRESS POINTER TO 0
3151 014442 016705 165140 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3152 014446 005767 164134 TST MMAVA ;CHECK FOR MEM MGMT AVAILABLE
3153 014452 001514 BEQ 10$ ;BRANCH IF NO MEM MGMT
3154 014454 005037 172344 CLR @#KIPAR2 ;SET UP 3RD PAR TO BANK0
3155 014460 012702 040000 MOV #40000, R2 ;RESET VIRTUAL ADR POINTER
3156 014464 036767 165054 165036 1$: BIT BITPT, T$TMAP ;CHECK IF THIS BANK TO BE TESTED
3157 014472 001015 BNE 2$ ;BRANCH IF MATCH
3158 014474 036767 165046 165030 BIT BITPT+2, T$TMAP+2 ;CHECK IN HI MAP
3159 014502 001011 BNE 2$ ;BRANCH IF MATCH
3160 014504 002737 000200 172344 ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3161 014512 006367 165026 ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
3162 014516 006167 165024 ROL BITPT+2 ;...HI POINTER.
3163 014522 100360 BPL 1$ ;BR IF MORE.
3164 014524 000000 HALT ;FATAL ERROR!!! NO 4K BANK FOUND?
3165 014526 036767 165012 165046 2$: BIT BITPT, LADMAP ;CHECK IF LAST BANK.
3166 014534 001004 BNE 3$ ;BR IF LAST BANK.
3167 014536 036767 165004 165040 BIT BITPT+2, LADMAP+2 ;CHECK IF LAST BANK.
3168 014544 001405 BEQ 4$ ;BR IF NOT LAST BANK.
3169 014546 016705 165026 MTF LADMSK, R5 ;SET MASK TO FIND LAST ADR.
3170 014552 002767 020000 165016 BIC #20000, T$PLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
3171 014560 013737 172344 172346 4$: MOV @#KIPAR2, @#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3172 014566 016767 164752 164754 BITPT, T$MPPT ;COPY BITPT...LO 64K.
3173 014574 016767 164746 164750 MOV BITPT+2, T$MPPT+2 ;...HI 64K.
3174 014602 032705 020000 BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
3175 014606 001505 BEQ 21$ ;BRANCH IF NOT 8K.
3176 014610 002737 000200 172346 5$: ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
3177 014616 006367 164726 ASL T$MPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
```

```
3178 014622 006167 164724 ;...HI POINTER.
3179 014626 100473 BMI 20$ ;BR IF NO MORE.
3180 014630 036767 164714 164672 BIT T$MPPT, T$TMAP ;CHECK IF BANK TO BE TESTED.
3181 014636 001004 BNE 6$ ;BRANCH IF A MATCH.
3182 014640 036767 164706 164564 BIT T$MPPT+2, T$TMAP+2 ;CHECK FOR HI 64K BANKS.
3183 014646 001760 BEQ 5$ ;BRANCH IF NO MEMORY.
3184 014650 036767 164674 164724 6$: BIT T$MPPT, LADMAP ;CHECK IF LAST BANK.
3185 014656 001004 BNE 7$ ;BRANCH IF A MATCH
3186 014660 036767 164666 164716 BIT T$MPPT+2, LADMAP+2 ;CHECK HI 64K
3187 014666 001455 BEQ 21$ ;BR IF NOT LAST BANK.
3188 014670 016705 164704 7$: MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3189 014674 002767 020000 164674 BIT #20000, T$PLAD ;MAKE SURE LAST ADDRESS IS IN BANK 3.
3190 014702 000447 BR 21$ ;BR TO FINISH UP.
3191
3192 014704 036767 164634 164616 10$: BIT BITPT, T$TMAP ;CHECK IF THIS BANK TO BE TESTED.
3193 014712 001006 BNE 11$ ;BR IF MATCH.
3194 014714 002702 020000 ADD #20000, R2 ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3195 014720 106367 164620 BITPT ;UPDATE BANK POINTER TO NEXT BANK.
3196 014724 100367 BPL 10$ ;BR IF MORE BANKS.
3197 014726 000000 HALT ;FATAL ERROR!!! NO 4K BANK FOUND?
3198 014730 016767 164610 164612 11$: BITPT, T$MPPT ;COPY BANK POINTER.
3199 014736 036767 164602 164636 BIT BITPT, LADMAP ;CHECK IF LAST BANK.
3200 014744 001021 BNE 12$ ;BR IF LAST BANK.
3201 014746 032705 020000 BIT #BIT13, R5 ;CHECK FOR 8K BLOCK SIZE.
3202 014752 001423 BEQ 21$ ;BRANCH IF SMALLER BLOCK SIZE.
3203 014754 106367 164570 ASLB T$MPPT ;POINT TO NEXT BANK.
3204 014760 100416 BMI 20$ ;BRANCH IF OVERFLOW.
3205 014762 036767 164562 164540 BNE T$MPPT, T$TMAP ;CHECK IF BANK TO BE TESTED.
3206 014770 001412 BEQ 20$ ;BRANCH IF NOT TO BE TESTED.
3207 014772 112767 000011 164557 MOVB #11, FLAG8K ;SET 8K BLOCK SIZE FLAG.
3208 015000 016767 164544 164574 BIT T$MPPT, LADMAP ;CHECK FOR LAST BANK.
3209 015006 001403 BEQ 20$ ;BR IF NOT LAST BANK.
3210 015010 016705 164564 12$: MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3211 015014 000402 BR 21$ ;SKIP MASK RESET.
3212 015016 012705 017777 20$: MOV #MASK4K, R5 ;RESET MASK TO 4K BLOCK SIZE.
3213 015022 006767 164516 164520 21$: BIS BITPT, T$MPPT ;SET T$MPPT FOR FLAGGING LAST BANK.
3214 015030 006767 164512 164514 BIS BITPT+2, T$MPPT+2
3215 015036 036767 164502 164524 BIT BITPT, FADMAP ;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
3216 015044 001004 BNE 22$ ;BR IF FIRST BANK.
3217 015046 036767 164474 164516 BIT BITPT+2, FADMAP+2 ;CHECK HI 64K.
3218 015054 001450 BEQ INITEX ;BR IF NOT FIRST BANK.
3219 015056 016702 164502 22$: MOV T$MPPT, R2 ;RESET ADDRESS POINTER TO FIRST ADR.
3220 015062 000445 BR INITEX
3221
3222 015064 016705 164516 INITON: MOV BLKMSK, R5 ;RESET R5 TO CURRENT BLOCK MASK.
3223 015070 005002 CLR R2 ;INIT ADDRESS POINTER.
3224 015072 005767 163510 TST MMAVA ;CHECK FOR MEM MGMT
3225 015076 001411 BEQ 31$ ;BRANCH IF NO MEM MGMT
3226 015100 012767 100000 164440 MOV #BIT15, BITPT+2 ;SET POINTER TO TOP BIT
3227 015106 005067 164432 CLR BITPT
3228 015112 012737 007600 172344 MOV #7600, @#KIPAR2 ;SET PAR TO TOP OF MEM
3229 015120 000443 BR 32$ ;BRANCH TO COMMON AREA
3230
3231 015122 012767 000443 164414 31$: MOV #BIT8, BITPT ;SET UP BANK POINTER
3232 015130 012767 015152 164416 32$: MOV #33$, M$MORE ;SET "MMDOWN" EXIT ADDRESS.
3233 015136 006767 163436 164410 ADD RELOC, M$MORE ;ADD OFFSET
```

```
3234 015144 004767 000524 JSR PC, MMDDOWN ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3235 015150 000000 HALT ;FATAL ERROR!!! NO MEM INDICATED IN MEM MAP ABOVE BK!
3236 015152 036767 164366 164422 33$: BIT BITPT, LADMAP ;CHECK FOR NON BOUNDARY LAST ADDR.
3237 015160 001011 BNE 345 ;BR IF LAST BANK FLAG FOUND.
3238 015162 036767 164360 164414 BIT BITPT+2, LADMAP+2 ;CHECK FOR NON BOUNDARY LAST ADDR.
3239 015170 001402 BEQ INITEX ;BR IF NO LAD FLG FOUND.
3240 015172 016702 164376 34$: MOV LSTADR, R2 ;SET UP R2.
3241 015176 010467 164352 INITEX: MC ' R4, MMORE ;PUT RETURN PC INTO "MMORE"
3242 015202 000204 RTS R4 ;RETURN
3243
3244 ;*****
3245 ;* COMMON UPWARDS ADDRESSING ROUTINE
3246 ;* FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
3247 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
3248 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3249 ;*****
3250 015204 036767 164340 164370 MMUP: BIT TMPPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3251 015212 001122 BNE 10$ ;BR IF LAST BANK.
3252 015214 036767 164332 164362 BIT TMPPT+2, LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3253 015222 001116 BNE 10$ ;BR IF LAST BANK.
3254 015224 016705 164356 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3255 015230 005767 163752 TST MMAPVA ;CHECK FOR MEM MGMT AVAILABLE
3256 015234 001515 BEQ 20$ ;BRANCH IF NO MEM MGMT
3257 015236 012702 040000 MOV #40000, R2 ;RESET VIRTUAL ADR POINTER
3258 015242 062737 000200 172344 15: ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3259 015250 006367 164270 ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
3260 015254 006167 164266 ROL BITPT+2 ;...HI POINTER.
3261 015260 100577 BMI 32$ ;BR IF ALL DONE.
3262 015262 036767 164256 164240 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS
3263 015270 001004 BNE 2$ ;BRANCH IF MATCH
3264 015272 036767 164250 164232 BIT BITPT+2, TSTMAP+2 ;CHECK IN HI MAP
3265 015280 001760 BEQ 1$ ;BRANCH IF NO MATCH
3266 015302 036767 164236 164272 25: BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3267 015310 001004 BNE 3$ ;BRANCH IF LAST BANK FLAG.
3268 015312 036767 164230 164264 BIT BITPT+2, LADMAP+2 ;CHECK IF LAST BANK FLAG.
3269 015320 001405 BEQ 4$ ;BR IF NOT LAST BANK.
3270 015322 016705 164252 35: MOV LADMSK, R5 ;RESET MASK.
3271 015326 042767 020000 164242 BIC #20000, TMAPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3272 015334 016767 164204 164206 45: MOV BITPT, TMPPT ;COPY BITPT...LO 64K.
3273 015342 016767 164200 164202 MOV BITPT+2, TMPPT+2 ;...HI 64K.
3274 015350 032705 020000 BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
3275 015354 001530 BEQ 31$ ;BRANCH IF NOT.
3276 015356 013737 172344 172346 MOV @#KIPAR2, @#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3277 015364 062737 000200 172346 55: ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
3278 015372 006367 164152 ASL TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
3279 015376 006167 164150 ROL TMPPT+2 ;...HI POINTER.
3280 015402 000513 BMI 30$ ;BR IF NO MORE.
3281 015404 036767 164140 164116 65: BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3282 015412 001004 BNE 7$ ;BRANCH IF A MATCH.
3283 015414 036767 164132 164110 BIT TMPPT+2, TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3284 015422 001760 BEQ 5$ ;BRANCH IF NO MEMORY
3285 015424 036767 164120 164150 75: BIT TMPPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3286 015432 001004 BNE 8$ ;BRANCH IF A MATCH
3287 015434 036767 164112 164142 BIT TMPPT+2, LADMAP+2 ;CHECK HI 64K
3288 015442 001475 BEQ 31$ ;BR IF NO LAST BANK FLAG.
3289 015444 016705 164130 85: MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADDRESS.
```

```
3290 015450 052767 020000 164120 BIS #20000, TMAPLAD ;SET VIRTUAL ADR TO BANK 3.
3291 015456 000467 BR 31$
3292
3293 015460 026702 164112 10$: CMP TMAPLAD, R2 ;CHECK IF LAST ADR REACHED.
3294 015464 001064 BNE 31$ ;BR IF MORE.
3295 015466 000474 BR 32$ ;BR IF ALL DONE.
3296
3297 015470 106267 164063 20$: ASRB FLAGBK ;SHIFT BK FLAG
3298 015474 001407 BEQ 22$ ;BR IF NOT BK BLOCK.
3299 015476 103455 BCS 30$ ;BR IF ANOTHER 4K.
3300 015500 155067 164053 CLRB FLAGBK ;CLEAR OUT ALL FLAGS.
3301 015504 162702 040000 SUB #40000, R2 ;BACK UP BK
3302 015510 062702 020000 21$: ADD #20000, R2 ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3303 015514 106367 164024 22$: ASLB BITPT ;UPDATE POINTER.
3304 015520 100457 BMI 32$ ;BRANCH WHEN END IS REACHED.
3305 015522 036767 164018 164000 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS.
3306 015530 001767 BEQ 21$ ;BRANCH IF NO MATCH.
3307 015532 036767 164006 164042 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3308 015540 001402 BEQ 23$ ;BR IF NO MATCH.
3309 015542 016705 164032 MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3310 015546 016767 163772 163774 23$: BIT BITPT, TMPPT ;SET UP TMP POINTER.
3311 015554 032705 020000 BIT #BIT13, R5 ;CHECK FOR BK BLOCK SIZE.
3312 015560 001426 BEQ 31$ ;BRANCH IF SMALLER BLOCK SIZE.
3313 015562 163667 163762 ASLB TMPPT ;POINT TO NEXT BANK.
3314 015566 100421 BMI 30$ ;BRANCH IF OVERFLOW.
3315 015570 036767 163754 163732 BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3316 015576 001415 BEQ 30$ ;BRANCH IF NOT TO BE TESTED.
3317 015600 036767 163740 163774 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3318 015606 112767 000011 163743 MOVB #11, FLAGBK ;SET BK BLOCK FLAG.
3319 015614 036767 163724 163760 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3320 015622 001403 BEQ 30$ ;BR IF NO FLAG.
3321 015624 016705 163750 MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3322 015630 000402 BR 31$
3323 015632 012705 017777 30$: MOV #MASK4K, R5 ;SET MASK TO 4K.
3324 015636 056767 163702 163704 31$: BIS BITPT, TMPPT ;SET TMPPT FOR FINDING LAST ADR.
3325 015644 056767 163676 163700 BIS BITPT+2, TMPPT+2
3326 015652 016716 163676 MOV MMORE, (SP) ;FUDGE RETURN ADDRESS TO LOOP.
3327 015656 000207 RTS PC ;RETURN
3328
3329 015660 005767 164412 ;* BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
3330 015664 001402 32$: TST MPRX ;CHECK FOR ANY PARITY REGISTERS PRESENT.
3331 015666 004767 001744 BEQ 33$ ;BR IF NONE.
3332 015672 000207 33$: MSR PC, CKPWER ;CHECK FOR PARITY MEMORY ERRORS.
3333 RTS PC ;STRAIGHT RETURN.
3334
3335 ;*****
3336 ;* MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
3337 ;* FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
3338 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
3339 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3340 ;*****
3340 015674 036767 163644 163666 MMDDOWN: BIT BITPT, FADMAP ;CHECK FOR FIRST ADR FLAG.
3341 015702 001004 BNE 15$ ;BR IF FIRST ADR IN THIS BANK.
3342 015704 026767 163636 163660 BIT BITPT+2, FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
3343 015712 001404 BEQ 2$ ;BR IF NO FLAG
3344 015714 026702 163644 15: CMP TMPFAD, R2 ;CHECK IF FIRST ADDRESS REACHED.
3345 015720 001052 BNE 9$ ;BR IF MORE.
```

```
3346 015722 000453 BR 10$ ;BR IF ALL DONE.
3347 015724 005767 162656 2$: TST MMAVA ;CHECK IF MEM MGMT IS AVAILABLE
3348 015730 001425 BEQ 6$ ;BRANCH IF NOT
3349 015732 162737 000200 172344 3$: SUB #200, @#KIPAR2 ;LOWER MEM MGMT PAR BY 4K
3350 015740 006067 163602 RDR BITPT+2 ;MOV POINTER TO NEXT LOWER BANK...HI MAP.
3351 015744 026067 163574 RDR BITPT ;...LO MAP.
3352 015750 103440 BCS 10$ ;BR IF NO MORE.
3353 015752 036767 163566 163550 BIT BITPT, TSTMAP ;CHECK FOR BANK EXISTING
3354 015760 001004 BNE 4$ ;BR IF BANK TO BE TESTED.
3355 015762 036767 163560 163542 BIT BITPT+2, TSTMAP+2 ;CHECK FOR BANK IN HI MAP.
3356 015770 001760 BEQ 3$ ;BR IF NOT THERE.
3357 015772 012702 060000 4$: MOV #60000, R2 ;SET ADR POINTER TO TOP OF BANK
3358 015776 000411 BR 7$ ;GO TO COMMON EXIT
3359 016000 162702 020000 5$: SUB #20000, R2 ;BACK POINTER DOWN ONE BANK
3360 016004 006267 163534 6$: ASR BITPT ;MOVE POINTER TO NEXT LOWER BANK
3361 016010 103420 BCS 10$ ;BRANCH TO EXIT IF NO MORE MEM
3362 016012 036767 163526 163510 BIT BITPT, TSTMAP ;CHECK IF BANK EXISTS
3363 016020 001767 BEQ 5$ ;BRANCH IF BANK DOESN'T EXIST
3364 016022 026767 163516 163540 7$: BIT BITPT, FADMAP ;CHECK IF FIRST BANK FLAG.
3365 016030 001004 BNE 8$ ;BR IF FIRST BANK.
3366 016032 026767 163510 163532 BIT BITPT+2, FADMAP+2 ;CHECK IF FIRST BANK FLAG.
3367 016040 001402 BEQ 9$ ;BR IF NO FLAG FOUND.
3368 016042 016705 163520 8$: MOV FADMSK, R5 ;SET UP R5 TO FIND FIRST ADDRESS.
3369 016046 016716 163502 9$: MOV MMORE, (SP) ;RESET RETURN ADDRESS
3370 016052 000207 10$: RTS PC ;RETURN
```

```
3371 .SBTTL SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
3372 ;*****
3373 ;* SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
3374 ;* BITS 16 AND 17 ARE IN STMP0.
3375 ;*****
3376 016054 010200 PHYADR: MOV R2, R0 ;VIRTUAL INTO R0
3377 016056 005057 CLR STMP0 ;CLEAR TEMP SAVE OF HIGH BITS
3378 016062 005767 162520 TST MMAVA ;CHECK FOR MEM MGMT AVAILABLE
3379 016066 001417 BEQ 1$ ;BRANCH IF NO MEM MGMT
3380 016070 010146 MOV R1, -(SP) ;PUSH R1 ON STACK
3381 016072 013701 172344 MOV @#KIPAR2, R1 ;GET PAR TO BE ADDED TO VIRTUAL
3382 016076 006301 ASL R1 ;SHIFT IT 6 TIMES
3383 016100 006301 ASL R1
3384 016102 006301 ASL R1
3385 016104 006301 ASL R1
3386 016106 006301 ASL R1
3387 016110 006167 163044 RDL STMP0 ;SAVE EXTRA BITS
3388 016114 006301 ASL R1
3389 016116 006167 163036 RDL STMP0
3390 016122 060100 ADD R1, R0 ;ADD SHIFTED PAR TO VIRTUAL
3391 016124 012601 MOV (SP)+, R1 ;POP STACK INTO R1
3392 016126 000207 1$: RTS PC ;RETURN
3393
3394 ;*****
3395 ;* SUBROUTINE TO PUT BANK NUMBER INTO R0.
3396 ;*****
3397 016130 005000 BANKNO: CLR R0 ;INIT R0
3398 016132 010146 MOV R1, -(SP) ;PUSH R1 ON STACK
3399 016134 010246 MOV R2, -(SP) ;PUSH R2 ON STACK
3400 016136 016701 163402 MOV BITPT, R1 ;GET BANK MAP POINTER...LO 64K.
3401 016142 016702 163400 MOV BITPT+2, R2 ;...HI 64K.
3402 016146 006202 1$: ASR R2 ;SHIFT POINTER...HI
3403 016150 0060C1 RDR R1 ;...LO
3404 016152 103403 BCS 2$ ;BR WHEN POINTER FOUND.
3405 016154 1052C0 INCB R0 ;COUNT BANKS.
3406 016156 100373 BPL 1$ ;BR IF NOT OVERFLOW.
3407 016160 0000C0 HALT ;FATAL ERRDR!!! NO POINTER FOUND.
3408 016162
3409 016162 012602 2$: MOV (SP)+, R2 ;POP STACK INTO R2
3410 016164 012601 MOV (SP)+, R1 ;POP STACK INTO R1
3411 016166 000207 RTS PC ;RETURN
3412
3413 ;*****
3414 ;* SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
3415 ;*****
3416 016170 SETCON:
3417 016170 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3418 016174 004467 176232 MOV R0, (R2)+ ;MOV CONSTANT INTO MEMORY
3419 016176 030502 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
3420 016200 001375 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
3421 016202 004767 176776 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
3422 016206 000207 RTS PC ;RETURN
```

```

3423 ;*****
3424 ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
3425 ;*****
3426 016210 106112 ROTATE: ROLB (R2) ;(R2)=177776 OR 000001
3427 016212 106112 ROLB (R2) ;(R2)=177775 OR 000002
3428 016214 106112 ROLB (R2) ;(R2)=177773 OR 000004
3429 016216 106112 ROLB (R2) ;(R2)=177767 OR 000010
3430 016220 106112 ROLB (R2) ;(R2)=177757 OR 000020
3431 016222 106112 ROLB (R2) ;(R2)=177737 OR 000040
3432 016224 106112 ROLB (R2) ;(R2)=177677 OR 000100
3433 016226 106112 ROLB (R2) ;(R2)=177777 OR 000000
3434 016230 106122 ROLB (R2)+ ;(R2)=177577 OR 000200
3435 016232 106112 ROLB (R2) ;(R2)=177377 OR 000400
3436 016234 106112 ROLB (R2) ;(R2)=176777 OR 001000
3437 016236 106112 ROLB (R2) ;(R2)=175777 OR 002000
3438 016240 106112 ROLB (R2) ;(R2)=173777 OR 004000
3439 016242 106112 ROLB (R2) ;(R2)=167777 OR 010000
3440 016244 106112 ROLB (R2) ;(R2)=157777 OR 020000
3441 016246 106112 ROLL (R2) ;(R2)=137777 OR 040000
3442 016250 106112 ROLB (R2) ;(R2)=077777 OR 100000
3443 016252 106122 ROLB (R2)+ ;(R2)=177777 OR 000000
3444 016254 000207 RTS PC ;RETURN
3445
3446 ;*****
3447 ;* SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
3448 ;*****
3449 016256 012704 000020 W3X9: MOV #16, R4 ;EACH LOOP WRITES 256. WORDS
3450
3451 016262 010022 25: MOV R0, (R2)+
3452 016264 010022 MOV R0, (R2)+
3453 016266 010022 MOV R0, (R2)+
3454 016270 010022 MOV R0, (R2)+
3455
3456 016272 010322 MOV R3, (R2)+
3457 016274 010322 MOV R3, (R2)+
3458 016276 010322 MOV R3, (R2)+
3459 016300 010322 MOV R3, (R2)+
3460
3461 016302 010022 MOV R0, (R2)+
3462 016304 010022 MOV R0, (R2)+
3463 016306 010022 MOV R0, (R2)+
3464 016310 010022 MOV R0, (R2)+
3465
3466 016312 010322 MOV R3, (R2)+
3467 016314 010322 MOV R3, (R2)+
3468 016316 010322 MOV R3, (R2)+
3469 016320 010322 MOV R3, (R2)+
3470
3471 016322 005304 DEC R4
3472 016324 001356 BNE 25
3473 016326 000046 MOV R0, -(SP) ;SAVE R0
3474 016330 003000 MCV R3, R0 ;PUT R3 INTO R0
3475 016332 012603 MOV (SP)+, R3 ;PUT SAVED R0 INTO R3
3476 016334 000207 RTS PC ;RETURN
    
```

```

3477 .SBTTL RELOCATION SUBROUTINES.
3478 ;*****
3479 ;* ROUTINE TO RELOCATE PROGRAM CODE
3480 ;*****
3481 016336 RELOC:
3482 016336 010246 MOV R2, -(SP) ;:PUSH R2 ON STACK
3483 016340 010346 MOV R3, -(SP) ;:PUSH R3 ON STACK
3484 016342 010446 MOV R4, -(SP) ;:PUSH R4 ON STACK
3485 016344 012502 45: MOV (R5)+, R2 ;GET FIRST LOCATION.
3486 016346 012503 MOV (R5)+, R3 ;GET FIRST LOCATION OF DESTINATION.
3487 016350 012704 020000 MOV #20000, R4 ;SET UP BK COUNTER.
3488 016354 012223 15: MOV (R2)+, (R3)+ ;MOV THE DATA.
3489 016356 005304 DEC R4 ;COUNT THE WORDS.
3490 016360 001375 BNE 15 ;BR IF MORE.
3491 016362 012704 020000 MOV #20000, R4 ;RESET THE COUNTER.
3492 016366 024243 25: CMP -(R2), -(R3) ;CHECK THE DATA JUST MOVED.
3493 016370 001417 BEQ 35 ;BR IF DATA OK.
3494 016372 011267 162526 MCV (R2), $GDDAT ;GET SOURCE DATA.
3495 016376 011367 162524 MCV (R3), $BDADR ;GET DESTINATION DATA.
3496 016402 010267 162512 MOV R2, $GDDADR ;GET SOURCE ADDRESS.
3497 016406 010367 162510 MOV R3, $BDADR ;GET DESTINATION ADDRESS.
3498 016412 004767 003222 JSR PC, $ERRR ;*** ERROR *** (GO TYPE A MESSAGE)
3499 016416 000023 WORD 23 ;FATAL ERRR!!! RELOCATION FAILED.
3500 016420 000000 HALT
3501 016422 162705 000004 SUB #4, R5 ;ADJUST RETURN POINTER.
3502 016426 000746 BR 45 ;GO BACK AND TRY AGAIN.
3503 016430 005304 35: DEC R4 ;COUNT WORDS.
3504 016432 001355 BNE 25 ;BR IF MORE.
3505 016434 004567 005052 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3506 016440 026542 .WORD PRELOC ;ADDRESS OF MESSAGE TO BE TYPED
3507
3508 016442 010346 MOV R3, -(SP) ;PUT THE DATA ON THE STACK.
3509 016444 004767 JSR PC, $STYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
3510 016450 012604 MOV (SP)+, R4 ;POP STACK INTO R4
3511 016452 012603 MOV (SP)+, R3 ;POP STACK INTO R3
3512 016454 012602 MOV (SP)+, R2 ;POP STACK INTO R2
3513 016456 000205 RTS R5 ;RETURN
3514 ;*****
3515 ;* SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
3516 ;*****
3517 016460 022767 000003 162114 RELTOP: CMP #3, PRGMAP ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
3518 016466 001401 BEQ 15 ;BR IF OK
3519 016470 000000 HALT ;FATAL ERROR!!! PROG SHOULD BE IN BANKS 0 AND 1
3520
3521 016472 010046 15: MOV R0, -(SP) ;:PUSH R0 ON STACK
3522 016474 010146 MOV R1, -(SP) ;:PUSH R1 ON STACK
3523 016476 005767 162104 TST MMAVA
3524 016502 001465 BEQ 105
3525 016504 002737 007600 172346 MOV #7600, @#KIPAR3 ;SET PAR TO TOP OF MEM
3526 016512 005000 CLR R0 ;INIT BANK POINTER...LO 64K
3527 016514 012701 100000 MOV #BIT15, R1 ;...HI 64K.
3528 016520 162737 000200 172346 25: SLB #200, @#KIPAR3 ;BACK DOWN ONE BANK.
3529 016526 006001 RCR R1 ;MOVE POINTER...HI 64K.
3530 016530 006000 RCR R0 ;...LO 64K.
3531 016532 103500 BCS 905
3532 016534 000167 162766 BIT R1, MEMMAP+2 ;CHECK FOR BANK EXISTS.
    
```

```
3533 016540 001003 BNE 3$ ;BR IF AVAILABLE
3534 016542 030067 162756 BIT R0, MEMMAP ;CHECK FOR BANK EXISTS.
3535 016546 001764 BEQ 2$ ;BR IF NO BANK FOUND.
3536 016550 013777 172346 172344 3$: MOV @#KIPAR3,@#KIPAR2 ;COPY PAR
3537 016556 010046 MOV RO,-(SP) ;PUSH RO ON STACK
3538 016560 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
3539 016562 162737 000200 172344 4$: SUB #200, @#KIPAR2 ;BACK DOWN WITH LOW PAR.
3540 016570 060601 RC ;SHIFT POINTER.
3541 016572 060600 RDR RO ;...LO 64K.
3542 016574 103457 BCS 90$ ;BR IF OVERFLOW.
3543 016576 016167 162724 5$: BIT R1, MEMMAP+2 ;CHECK IF BANK EXISTS...HI 64K.
3544 016602 001003 BNE 6$ ;BR IF BANK EXISTS.
3545 016604 030067 162714 BIT R0, MEMMAP ;CHECK IF BANK EXISTS...LO 64K.
3546 016610 001764 BEQ 4$ ;BR IF BANK DOESN'T EXIST.
3547 016612 052601 6$: BIS (SP)+, R1 ;GET SECOND BANK POINTER.
3548 016614 052600 BIS (SP)+, RO ;...LO 64K.
3549 016616 030067 161760 BIT RO, PRGMAP ;CHECK FOR CONFLICT.
3550 016622 001044 BNE 90$ ;ABORT IF DESTINATION OVERLAYS SOURCE.
3551 016624 004567 177506 JSR R5, RELOC ;GO RELOCATE PROGRAM.
3552 016630 000000 .WORD 0 ;SOURCE FIRST ADDRESS.
3553 016632 040000 .WORD 40000 ;DESTINATION FIRST ADDRESS.
3554 016634 013737 172744 172340 MOV @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK.
3555 016642 013737 172346 172342 MOV @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
; * PROGRAM SHOULD NOW BE EXECUTING OUT OF LAST TWO BANKS OF MEMORY.
3557 016650 010167 161730 MOV R1, PRGMAP+2 ;RESET PROGRAM MAP.
3558 016654 000473 BR 30$ ;BR TO COMMON EXIT.
3559
3560 016656 012700 000400 10$: MOV #BITB, RO ;SET BANK POINTER TO TOP OF MEM.
3561 016662 050011 CLR R1 ;SET ADDRESS POINTER TO TOP.
3562 016664 162701 020000 11$: SUB #20000, R1 ;BACK DOWN ONE BANK.
3563 016670 060200 ASR RO ;MOVE POINTER DOWN ONE BANK.
3564 016672 103420 BCS 90$ ;BR IF OVERFLOW.
3565 016674 030067 162624 BIT R0, MEMMAP ;CHECK IF THIS BANK EXISTS.
3566 016700 001771 BEQ 11$ ;BR IF NON-EXISTANT BANK.
3567 016702 162701 020000 SUB #20000, R1 ;BACK DOWN TO NEXT BANK.
3568 016706 060200 ASR RO ;MOV POINTER DOWN ONE BANK.
3569 016710 103411 BCS 90$ ;BR IF OVERFLOW.
3570 016712 030067 162606 BIT R0, MEMMAP ;CHECK IF THIS BANK EXISTS.
3571 016716 001762 BEQ 11$ ;BR TO START OVER IF NO LOWER BANK.
3572 016720 010046 MOV RO, -(SP) ;SAVE THE POINTER.
3573 016722 060300 ASL RO ;RESET POINTER TO HI BANK.
3574 016724 052600 BIS (SP)+, RO ;SET BIT FOR LO BANK.
3575 016726 030067 161650 BIT RO, PRGMAP ;CHECK FOR A PROGRAM CONFLICT.
3576 016732 001401 BEQ 12$ ;BR IF NO CONFLICT.
3577 016734 90$:
3578 016734 000000 HALT ;FATAL ERROR!!! NOT ENOUGH MEMORY??
3579 016736 010167 000006 12$: MOV R1, 13$ ;SET DATA FOR RELOCATION SUBROUTINE.
3580 016742 004567 177370 JSR R5, RELOC ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3581 016746 000000 .WORD 0 ;SOURCE STARTING ADDRESS.
3582 016750 000000 .WORD 0 ;DESTINATION STARTING ADDRESS.
3583 016752 010167 161622 MOV R1, RELOC ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3584 016756 060107 ADD R1, PC ;JUMP TO RELOCATED PROGRAM
; * PROGRAM NOW EXECUTING OUT OF TOP OF MEMORY.
3585 ADD R1, SP ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3587 MOV R1, RELOC ;SET THE RELOCATION FACTOR.
3588 ADD R1, @#ERRVEC ;ADJUST ERROR VECTOR.
```

```
3589 016772 060137 000024 ADD R1, @#PWREVC ;ADJUST POWER FAIL VECTOR.
3590 016776 060137 000114 ADD R1, @#PARVEC ;ADJUST PAPITY ERROR VECTOR.
3591 017002 026727 162132 177570 CMP SWR, #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
3592 017010 001404 BEQ 14$ ;BR IF HARDWARE SWITCH REGISTER.
3593 017012 060167 162122 ADD R1, SWR ;ADJUST SOFTWARE SWITCH REGISTER.
3594 017016 060167 162120 ADD R1, DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3595 017020 062701 001622 14$: ADD #RADTAB,R1 ;POINT TO THE RELATIVE RELOCATION TABLE.
3596 017026 066721 161546 15$: ADD RELOC, (R1)+ ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
3597 017032 057221 16$ : TST (R1)+ ;CHECK FOR INTERUM TERMINATOR.
3598 017034 061776 BEQ 16$ ;BR SO AS TO NOT MODIFY ZERO.
3599 017036 024127 177777 CMP -(R1), #-1 ;CHECK FOR END OF TABLE.
3600 017042 001371 BNE 15$ ;BR IF MORE IN TABLE.
3601 017044 010067 161532 30$: MOV RO, PRGMAP ;SET NEW PROGRAM MAP...LO 64K.
3602 017050 012601 MOV (SP)+,R1 ;POP STACK INTO R1
3603 017052 012600 MOV (SP)+,RO ;POP STACK INTO RO
3604 017054 066716 161520 ADD RELOC, (SP) ;ADJUST RETURN ADDRESS.
3605 017060 000207 RTS PC ;RETURN
3606
3607 ;*****
3608 ;* SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
3609 ;*****
3610 017062 032767 000003 161512 RELO: BIT #3, PRGMAP ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3611 017070 001401 BEQ 1$ ;BR IF NO CONFLICT.
3612 017072 000000 HALT ;FATAL ERROR!!! PROGRAM ALREADY IN BANKS 0 OR 1!!!!
3613 017074 065767 161506 1$: TST MVA ;CHECK FOR NEW MGMT.
3614 017100 001417 BEQ 10$ ;BR IF NO MEMMGMT.
3615 017102 005037 172344 CLR @#KIPAR2 ;SET PAR 2 TO BANK 0.
3616 017106 012737 000200 172346 MOV #200, @#KIPAR3 ;SET PAR 3 TO BANK 1.
3617 017114 004567 177216 JSR R5, RELOC ;GO MOVE BK INTO BANKS 0 AND 1.
3618 017120 000000 .WORD 0 ;SOURCE STARTING ADDRESS.
3619 017122 040000 .WORD 40000 ;DESTINATION STARTING ADDRESS.
3620 017124 005037 172340 CLR @#KIPAR0 ;RESTORE PAR 0 TO BANK 0.
3621 017130 012737 000200 172342 MOV #200, @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
; * PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
3623 017136 000444 BR 30$ ;BR TO COMMON EXIT.
3624
3625 017140 016746 161434 10$: MOV RELOC, -(SP) ;PUT RELOCATION FACTOR ONTO THE STACK.
3626 017144 011667 000004 MOV (SP), 20$ ;SET DATA FOR RELOC SUBROUTINE.
3627 017150 004567 177162 JSR R5, RELOC ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
3628 017154 000000 .WORD 0 ;SOURCE STARTING ADDRESS.
3629 017156 000000 .WORD 0 ;DESTINATION STARTING ADDRESS.
3630 017160 161607 SUB (SP), PC ;JUMP TO RELOCATED PROGRAM.
; * THE PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
3632 017162 161613 SUB (SP), SP ;RESET THE STACK POINTER.
3633 017164 010046 MOV RO,-(SP) ;PUSH RO ON STACK
3634 017166 012700 001622 MOV #RADTAB,RO ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3635 017172 166820 000002 21$: SUB 2(SP), (RO)+ ;RESET ADDRESSES TO UNRELOCATED VALUES.
3636 017176 065720 22$: TST (RO)+ ;CHECK FOR TERMINATORS.
3637 017200 001776 BEQ 22$ ;BR OVER TERMINATORS.
3638 017202 024027 177777 CMP -(RO), #-1 ;CHECK FOR END OF TABLE INDICATOR.
3639 017206 011371 BNE 21$ ;BR IF MORE ADDRESSES IN TABLE.
3640 017210 012600 MOV (SP)+,RO ;POP STACK INTO RO
3641 017212 161637 000004 SUB (SP), @#ERRVEC ;ADJUST ERROR VECTOR.
3642 017216 161637 000024 SUB (SP), @#PWREVC ;ADJUST POWER FAIL VECTOR.
3643 017222 161637 000114 SUB (SP), @#PARVEC ;ADJUST PAPITY ERROR VECTOR.
3644 017226 026727 161706 177570 CMP SWR, #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
```



```

3645 017234 001404 BEQ 23$ ;BR IF HARDWARE SWITCH REGISTER.
3646 017235 161667 161676 SUB (SP), SWR ;ADJUST SOFTWARE SWITCH REGISTER.
3647 017242 161667 161674 SUB (SP), DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3648 017246 162616 23$: SUB (SP)+, (SP) ;ADJUST RETURN ADDRESS.
3649 017250 005067 161324 30$: CLR RELOCF ;RESET RELOCATION FACTOR.
3650 017254 012767 000003 161120 MOV #3, PRCMAP ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
3651 017262 005067 161316 CLR PRGMAP+2 ;...HI 64K.
3652 017266 000207 RTS PC ;RETURN.
3653
3654
3655 ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
3656 ;* WHENCE IT CAME. THE LOADER AREA IS SAVED AT THE END OF THE BK OF
3657 ;* PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
3658
3659 017270 016700 162224 RESLDR: MOV LMAD, R0 ;CHECK IF THE LOADERS WERE SAVED.
3660 017274 001001 BNE 1$ ;BR IF LOADER AREA WAS SAVED.
3661 017276 000000 HALT ;FATAL ERROR!!! CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
3662 017300 005767 161302 1$: TS MMAPVA ;CHECK FOR MEM MGMT.
3663 017304 001402 BEO 2$ ;SKIP IF NO MEM MGMT.
3664 017308 005037 177572 CLR @#SR0 ;DISABLE MEM MGMT.
3665 017312 012701 040000 2$: MOV #40000, R1 ;GET END OF BK, ASSUME PROG NOT RELOCATED.
3666 017316 012702 002734 MOV #1500., R2 ;GET COUNTER.
3667 017322 014140 3$: MOV -(R1), -(R0) ;MOVE THE LOADER AREA.
3668 017324 0053L2 DEC R2 ;COUNT HOW LONG THE AREA IS.
3669 017326 001375 BNE 3$ ;BR IF NOT MORE TO MOVE.
3670 017330 005067 162164 CLR LMAD ;CLEAR MONITOR SAVED FLAG
3671 017334 005767 161246 TST MMAPVA ;CHECK FOR MEM MGMT.
3672 017340 001402 BLJ 4$ ;BR IF NO MEM MGMT.
3673 017342 005237 177572 INC @#SR0 ;ENABLE MEM MGMT.
3674 017346 000207 4$: RTS PC ;RETURN.
3675
3676 ;* ROUTINE TO SAVE THE LOADERS AT THE END OF BK.
3677 017350 005767 162144 SAVLDR: TST LMAD ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
3678 017354 001024 BNE 4$ ;BRANCH IF ALREADY SAVED
3679 017356 012700 040000 MOV #40000, R0 ;GET END OF BK
3680 017362 010001 MOV R0, R1 ;GET END OF BK
3681 017364 012737 017376 000004 MOV #2$, @#ERRVEC ;SET UP TIMEOUT VECTOR
3682 017372 011020 1$: MOV (R0), (R0)+ ;SEARCH FOR END OF MEMORY
3683 017374 000776 BR 1$ ;KEEP SEARCHING
3684 017376 025826 2$: CMP (SP)+, (SP)+ ;RESTORE STACK POINTER
3685 017400 012737 025114 000004 MOV #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
3686 017406 010046 MOV R0, -(SP) ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3687 017410 012702 002734 MOV #1500., R2 ;SET UP WORD COUNTER
3688 017414 014041 3$: MOV -(R0), -(R1) ;SAVE THE LOADERS
3689 017416 005302 DEC R2 ;COUNT THE WORDS
3690 017420 001375 BNE 3$ ;BRANCH IF MORE WORDS
3691 017422 012667 162072 MOV (SP)+, LMAD ;SAVE LAST MEMORY ADDRESS
3692 017426 000207 4$: RTS PC ;RETURN
    
```

```

3693 .SBTTL PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
3694
3695 ;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
3696 ;* FIND OUT WHICH REGISTER DETECTED THE ERROR.
3697 ;* THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
3698
3699 017430 011667 161465 P$SRV: MOV (SP), SBDADR ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
3700 017434 004567 004052 JLT R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3701 017440 026501 .WORD UNEXPT ;ADDRESS OF MESSAGE TO BE TYPED
3702 ;"UNEXPECTED MEMORY PARITY TRAP."
3703 017442 010146 MOV R1, -(SP) ;PUSH R1 ON STACK
3704 017444 010346 MOV R3, -(SP) ;PUSH R3 ON STACK
3705 017446 016703 162156 MOV .MPRX, R3 ;GET POINTER TO PARITY REGISTERS.
3706 017452 005733 TST @#(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3707 017454 100415 1$: BMI 3$ ;BR IF THIS REGISTER SHOWS THE ERROR.
3708 017456 005713 TST (R3) ;CHECK FOR TABLE TERMINATOR.
3709 017460 001374 BNE 1$ ;BR IF MORE REGISTERS.
3710 017462 004767 002152 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3711 ;***ERROR*** NO REGISTER INDICATED ERROR
3712 017466 000024 .WORD 24 ;ERROR TYPE CODE.
3713 017470 000417 BR 4$ ;EXIT
3714 017472 005713 2$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
3715 017474 001415 BEQ 4$ ;BR IF NO MORE PARITY REGISTERS.
3716 017476 005733 TST @#(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3717 017500 100374 BPL 2$ ;BR IF NO ERROR FLAG.
3718 017502 004567 004004 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3719 017506 026572 .WORD MTOE ;ADDRESS OF MESSAGE TO BE TYPED
3720 ;"MORE THAN ONE ERROR FOUND."
3721 017510 3$:
3722 017510 004767 000610 JSR PC, SPRNTQ ;SET UP VALUES FOR ERROR PRINTING.
3723 017514 004767 002120 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3724 017520 000025 .WORD 25 ;ERROR TYPE CODE.
3725 017522 004767 000216 JSR PC, PSCAN ;GO SCAN MEMORY FOR BAD PARITY.
3726 017526 000761 BR 2$ ;GO LOOK FOR MORE ERRORS.
3727 017530 4$:
3728 017530 012603 MOV (SP)+, R3 ;POP STACK INTO R3
3729 017532 012601 MOV (SP)+, R1 ;POP STACK INTO R1
3730 017534 000020 RTI ;RETURN.
3731
3732 ;* ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
3733 ;* THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
3734
3735 MAMF: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
3736 017536 005767 162534 BEQ MAMF2 ;EXIT IF NO PARITY REGISTERS.
3737 017542 001434 BIT #SW6, @SWR ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
3738 017544 002777 000100 161366 BNE MAMF2 ;EXIT IF NO PARITY ERROR DETECTION.
3739 017546 001030 TST RELOCF ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
3740 017554 005767 161020 BEQ SETAE ;BR IF PROG IN BANK 0.
3741 017560 001410 BIT #SW5, @SWR ;CHECK IF VECTORS PROTECTED.
3742 017562 002777 000040 161350 BNE SETAE ;BR IF VECTOR AREA PROTECTED.
3743 017570 001004 CMP FSTADR, #1000 ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
3744 017572 026727 161764 001000 BLM MAMF2 ;EXIT IF VECTORS EXPOSED TO TESTING.
3745 017600 103415
    
```

```

3746 017602 016737 162030 000114 SETAE: MOV    ,PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
3747 017610 005037 000116 CLR    @#PARVEC+2 ;PRIORITY LEVEL 0 ON TRAP
3748 017614 010346 MOV    R3,-(SP) ;PUSH R3 ON STACK
3749 017616 016703 162006 MOV    ,MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
3750 017622 052733 000001 MAMF1: BIS    #AE, @(R3)+ ;SET ACTION ENABLE BIT IN PARITY REG
3751 017626 005713 TST    (R3) ;CHECK FOR END OF TABLE.
3752 017630 001374 BNE    MAMF1 ;BR IF MORE PARITY REGISTERS.
3753 017632 012603 MOV    (SP)+,R3 ;POP STACK INTO R3
3754 017634 000207 MAMF2: RTS    PC ;RETURN.
3755
3756 ;*****
3757 ;* SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
3758 ;*****
3759 017636 005767 1624: CKPMER: TST    MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
3760 017642 001437 BEQ    4$ ;BR IF NO PARITY REGISTERS.
3761 017644 032777 000100 161266 BIT    #SW6, @SWR ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
3762 017652 001033 BNE    4$ ;BR IF PARITY ERROR CHECKING INHIBITED.
3763 017654 010346 MOV    R3,-(SP) ;PUSH R3 ON STACK
3764 017656 016703 161746 MOV    ,MPRX, R3 ;GET PARITY REG TABLE POINTER.
3765 017662 005733 1$: TST    @(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3766 017664 100023 BPL    3$ ;BR IF NO ERROR
3767 017666 032773 000001 177776 BIT    #BIT0, @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
3768 017674 001010 BNE    2$ ;BR IF NO ACTION ENABLE.
3769 017676 004767 000422 64$: JSR    PC, SPRNTQ ;SET UP VALUES FOR ERROR PRINTING.
3770 017702 004767 001732 JSR    PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3771 017706 000026 .WORD 26 ;ERROR TYPE CODE.
3772 017710 000411 BR    3$
3773 017712 004767 000026 2$: JSR    PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3774 017716
3775 017716 004767 000402 65$: JSR    PC, SPRNTQ ;SET UP VALUES FOR ERROR PRINTING.
3776 017722 004767 001712 JSR    PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3777 017726 000027 .WORD 27 ;ERROR TYPE CODE.
3778 017730 004767 000010 JSR    PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3779 017734 005713 3$: TST    (R3) ;CHECK FOR TABLE TERMINATOR.
3780 017736 001351 BNE    1$ ;BR IF MORE.
3781 017740 012603 MOV    (SP)+,R3 ;POP STACK INTO R3
3782 017742 000207 4$: RTS    PC ;RETURN.
3783
3784 ;*****
3785 ;* THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY.
3786 ;* TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
3787 ;* LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
3788 ;*****

```

```

3789 017744 PSCAN: MOV    R0,-(SP) ;PUSH R0 ON STACK
3790 017744 010046 MOV    R1,-(SP) ;PUSH R1 ON STACK
3791 017746 010146 MOV    R2,-(SP) ;PUSH R2 ON STACK
3792 017750 010246 MOV    R3,-(SP) ;PUSH R3 ON STACK
3793 017752 010346 MOV    R4,-(SP) ;PUSH R4 ON STACK
3794 017754 010446 MOV    @#114,-(SP) ;PUSH @#114 ON STACK
3795 017756 013746 000114 MOV    @#116,-(SP) ;PUSH @#116 ON STACK
3796 017762 013746 000116 MOV    @#116,-(SP) ;PUSH @#116 ON STACK
3797 017766 004567 003520 JSR    R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3798 017772 026636 .WORD SCANM ;"SCANNING MEMORY FOR BAD PARITY."
3799
3800 017774 012700 000001 MOV    #BIT0, R0 ;SET BIT POINTER TO FIRST BANK.
3801 020000 005001 CLR    R1 ;CLR HI 64K POINTER.
3802 020002 005002 CLR    R2 ;INIT ADDRESS POINTER.
3803 020004 005004 CLR    R4 ;INIT ERROR DETECTED FLAG.
3804 020006 004767 000256 JSR    PC, CLRPAR ;CLEAR THE PARITY REGISTERS.
3805 020012 012737 000116 000114 MOV    #116, @#114 ;HALT IF ANOTHER PARITY TRAP.
3806 020020 005037 000116 CL: @#116
3807 020024 005767 160556 MVAVA ;CHECK FOR MEMORY MANAGEMENT.
3808 020030 001406 BEQ    1$ ;BR IF NO MEM MGMT.
3809 020032 013746 172344 MOV    @#KIPAR2,-(SP) ;PUSH @#KIPAR2 ON STACK
3810 020036 005037 172344 CLR    @#KIPAR2 ;INIT MEM MGMT TO POINT TO BANK 0.
3811 020042 012702 040000 MOV    #40000, R2 ;SET ADR POINTER TO PAR2.
3812 020046 030067 161452 1$: BIT    R0, MEMMAP ;CHECK IF THIS BANK OF MEM EXISTS.
3813 020052 001003 BNE    2$ ;BR IF THIS BANK EXISTS.
3814 020054 030167 161446 BIT    R1, MEMMAP+2 ;CHECK HI 64K MAP.
3815 020060 001442 BEQ    10$ ;BR IF THIS BANK DOESN'T EXIST.
3816
3817 020062 010146 2$: MOV    R1,-(SP) ;PUSH R1 ON STACK
3818 020064 111201 3$: MOV    (R2), R1 ;READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
3819 020066 015703 161536 MOV    ,MPRX, R3 ;SET UP POINTER TO PARITY REGISTERS.
3820 020072 005733 4$: TST    @(R3)+ ;CHECK FOR THE ERROR FLAG.
3821 020074 100024 BPL    6$ ;BR IF NO ERROR FLAG.
3822 020076 005704 TST    R4 ;CHECK IF FIRST ERROR, THIS SCAN.
3823 020100 001003 BNE    5$ ;BR IF MORE THAN ONE ERROR FOUND.
3824 020102 005367 161004 DEC    $ERTTL ;ADJUST ERROR COUNT.
3825 020106 005204 INC    R4 ;SET FLAG TO INDICATE ERROR FOUND.
3826
3827 020110 004767 000210 5$: JSR    PC, SPRNTQ ;SET UP VALUES FOR ERROR PRINTING.
3828 020114 004767 001520 64$: JSR    PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3829 020120 000030 .WORD 30 ;ERROR TYPE CODE
3830 020122 111212 MOV    (R2), (R2) ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
3831 020124 005053 CLR    @-(R3) ;CLEAR THE ERROR FLAG.
3832 020126 105712 TST    (R2) ;CHECK IF THE PARITY ERROR WAS CLEARED.
3833 020130 005733 TST    @(R3)+ ;CHECK FOR THE ERROR FLAG.
3834 020132 100005 BPL    6$ ;BR IF IT IS OK.
3835 020134 004567 003352 JSR    R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3836 020140 026700 .WORD PEWNC ;ADDRESS OF MESSAGE TO BE TYPED
3837
3838 020142 005073 177776 CLR    @-2(R3) ;"PARITY ERROR WILL NOT CLEAR."
3839 020146 005713 65$: TST    (R2) ;CLEAR OUT THE PARITY ERROR FLAG.
3840 020150 001350 TST    (R3) ;CHECK FOR THE END OF REG ADR TABLE.
3841 020152 005202 BNE    4$ ;BR IF MORE PARITY REGISTERS.
3842 020154 032702 017777 INC    R2 ;GO TO NEXT MEMORY ADDRESS.
3843 020160 001341 BIT    #MASK4K,R2 ;CHECK FOR END OF 4K BANK.
3844 020162 012601 BNE    3$ ;BR IF MORE MEMORY THIS BANK.
3845 020164 001341 MOV    (SP)+,R1 ;POP STACK INTO R1

```

```

3845 020164 000402 BR 115 ;BR TO CHECK FOR NEXT BANK.
3846 020166 062702 020000 105: ADD #20000, R2 ;SKIP BANKS THAT AREN'T THERE.
3847 020172 005767 160410 115: TST MMAVA ;CHECK FOR MEM MGMT.
3848 020176 001413 BEQ 125 ;BR IF NO MEM MGMT.
3849 020200 062737 000200 172344 ADD #200, @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
3850 020206 012702 040000 MOV #40000, R2 ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
3851 020212 066300 ASL R0 ;UPDATE BANK POINTER.
3852 020214 006101 RC R1 ;...HI 64K.
3853 020216 100313 BPL 15 ;BR IF MORE BANKS.
3854 020220 012637 172344 MOV (SP)+, @#KIPAR2 ;;POP STACK INTO @#KIPAR2
3855 020224 070402 BR 205 ;GO CHECK IF ANY ERRORS FOUND.
3856 020226 166300 125: ASLB R0 ;UPDATE POINTER TO NEXT BANK.
3857 020230 100306 BPL 15 ;BR IF MORE BANKS.
3858 020232 065704 205: TST R4 ;CHECK IF ANY PARITY ERRORS DETECTED.
3859 020234 001003 BNE 215 ;BR IF ERRORS DETECTED.
3860 020236 004567 003250 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3861 020242 025706 .WORD NOPES ;ADDRESS OF MESSAGE TO BE TYPED
3862 020244 215:
3863 020244 012637 000116 MOV (SP)+, @#116 ;;POP STACK INTO @#116
3864 020250 012637 000114 MOV (SP)+, @#114 ;;POP STACK INTO @#114
3865 020254 012604 MOV (SP)+, R4 ;;POP STACK INTO R4
3866 020256 012603 MOV (SP)+, R3 ;;POP STACK INTO R3
3867 020260 012602 MOV (SP)+, R2 ;;POP STACK INTO R2
3868 020262 012601 MOV (SP)+, R1 ;;POP STACK INTO R1
3869 020264 012500 MOV (SP)+, R0 ;;POP STACK INTO R0
3870 020266 000207 RTS PC ;RETURN.
3871
3872 ;*****
3873 ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
3874 ;*****
3875 CLRPAR:
3876 020270 010346 MOV R3, -(SP) ;;PUSH R3 ON STACK
3877 020272 016703 161332 MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
3878 020276 065713 15: TST (R3) ;CHECK FOR THE TABLE TERMINATOR.
3879 020300 001402 BEQ 25 ;BR IF DONE ALL PARITY REGISTERS.
3880 020302 005033 CLR @ (R3)+ ;CLEAR THE PARITY REGISTER.
3881 020304 000774 BR 15 ;BR FOR MORE
3882 25:
3883 020306 MOV (SP)+, R3 ;;POP STACK INTO R3
3884 020310 000207 RTS PC ;RETURN.
3885
3886 .SBTTL SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
3887 ;*****
3888 ;* THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.SCMTAG)
3889 ;* FOR ERROR PRINTOUT BY .SERROR & .SERRTYP ROUTINES FROM **SYSMAC**
3890 ;*****
3891 020312 010267 160602 SPRINT: MOV R2, SGDADR ;SAVE THE ADDRESS UNDER TEST.
3892 020316 005067 160602 CLR SGDDAT ;SHOULD BE DATA IS "0".
3893 020322 000430 BNE SPRNTB
3894
3895 020324 014367 160630 SPRNTQ: MOV -(R0), STMP0 ;GET THE PARITY REGISTER ADDRESS.
3896 020330 013367 160626 MOV @ (R3)+, STMP1 ;GET THE CONTENTS OF THE PARITY REG.
3897 020334 000402 BR SPRNT0
3898
3899 020336 011367 160616 SPRNTP: MOV (R3), STMP0 ;GET THE PARITY REGISTER ADDRESS.
3900 020342 010267 160552 SPRNT0: MOV R2, SGDADR ;GET THE MEMORY ADDRESS BEING TESTED
    
```

```

3901 020346 000414 BR SPRNTA ;BR TO COMMON SECTION.
3902
3903 020350 010267 160544 SPRNT1: MOV R2, SGDADR ;GET THE MEMORY ADDRESS BEING TESTED
3904 020354 005367 160540 DEC SGDADR ;ADJUST IT FOR PRINTOUT.
3905 020360 000407 BR SPRNTA ;BR TO COMMON SECTION.
3906
3907 020362 010367 160572 SPRNT2: MOV R3, STMP0 ;GET THE DATA IN R3.
3908 020366 010267 160526 SPRNT3: MOV R2, SGDADR ;GET THE MEMORY ADDRESS BEING TESTED
3909 020372 162767 000002 160520 SUB #2, SGDADR ;ADJUST IT FOR PRINTOUT.
3910 020400 010067 160520 SPRNTA: MOV R0, SGDDAT ;GET WHAT THE DATA SHOULD BE
3911 020404 010167 160516 SPRNTB: MOV R0, SGDDAT ;GET WHAT THE DATA WAS
3912 020410 000207 RTN PC ;RETURN TO ENTER ERROR ROUTINES
3913
3914 ;*****
3915 ;* SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
3916 ;* R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
3917 ;*****
3918 TYPMAP: TST (R0) ;CHECK IF ANY MEMORY IN MAP...LO 64K.
3919 020414 001007 BNE 15 ;BR IF MEMORY IN MAP.
3920 020416 005760 000002 TST 2(R0) ;...HI 64K.
3921 020422 001004 BNE 15 ;BR IF MEMORY IN MAP.
3922 020424 004567 003062 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3923 020430 026266 .WORD NOMEM ;ADDRESS OF MESSAGE TO BE TYPED
3924 ;"NO MEMORY FOUND."
3925 020432 000475 BR 65 ;EXIT
3926 15:
3927 020434 010146 MOV R1, -(SP) ;;PUSH R1 ON STACK
3928 020436 010246 MOV R2, -(SP) ;;PUSH R2 ON STACK
3929 020440 010346 MOV R3, -(SP) ;;PUSH R3 ON STACK
3930 020442 010446 MOV R4, -(SP) ;;PUSH R4 ON STACK
3931 020444 012701 000001 MOV #BIT0, R1 ;SET UP BANK POINTER...LO 64K.
3932 020450 005002 CLR R2 ;...HI 64K.
3933 020452 012703 177777 MOV #-1, R3 ;SET UP ADDRESS POINTER TO -1.
3934 020456 010304 MOV R3, R4 ;HI BITS OF ADDRESS AS WILL.
3935 020460 030110 25: BIT R1, (R0) ;CHECK THE MAP FOR THIS BANK.
3936 020462 001014 BNE 35 ;BR IF THIS BANK PRESENT.
3937 020464 030260 000002 BIT R2, 2(R0) ;CHECK HI 64K MAP.
3938 020470 001011 BNE 35 ;BR IF THIS BANK PRESENT.
3939 020472 105703 TSTB R3 ;CHECK FOR PREVIOUS PRINTOUT.
3940 020474 001042 BNE 55 ;BR IF ALREADY TYPED "TO"
3941 020476 105703 000001 SUB #1, R3 ;BACK UP TO LAST ADDR OF PREVIOUS BANK.
3942 020502 005604 SBC R4 ;...HI ADDRESS BITS.
3943 020504 004567 003002 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3944 020510 025517 .WORD TO ;ADDRESS OF MESSAGE TO BE TYPED
3945 020512 000410 BR 45 ;GO TO TYPE THE ADDRESS.
3946 020514 105703 35: TSTB R3 ;CHECK FOR PREVIOUS TYPEOUT.
3947 020516 001431 BEQ 55 ;BR IF ALREADY TYPE "FROM".
3948 020520 062703 000001 ADD #1, R3 ;POINT TO FIRST ADDRESS OF THIS BANK.
3949 020524 005504 ADC R4 ;...HI BITS OF ADDRESS.
3950 020526 004567 002760 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3951 020532 025507 .WORD FROM ;ADDRESS OF MESSAGE TO BE TYPED
3952 45:
3953 020534 010346 MOV R3, -(SP) ;;PUSH R3 ON STACK
3954 020536 010446 MOV R4, -(SP) ;;PUSH R4 ON STACK
3955 020540 006303 ASL R3 ;BIT 15 INTO C-BIT
3956 020542 006104 ROL R4 ;BIT 15 INTO R4.
    
```

```
3957 020544 006003 ROR R3 ;RESTORE BITS 14-0.
3958 020546 010446 MOV R4,-(SP) ;SAVE R4 FOR TYPEOUT
3959 ;TYPE ADDRESS BITS 21-15
3960 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOS ROUTINE
3961 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3962 020550 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3963 020554 004767 004104 JSR PC, STYPOS ;GO TO THE SUBROUTINE
3964 020560 003 L BYTE 3 ;TYPE 3 DIGIT(S)
3965 020561 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
3966 020562 010346 MOV R3,-(SP) ;SAVE R3 FOR TYPEOUT
3967 ;TYPE ADDRESS BITS 14-0
3968 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOS ROUTINE
3969 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3970 020564 013746 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3971 020570 004767 004070 JSR PC, ST.POS ;GO TO THE SUBROUTINE
3972 020574 005 .BYTE 5 ;TYPE 5 DIGIT(S)
3973 020575 001 .BYTE 1 ;TYPE LEADING ZEROS
3974 020576 012604 MOV (SP)+,R4 ;POP STACK INTO R4
3975 020600 012603 MOV (SP)+,R3 ;POP STACK INTO R3
3976 020602 062703 020000 5S: ADD #20000,R3 ;UPDATE TO NEXT BANK.
3977 020606 005504 ADC R4 ;...HI ADDRESS BITS.
3978 020610 006301 ASL R1 ;SHIFT POINTER...LO 64K.
3979 020612 006102 ROL R2 ;...HI 64K.
3980 020614 103321 BCC 2S ;BR IF MORE BANKS.
3981 020616 012604 MOV (SP)+,R4 ;POP STACK INTO R4
3982 020620 012603 MOV (SP)+,R3 ;POP STACK INTO R3
3983 020622 012602 MOV (SP)+,R2 ;POP STACK INTO R2
3984 020624 012601 MOV (SP)+,R1 ;POP STACK INTO R1
3985 020626 002027 6S: RTS PC ;RETURN.
3986
3987 .SBTTL SCOPE HANDLER ROUTINE
3988
3989 ;*****
3990 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3991 ;*AND LOAD THE TEST NUMBER(STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3992 ;*AND LOAD THE ERROR FLAG(SERFLG) INTO DISPLAY<4:08>
3993 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3994 ;*SW14=1 LOOP ON TEST
3995 ;*SW11=1 INHIBIT ITERATIONS
3996 ;*SW09=1 LOOP ON ERROR
3997 ;*SW08=1 LOOP ON TEST IN SWR<4:0>
3998 ;*CALL
3999 ;* SCOPE ;:SCOPE=IOT
4000
4001 .SCOPE:
4002 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4003 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4004 020630 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4005 020634 004767 001524 JSR PC, SCKSWR ;GO TO THE SUBROUTINE
4006 020640 012504 MOV (R5)+, R4 ;SAVE MINIMUM BLOCK MASK NEXT TEST.
4007 020642 010516 MOV R5, (SP) ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
4008 020644 032777 040000 160266 1S: BIT #BIT14,@SWR ;LOOP ON PRESENT TEST?
4009 020652 001117 BNE SOVER ;YES IF SW14=1
4010 ;####START OF CODE FOR THE XOR TESTER####
4011 020654 000416 6S XTSTR: BR 6S ;IF RUNNING ON THE "XOR" TESTER CHANGE
4012 ;THIS INSTRUCTION TO A "NOP" (NOP=240)
```

```
4013 020656 013746 000004 MOV @#ERRVEC, -(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
4014 020662 012737 020702 000004 MOV #5S,@#ERRVEC ;SET FOR TIMEOUT
4015 020670 005737 107060 TST @#177060 ;TIME OUT ON XOR?
4016 020674 012637 000004 MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
4017 020700 000466 BR SSVLAD ;GO TO THE NEXT TEST
4018 020702 022626 5S: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
4019 020704 012637 000004 MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
4020 020710 000426 BR 7S ;LOOP ON THE PRESENT TEST
4021 020712 6S:####END OF CODE FOR THE XOR TESTER####
4022 020712 022777 000400 160220 BIT #BIT09,@SWR ;LOOP ON SPEC. TEST?
4023 020720 001407 BEQ 2S ;BR IF NO
4024 020722 017746 160212 MOV @SWR, -(SP) ;SET DESIRED TEST NUM. FROM SWR
4025 020726 042716 000340 BIC #SSWRMK,(SP) ;STRIP AWAY UNDESIRED BITS
4026 020732 122667 160112 CMPB (SP)+,STSTNM ;ON THE RIGHT TEST?
4027 020736 001465 BEQ SOVER ;BR IF YES
4028 020740 105767 160137 2S: TSTB SERFLG ;HAS AN ERROR OCCURRED?
4029 020744 001421 BEQ 3S ;BR IF NO
4030 020746 126767 160143 160127 CMPB SERMAX,SERFLG ;MAX. ERRORS FOR THIS TEST OCCURRED?
4031 020754 101015 BHI 3S ;BR IF NO
4032 020756 022777 001000 160154 BIT #BIT09,@SWR ;LOOP ON EFROR?
4033 020764 001404 BEQ 4S ;BR IF NO
4034 020766 016767 160116 160112 7S: MOV $LPERR,$LPADR ;SET LOOP ADDRESS TO LAST SCOPE
4035 020774 000446 BR SOVER
4036 020776 105067 160101 4S: CLR SERFLG ;ZERO THE ERROR FLAG
4037 021002 005067 160162 CLR $TIMES ;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4038 021006 000415 BR 1S ;ESCAPE TO THE NEXT TEST
4039 021010 032777 004000 160122 3S: BIT #BIT11,@SWR ;INHIBIT ITERATIONS?
4040 021016 001011 BNE 1S ;BR IF YES
4041 021020 005767 160166 TST $PASS ;IF FIRST PASS OF PROGRAM
4042 021024 001406 BEQ 1S ;INTO INHIBIT ITERATIONS
4043 021026 005267 160052 INC $ICNT ;INCREMENT ITERATION COUNT
4044 021032 026767 160132 160044 CMP $TIMES,$ICNT ;CHECK THE NUMBER OF ITERATIONS MADE
4045 021040 002024 BGE SOVER ;BR IF MORE ITERATION REQUIRED
4046 021042 012767 000001 160034 1S: MOV #1,$ICNT ;REINITIALIZE THE ITERATION COUNTER
4047 021050 016767 000552 160112 MOV $MXCNT,$TIMES ;SET NUMBER OF ITERATIONS TO DO
4048 021056 105267 160020 SSVLAD: INCJ $STSTNM ;COUNT TEST NUMBERS
4049 021062 116767 160014 160120 MOVJ $STSTNM,$IESTN ;SET TEST NUMBER IN APT MAILBOX
4050 021070 011667 160012 MCV (SP), $LPADR ;SAVE SCOPE LOOP ADDRESS
4051 021074 011667 160010 MCV (SP), $LPERR ;SAVE ERROR LOOP ADDRESS
4052 021100 005067 160056 CLR $ESCAPE ;CLEAR THE ESCAPE FROM ERROR ADDRESS
4053 021104 112767 000001 160003 #1,SERMAX ;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4054 021112 016777 157764 160022 $OVER: MOV $STSTNM,@DISP ;DISPLAY TEST NUMBER
4055 021120 016716 157762 MOV $LPADR,(SP) ;FUOGE RETURN ADDRESS
4056 021124 0205 3S INSERT: CMP R5, (SP) ;CHECK FOR LOOP ON TEST.
4057 021126 001402 BEQ 1S ;BR IF START NEXT TEST.
4058 021130 000167 000470 JMP ENDINS ;JMP IF LOOP ON LAST TEST.
4059 021134 012767 037777 160444 1S: MOV #37777, $BLKMSK ;SET BK BOUNDRY MASK.
4060 021142 005767 160044 TST $PASS ;CHECK FOR PASS 0.
4061 021146 001404 BEQ 2S ;BR IF PASS 0
4062 021150 126777 157726 000021 CMPB $STSTNM, #21 ;CHECK IF IN SECTION 3.
4063 021156 113002 3S BHI 3S ;BR IF IN SECTION 3.
4064 021160 005267 160422 2S: ASR $BLKMSK ;RESET BOUNDRY TO 4K.
4065 021164 016767 160372 160372 3S: MOV $FSTADR, $TMPFAD ;SET FIRST ADDRESS.
4066 021172 005767 157402 TST RELOC ;CHECK IF PG RELOCATED.
4067 021176 001430 BEQ 4S ;BR IF NOT RELOCATED.
4068 021200 032777 000040 157732 BIT #SW05, @SWR ;CHECK IF LOC 0-776 TO BE PROTECTED.
```

```

CZQMCF0 0-124K MEMORY EXERCISER, 16K VER MACY11 30A(1052) 20-FEB-78 07:56 PAGE 84
CZQMCF.P11 14-FEB-78 08:19 SCOPE HANDLER ROUTINE SEQ 0166

4069 021206 001424 BEQ 4S ;BR IF SW NOT SET.
4070 021210 026727 160350 001000 CMP TMPFAD, #1000 ;CHECK IF NOT BEING TESTED.
4071 021216 133020 4S ;BR IF ALREADY PROTECTED.
4072 021220 012767 001000 160336 MOV #1000, TMPFAD ;RESET FIRST ADDRESS.
4073 021226 052767 000001 160334 BIS #BIT0, FADMAP ;SET FLAG IN FIRST BANK.
4074 021234 026727 160334 001000 CMP LSTADR, #1000 ;CHECK IF GONE PAST LAST ADR.
4075 021242 101006 BHI 4S ;BR IF ENOUGH MEMORY.
4076 021244 004567 002242 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4077 021250 026737 .WORD NOMTST ;ADDRESS OF MESSAGE TO BE TYPED
4078 ;"NO MEMORY TESTED"
4079 021252 016716 160366 MOV .TST32, (SP) ;ADJUST RETURN ADR FOR ABORT.
4080 021256 000207 RTF PC ;ABORT.
4081 021260 016767 160310 160310 4S: MOV LSTADR, TMAPLAD ;GET LAST ADDRESS.
4082 021266 016767 160242 160234 MOV SAVTST, TSTMAP ;GET TEST MAP, LO 64K.
4083 021274 016767 160236 160230 MOV SAVTST+2, TSTMAP+2 ;...HI 64K.
4084 021302 046767 157274 160220 BIC PRGMAP, TSTMAP ;DON'T TEST OVER THE PROGRAM.
4085 021310 046767 157270 160214 BIC PRGMAP+2, TSTMAP+2
4086 021316 056767 157670 TS $PASS ;CHECK FOR FIRST PASS
4087 021322 031011 BNE 10S ;BR IF NOT FIRST PASS.
4088 021324 032767 000003 160176 BIT #3, TSTMAP ;CHECK IF FIRST TWO BANKS AVAILABLE.
4089 021332 001405 BEQ 10S ;NOT TESTING FIRST 2 BANKS.
4090 021334 042767 177774 160166 BIC #177774, TSTMAP ;CLR ALL BUT FIRST 2 BANKS.
4091 021342 005067 160164 CLR TSTMAP+2
4092 021346 005774 10S: TST R4 ;CHECK FOR A MINIMUM BLOCK SIZE.
4093 021350 001503 BEQ 20S ;BR IF NO MIN BLOCK SIZE.
4094 021352 030467 160206 BIT R4, TMPFAD ;CHECK IF FIRST ADR ON BLOCK BOUNDARY.
4095 021356 001416 BEQ 11S ;BR IF FIRST ADR ON BLOCK BOUNDARY.
4096 021360 050467 160200 BIC R4, TMPFAD ;ADJUST FIRST ADR TO END OF BLOCK.
4097 021364 005267 160174 INC TMPFAD ;FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4098 021370 032767 017777 160166 BIT #MASK4K, TMPFAD ;CHECK IF FIRST ADR REACHED 4K BOUNDARY.
4099 021376 001006 BNE 11S ;BR IF NOT ON 4K BOUNDARY.
4100 021400 046767 160164 160122 BIC FADMAP, TSTMAP ;DON'T TEST FIRST BANK.
4101 021406 046767 160160 160116 BIC FADMAP+2, TSTMAP+2
4102 021414 030467 160156 11S: BIT R4, TMAPLAD ;CHECK IF LAST ADR ON BLOCK BOUNDARY.
4103 021420 001414 BEQ 12S ;BR IF ON BLOCK BOUNDARY.
4104 021422 030467 160150 BIC R4, TMAPLAD ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDARY.
4105 021426 032767 017777 160142 BIT #MASK4K, TMAPLAD ;CHECK IF ADJUSTED TO 4K BOUNDARY.
4106 021434 001006 BNE 12S ;BR IF NOT ON 4K BOUNDARY.
4107 021436 046767 160140 160064 BIC LADMAP, TSTMAP ;SKIP TESTING LAST BANK.
4108 021444 046767 160134 160060 BIC LADMAP+2, TSTMAP+2
4109 021452 036767 160112 *50122 12S: BIT FADMAP, LADMAP ;CHECK IF FIRST AND LAST IN SAME BANK.
4110 021460 001004 BNE 13S ;BR IF IN SAME BANK.
4111 021462 036767 160104 160114 BIT FADMAP+2, LADMAP+2 ;...UPPER 64K.
4112 021470 001404 BEQ 14S ;BR IF FIRST AND LAST NOT SAME BANK.
4113 021472 026767 160100 160064 13S: CMP TMAPLAD, TMPFAD ;CHECK IF ANY MEMORY LEFT.
4114 021500 001406 BLOS 15S ;BR IF NO MEMORY TO TEST.
4115 021502 056767 160022 14S: TST TSTMAP ;CHECK IF ANY BANKS LEFT TO TEST!!
4116 021506 001017 BNE 16S ;BR IF TEST MAP NOT EMPTY.
4117 021510 056767 160016 TST TSTMAP+2 ;CHECK FOR ANY BANKS.
4118 021514 001014 BNE 16S ;BR IF TEST MAP NOT EMPTY.
4119 021516 15S:
4120 021516 004567 001770 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4121 021522 026763 .WORD SKPMES ;ADDRESS OF MESSAGE TO BE TYPED
4122 ;"SKIPPING TEST #"
4123 021524 005046 CLR -(SP) ;CLEAR THE WORD ON THE STACK.
4124 021526 116716 157350 MOV $TSTNM, (SP) ;PUT THE DATA ON THE STACK.

```

```

CZQMCF0 0-124K MEMORY EXERCISER, 16K VER MACY11 30A(1052) 20-FEB-78 07:56 PAGE 85
CZQMCF.P11 14-FEB-78 08:19 SCOPE HANDLER ROUTINE SEQ 0167

4125 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOS ROUTINE
4126 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4127 021532 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4128 021535 004777 003122 JSR PC, $TYPOS ;GO TO THE SUBROUTINE
4129 021542 003 .BYTE 3 ;TYPE 3 DIGITS.
4130 021543 001 .BYTE 1 ;TYPE LEADING ZEROS.
4131 021544 000427 BR ENDINS ;RETURN TO SKIP TEST.
4132 021546 062716 AD #4, (SP) ;SKIP THE SKIP ON RETURN.
4133 021552 062767 000004 157326 ADD #4, $LADR ;ADJUST THE LOOP ADR PAST THE SKIP.
4134 021560 012767 017777 160000 20S: MOV #MASK4K, FADMSK ;GET 4K MASK.
4135 021566 016705 157772 MOV TMPFAD, R5 ;GET FIRST ADR.
4136 021572 040567 157770 21S: BIC R5, FADMSK ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
4137 021576 006305 ASL R5 ;MOVE LOWEST BIT UP ONE.
4138 021600 001374 BNE 21S ;LOOP UNTIL OVERFLOW.
4139 021602 012767 017777 157770 MOV #MASK4K, LADMSK ;SET MASK BITS
4140 021610 016705 157762 MOV TMAPLAD, R5 ;GET LAST ADR.
4141 021614 040567 157760 22S: BIC R5, LADMSK ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4142 021620 006305 ASL R5 ;MOVE LOWEST BIT OF LAST ADR UP ONE.
4143 021622 001374 BNE 22S ;LOOP UNTIL OVERFLOW.
4144 021624 000207 ENDINS: RTS PC ;EXIT SCOPE ROUTINE BACK TO TEST.
4145 021626 000004 SMXCNT: 4 ;MAX. NUMBER OF ITERATIONS
4146 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SCKSWR ROUTINE
4147 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4148 021630 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4149 021634 004767 000524 JSR PC, $CKSWR ;GO TO THE SUBROUTINE
4150 .SBTTL ERROR HANDLER ROUTINE
4151 ;*****
4152 ;* THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4153 ;* SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4154 ;* AND GO TO $ERRTPY ON ERROR
4155 ;* THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4156 ;* SW1=1 HALT ON ERROR
4157 ;* SW13=1 INHIBIT ERROR TYPEDUTS
4158 ;* SW10=1 BELL ON ERROR
4159 ;* SW09=1 LOOP ON ERROR
4160 ;* CALL
4161 ;* ERROR N ;:ERROR=EMT AND N=ERROR ITEM NUMBER
4162
4163
4164 021640 ERROR:
4165 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SCKSWR ROUTINE
4166 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4167 021640 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4168 021644 004767 000514 JSR PC, $CKSWR ;GO TO THE SUBROUTINE
4169 021650 062716 000002 ADD #2, (SP) ;ADJUST POINTER PAST CODE WORD.
4170 021654 105267 157223 7S: INCB $ERFLG ;SET THE ERROR FLAG
4171 021660 001775 BEQ 7S ;DON'T LET THE FLAG GO TO ZERO
4172 021662 016777 157214 157252 MOV $TSTNM, @DISP ;DISPLAY TEST NUMBER AND ERROR FLAG
4173 021670 032777 002000 157242 BIT #BIT0, @SWR ;BELL ON ERROR?
4174 021676 001403 BEQ 1S ;NO - SKIP
4175 021700 004567 001606 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4176 021704 001174 .WORD $BELL ;ADDRESS OF MESSAGE TO BE TYPED
4177 021706 005267 157200 1S: INC $ERTTL ;COUNT THE NUMBER OF ERRORS
4178 021712 011667 157200 MOV (SP), $ERRPC ;GET ADDRESS OF ERROR INSTRUCTION
4179 021716 162767 000002 157172 SUB #2, $ERRPC
4180 021724 117767 157166 157162 MOV @$ERRPC, $ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE

```

```

4181 021732 032777 020000 157200 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
4182 021740 0C1005 BNE 20$ ;;SKIP TYPEOUTS
4183 021742 0C4767 200116 JSR PC,SERRTYP ;;GO TO USEP ERROR ROUTINE
4184 021746 0C4567 001540 JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4185 021752 0C1201 .WORD $CRLF ;;ADDRESS OF MESSAGE TO BE TYPED
4186 021754 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
4187 021754 122767 000001 157242 BNE 25 ;;NO SKIP APT ERROR REPORT
4188 021762 001007 MOV#B $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
4189 021764 118767 157124 000004 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
4190 021772 0C4767 002044 21$: .BYTE 0
4191 021776 000 .BYTE 0
4192 021777 000 22$: BR 22$ ;;APT ERROR LOOP
4193 022000 0C0777 25: TST @SWR ;;HALT ON ERROR
4194 022002 0C5777 157110 BPL 3$ ;;SKIP IF CONTINUE
4195 022006 100005 HALT ;;HALT ON ERROR!
4196 022010 000000
4197
4198
4199 022012 013746 177776
4200 022016 0C4767 000342
4201 022022 032777 001000 157110 3$: BIT @#PSW, -(SP) ;;PUT THE PROCESSOR STATUS ON THE STACK
4202 022030 001402 JSR PC, $CKSWR ;;GO TO THE SUBROUTINE
4203 022032 016716 157052 BEQ 4$ ;;LOOP ON ERROR SWITCH SET?
4204 022036 0C5767 157130 MOV $LPERR,(SP) ;;BR IF NO
4205 022042 001402 45: TST $ESCAPE ;;FUJGE RETURN FOR LOOPING
4206 022044 016716 157122 BEQ 5$ ;;CHECK FOR AN ESCAPE ADDRESS
4207 022050 MOV $ESCAPE,(SP) ;;BR IF NONE
4208 022050 022737 014222 000042 55: CMP #SENDAD,@#42 ;;FUJGE RETURN ADDRESS FOR ESCAPE
4209 022056 001001 BNE 6$ ;;ACT-11 AUTO-ACCEPT?
4210 022060 000000 HALT ;;BRANCH IF NO
4211 022062 6$: RTS PC ;;YES
4212 022062 0C0207
4213
4214
4215 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
4216
4217
4218
4219
4220
4221 022064
4222 022064 0C4567 001422
4223 022070 001201
4224 022072 0100.5
4225 022074 0C5000
4226 022076 156700 157012
4227 022102 001007
4228
4229 022104 016746 157006
4230
4231
4232
4233 022110 013746 177776
4234 022114 0C4767 002570
4235 022120 0C00513
4236 022122 016767 156770 157364 1$: MOV $ERRPC, $VERPC ;;SET UP VIRTUAL PC FOR TYPEOUT.
    
```

```

4237 022130 166767 156444 157356 SUB RELOCF, $VERPC ;;MAKE VIRTUAL IF NOT ALREADY.
4238 022136 006300 RDC R0 ;;ADJUST THE INDEX SO THAT IT WILL
4239 022140 0C6300 ASL R0 ; WORK FOR THE ERROR TABLE
4240 022142 0C6300 ASL R0
4241 022144 0C6300 ASL R0
4242 022146 066700 157466 ADD .ERRTB, R0 ;;FORM TABLE POINTER
4243 022152 012067 000006 MOV (R0)+,2$ ;;PICKUP "ERROR MESSAGE" POINTER
4244 022156 001406 BLJ 3$ ;;SKIP TYPEOUT IF NO POINTER
4245 022160 0C4567 001326 JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4246 022164 0C0000 25: .WORD 0 ;;"ERROR MESSAGE" POINTER GOES HERE
4247 022166 0C4567 001320 JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4248 022172 001201 .WORD $CRLF ;;ADDRESS OF MESSAGE TO BE TYPED
4249 022174 012067 000006 3$: MOV (R0)+,4$ ;;PICKUP "DATA HEADER" POINTER
4250 022200 001406 BEQ 5$ ;;SKIP TYPEOUT IF 0
4251 022202 0C4567 001304 JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4252 022206 000000 .WORD 0 ;;"DATA HEADER" POINTER GOES HERE
4253 022210 0C4567 001276 JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4254 022214 0C1201 .WORD $CRLF ;;ADDRESS OF MESSAGE TO BE TYPED
4255 022216 010146 55: MOV R1,-(SP) ;;SAVE R1
4256 022220 012001 MOV (R0)+,R1 ;;PICKUP "DATA TABLE" POINTER
4257 022222 001451 BEQ 9$ ;;BR IF NO DATA TO BE TYPED
4258 022224 066701 156.50 ADD RELOCF, R1 ;;ADJUST POINTER
4259 022230 012000 MOV (R0)+,R0 ;;PICKUP "DATA FORMAT" POINTER
4260 022232 066700 156342 ADD RELOCF, R0 ;;ADJUST POINTER.
4261 022236 105720 6$: TSTB (R0)+ ;;CHECK THE FORMAT
4262 022240 001006 BNE 7$ ;;BR IF NOT 16-BIT OCTAL
4263 022242 013146 MOV @ (R1)+,-(SP) ;;SAVE @ (R1)+ FOR TYPEOUT
4264
4265
4266 022244 013746 177776
4267 022250 0C4767 002434
4268 022254 0C0426
4269 022256 100406
4270 022260 013146
4271
4272
4273 022262 013746 177776
4274 022266 0C4767 002140
4275 022272 0C0417
4276 022274 122760 177777 177777 17$: CMPB #-1, -(R0) ;;CHECK FOR 18-BIT ADDRESS FORMAT.
4277 022302 001004 BNE 18$ ;;BR IF NOT 18-BIT ADDRESS FORMAT.
4278 022304 013146 MOV @ (R1)+,-(SP) ;;PUT THE DATA ON THE STACK.
4279 022306 0C4767 002640 JSR PC, $STPAD ;;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4280 022312 0C0407 BR 8$ ;;SKIP
4281 022314
4282 022314 0C05046 18$: CLR -(SP) ;;CLEAR THE WORD ON THE STACK.
4283 022316 113116 MOV#B @ (R1)+, (SP) ;;PUT THE DATA ON THE STACK.
4284
4285
4286 022320 013746 177776
4287 022324 0C4767 002334
4288 022330 003
4289 022331 001
4290 022332 0C5711
4291 022334 001404 85: TST (R1) ;;IS THERE ANOTHER NUMBER?
4292 022336 0C4567 001150 BEQ 9$ ;;BR IF NO
4293 022336 0C4567 001150 JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
    
```

```

4293 022342 022362 .WORD 11$ ;ADDRESS OF MESSAGE TO BE TYPED
4294 022344 020734 BR 6$ ;LOOP
4295
4296 022346 012601 95: MOV (SP)+,R1 ;RESTORE R1
4297 022350 012600 105: MOV (SP)+,R0 ;RESTORE R0
4298 022352 004567 001134 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4299 022356 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4300 022360 000207 RTS PC ;RETURN
4301 022362 000011 11$: .ASCIZ / / ;TAB CHARACTER.
4302 .EVEN
4303 .SBTTL TTY INPUT ROUTINE
4304
4305 ;*****
4306 .ENABL LSB
4307
4308 ;*****
4309 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4310 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4311 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4312 ;*WHEN OPERATING IN TTY FLAG MODE.
4313 022364 022767 000176 156546 SCKSWR: CMP #SWREG,SWR ;IS THE SOFT-SWR SELECTED?
4314 022372 021104 BNE 15$ ;BRANCH IF NO
4315 022374 105777 156544 TSTB @STKS ;CHAR THERE?
4316 022400 100101 BPL 15$ ;IF NO, DON'T WAIT AROUND
4317 022402 117746 156540 MOVB @STKB,-(SP) ;SAVE THE CHAR
4318 022406 042716 177600 BIC #'C177,(SP) ;STRIP-OFF THE ASCII
4319 022412 022726 000007 CMP #7,(SP)+ ;IS IT A CONTROL G?
4320 022416 001072 BNE 15$ ;NO, RETURN TO USER
4321 022420 126727 156510 000001 CMPB SAUTDB,#1 ;ARE WE RUNNING IN AUTO-MODE?
4322 022426 001466 BEQ 15$ ;BRANCH IF YES
4323
4324 022430 004567 001056 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4325 022434 025311 .WORD SCNTLG ;ADDRESS OF MESSAGE TO BE TYPED
4326 022436
4327 022438 004567 001050 SGTSWR: JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4328 022442 023316 .WORD $MSWR ;ADDRESS OF MESSAGE TO BE TYPED
4329 022444 016746 155526 MOV SWREG,-(SP) ;SAVE SWREG FOR TYPEOUT
4330
4331 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOC ROUTINE
4332 ;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC*.
4333 022450 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4334 022454 004767 002230 JSR PC, STYPOC ;GO TO THE SUBROUTINE
4335 022460 004567 001026 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4336 022464 023327 .WORD $MNEW ;ADDRESS OF MESSAGE TO BE TYPED
4337 022466 005003 195: CLR -(SP) ;CLEAR COUNTER
4338 022470 005046 CLR -(SP) ;THE NEW SWR
4339 022472 105777 156446 75: TSTB @STKS ;CHAR THERE?
4340 022476 100375 BPL 7$ ;IF NOT TRY AGAIN
4341
4342 022500 117746 156442 MOVB @STKB,-(SP) ;PICK UP CHAR
4343 022504 042716 177600 BIC #'C177,(SP) ;MAKE IT 7-BIT ASCII
4344
4345
4346 022510 021627 000025 95: CMP (SP),#25 ;IS IT A CONTROL-U?
4347 022514 001006 BNE 105 ;BRANCH IF NOT
4348 022516 004567 000770 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
    
```

```

4349 022522 023304 .WORD SCNTLU ;ADDRESS OF MESSAGE TO BE TYPED
4350 022524 062706 000006 205: ADD #6,SP ;IGNORE PREVIOUS INPUT
4351 022530 000756 BR 19$ ;LET'S TRY IT AGAIN
4352
4353
4354 022532 021627 000015 105: CMP (SP),#15 ;IS IT A <CR>?
4355 022536 001023 BNE 16$ ;BRANCH IF NO
4356 022540 005766 000004 TST 4(SP) ;YES, IS IT THE FIRST CHAR?
4357 022544 001403 BEQ 115$ ;BRANCH IF YES
4358 022546 016677 000002 156364 MOV 2(SP),@SWR ;SAVE NEW SWR
4359 022554 062706 000006 115: ADD #6,SP ;CLEAR UP STACK
4360 022560 145:
4361 022560 004567 000726 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4362 022564 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4363 022566 126727 156343 000001 CMPB $INTAG,#1 ;RE-ENABLE TTY KBD INTERRUPTS?
4364 022574 001003 BNE 15$ ;BRANCH IF NOT
4365 022576 012777 000100 156340 MOV #100,@STKS ;RE-ENABLE TTY KBD INTERRUPTS
4366 022604 000302 RTN ;RETURN
4367 022606 004767 001142 155: JSR PC,STYPOC ;ECHO CHAR
4368 022612 021627 000060 165: CMP (SP),#60 ;CHAR < 0?
4369 022616 002420 BLT 18$ ;BRANCH IF YES
4370 022620 021627 000067 CMP (SP),#57 ;CHAR > 7?
4371 022624 003015 BGT 18$ ;BRANCH IF YES
4372 022626 042716 000060 BIC #60,(SP)+ ;STRIP-OFF ASCII
4373 022632 005766 000002 TST 2(SP) ;IS THIS THE FIRST CHAR
4374 022636 001403 BEQ 17$ ;BRANCH IF YES
4375 022640 006316 ASL (SP) ;NO, SHIFT PRESENT
4376 022642 006316 ASL (SP) ;CHAR OVER TO MAKE
4377 022644 006316 ASL (SP) ;ROOM FOR NEW ONE.
4378 022646 005266 000002 175: INC 2(SP) ;KEEP COUNT OF CHAR
4379 022652 006616 177776 BIS -2(SP),(SP) ;SET IN NEW CHAR
4380 022656 000705 BR 7$ ;GET THE NEXT ONE
4381 022660
4382 022660 004567 000626 185: JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4383 022664 001200 .WORD $QUES ;ADDRESS OF MESSAGE TO BE TYPED
4384 022666 000716 BR 20$ ;SIMULATE CONTROL-U
4385
4386
4387
4388 ;*****
4389 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4390 ;*CALL:
4391 ;* RDCHR ;INPUT A SINGLE CHARACTER FROM THE TTY
4392 ;* RETURN HERE ;CHARACTER IS ON THE STACK
4393 ;* ;WITH PARITY BIT STRIPPED OFF
4394 ;
4395
4396 022670 011646 SRDCHR: MOV (SP),-(SP) ;PUSH DOWN THE PC
4397 022672 016566 MOV 4(SP),2(SP) ;SAVE THE PS
4398 022700 105777 156240 15: TSTB @STKS ;WAIT FOR NEW ONE.
4399 022704 100375 BPL 15 ;A CHARACTER
4400 022706 117766 156234 000004 MCVB @STKB,4(SP) ;READ THE TTY
4401 022714 042766 177600 000004 BIC #'C<177>,4(SP) ;GET RID OF JUNK IF ANY
4402 022722 026627 000004 000023 CMP 4(SP),#23 ;IS IT A CONTROL-S?
4403 022730 001013 BNE 3$ ;BRANCH IF NO
4404 022732 105777 156206 25: TSTB @STKS ;WAIT FOR A CHARACTER
    
```

```
4405 022735 100375 BPL 2S ;;LOOP UNTIL ITS THERE
4406 022740 117745 MOV8 @STKB,-(SP) ;;GET CHARACTER
4407 022744 042716 BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII
4408 022750 022677 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
4409 022754 0C1356 BNE 2S ;;IF NOT DISCARD IT
4410 022756 0C0750 BR 1S ;;YES, RESUME
4411 022760 026627 000004 000140 3S: CMP 4(SP),#140 ;;IS IT UPPER CASE?
4412 022766 0C2407 BI - 4S ;;BRANCH IF YES
4413 022770 026627 000004 000175 CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
4414 022776 0C3003 BGT 4S ;;BRANCH IF YES
4415 023000 027666 000004 000004 BIC #40,4(SP) ;;MAKE IT UPPER CASE
4416 023006 000002 4S: RTI ;;GO BACK TO USER
4417 ;;*****
4418 ;;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4419 ;;CALL:
4420 ;; RDLIN ;;INPUT A STRING FROM THE TTY
4421 ;; RETURN HERE ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4422 ;; ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4423
4424 023010 010346 SRDLIN: MOV R3,-(SP) ;;SAVE R3
4425 023012 0C5046 CLR -(SP) ;;CLEAR THE RUBOUT KEY
4426 023014 012763 023774 1S: MOV #STTYIN,R3 ;;GET ADDRESS
4427 023020 022703 023304 2S: CMP #STTYIN+8.,R3 ;;BUFFER FULL?
4428 023024 101467 BLOS 4S ;;BR IF YES
4429
4430 ;;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SRDCHR ROUTINE
4431 ;;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
4431 023026 013746 177776 MOV @#PSW, -(SP) ;;PUT THE PROCESSOR STATUS ON THE STACK
4432 023032 047677 177632 JSR PC, SRDCHR ;;GO TO THE SUBROUTINE
4433 023036 112613 MOV8 (SP)+,(R3) ;;GET CHARACTER
4434 023040 122713 000177 10S: CMPB #177,(R3) ;;IS IT A RUBOUT
4435 023044 0C1024 BNE 5S ;;BR IF NO
4436 023046 005716 TST (SP) ;;IS THIS THE FIRST RUBOUT?
4437 023050 010100 BNE 5S ;;BR IF NO
4438 023052 112767 000171 000212 MOV8 #'\,9S ;;TYPE A BACK SLASH
4439 023060 004567 000426 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4440 023064 023272 .WORD 9S ;;ADDRESS OF MESSAGE TO BE TYPED
4441 023066 012716 177777 MCV #-1,(SP) ;;SET THE RUBOUT KEY
4442 023072 0C53C3 6S: DEC R3 ;;BACKUP BY ONE
4443 023074 023272 023274 CMP R3,#STTYIN ;;STACK EMPTY?
4444 023100 1C3441 SLO 4S ;;BR IF YES
4445 023102 111367 000164 MOV8 (R3),9S ;;SETUP TO TYPEOUT THE DELETED CHAR.
4446 023106 004567 000400 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4447 023112 023272 9S .WORD 9S ;;ADDRESS OF MESSAGE TO BE TYPED
4448 023114 0C0741 BR 2S ;;GO READ ANOTHER CHAR.
4449 023116 005716 5S: TST (SP) ;;RUBOUT KEY SET?
4450 023120 001407 BEQ 7S ;;BR IF NO
4451 023122 112767 000134 000142 MCVB #'\,9S ;;TYPE A BACK SLASH
4452 023130 004567 000356 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4453 023134 023272 .WORD 9S ;;ADDRESS OF MESSAGE TO BE TYPED
4454 023136 0C5016 CLR (SP) ;;CLEAR THE RUBOUT KEY
4455 023140 122713 000025 7S: CMPB #25,(R3) ;;IS CHARACTER A CTRL U?
4456 023144 0010C4 BNE 7S ;;BR IF NO
4457 023146 004567 000340 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4458 023152 0233C4 .WORD SCNTLU ;;ADDRESS OF MESSAGE TO BE TYPED
4459 023154 000777 BR 1S ;;GO START OVER
4460 023156 122713 8S: CMPL #22,(R3) ;;IS CHARACTER A "R"?
```

```
4461 023162 001014 BNE 3S ;;BRANCH IF NO
4462 023164 1C5013 CLR8 (R3) ;;CLEAR THE CHARACTER
4463 023166 0C4567 000320 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4464 023172 001201 .WORD SCRLF ;;ADDRESS OF MESSAGE TO BE TYPED
4465 023174 0C4567 000312 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4466 023200 023274 .WORD STTYIN ;;ADDRESS OF MESSAGE TO BE TYPED
4467 023202 0C0706 BR 2S ;;GO PICKUP ANOTHER CHARACTER
4468 023204 4S:
4469 023204 0C4567 000302 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4470 023210 001200 .WORD $QUES ;;ADDRESS OF MESSAGE TO BE TYPED
4471 023212 0C0700 BR 1S ;;CLEAR THE BUFFER AND LOOP
4472 023214 111367 000052 3S: MOV8 (R3),9S ;;ECHO THE CHARACTER
4473 023220 0C4567 000266 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4474 023224 0C3272 .WORD 9S ;;ADDRESS OF MESSAGE TO BE TYPED
4475 023226 122723 000015 CMPR #15,(R3)+ ;;CHECK FOR RETURN
4476 023232 0C1272 BNE 2S ;;LOOP IF NOT RETURN
4477 023234 105063 177777 CLR8 -(R3) ;;CLEAR RETURN (THE 15)
4478 023240 0C4567 000246 JET R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4479 023244 0012C2 .WORD SLF ;;ADDRESS OF MESSAGE TO BE TYPED
4480 023246 005726 TST (SP)+ ;;CLEAN RUBOUT KEY FROM THE STACK
4481 023250 012603 MOV (SP)+,R3 ;;RESTORE R3
4482 023252 011646 MOV (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
4483 023254 016566 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
4484 023262 0127C3 023274 000004 MOV #STTYIN,4(SP)
4485 023270 000002 RTI ;;RETURN
4486 023272 0C0 .BYTE 0 ;;STORAGE FOR ASCII CHAR. TO TYPE
4487 023274 0C0 .BYTE 0 ;;TERMINATOR
4488 023274 000010 $TTYIN: .LLKB 8. ;;RESERVE 8 BYTES FOR TTY INPUT
4489 023304 052536 005015 C00 $CNTLU: .ASCIZ /?U/<15><12> ;;CONTROL "U"
4490 023311 136 006507 000012 $CNTIG: .ASCIZ /?G/<15><12> ;;CONTROL "G"
4491 023316 0 5015 053523 020122 $MSWB: .ASCIZ <15><12>/SWR = /
4492 023324 0C0075 000 SMNEW: .ASCIZ / NEW = /
4493 023327 040 047040 053505
4494 023334 036440
4495 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
4496
4497 ;;*****
4498 ;;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
4499 ;;CHANGE IT TO BINARY.
4500 ;;THE INPUT CHARACTERS WILL BE CHECKED TO INSURE THEY ARE LEGAL
4501 ;;OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
4502 ;;FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
4503 ;;THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
4504 ;;CALL:
4505 ;; RDOCT ;;READ AN OCTAL NUMBER
4506 ;; RETURN HERE ;;LOW ORDER BITS ARE ON TOP OF THE STACK
4507 ;; ;;HIGH ORDER BITS ARE IN SHIOCT
4508
4509 023340 011646 SRDOCT: MOV (SP),-(SP) ;;PROVIDE SPACE FOR THE
4510 023342 016566 000004 000002 MOV 4(SP),2(SP) ;;INPUT NUMBER
4511 023350 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
4512 023352 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
4513 023354 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
4514 1S:
4515 ;;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SRDLIN ROUTINE
4516 ;;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
```



```
4517 023356 013746 177776      MOV    @#PSW, -(SP)  ;PUT THE PROCESSOR STATUS ON THE STACK
4518 023352 004767 177422      JSR    PC, $RDLIN  ;GO TO THE SUBROUTINE
4519 023366 012600      MOV    (SP)+,RO     ;GET ADDRESS OF 1ST CHARACTER
4520 023370 0100..7 000102      MOV    RO,55        ;AND SAVE IT
4521 023374 005001      CLR    R1           ;CLEAR DATA WORD
4522 023376 005002      CLR    R2
4523 023400 112046      2$:   MOVB   (RO)+,-(SP)  ;PICKUP THIS CHARACTER
4524 023402 001420      BLJ    3$          ;IF ZERO GET OUT
4525 023404 122716 000060      CMPB  #'0,(SP)     ;MAKE SURE THIS CHARACTER
4526 023410 003026      BGT    4$          ;IS AN OCTAL DIGIT
4527 023412 112716 000067      CMPB  #'7,(SP)
4528 023416 002423      BLT    4$
4529 023420 006301      ASL   R1           ;:*
4530 023422 006102      ROL   R2           ;:*4
4531 023424 006301      ASL   R1           ;:*4
4532 023426 006102      ROL   R2
4533 023430 006301      ASL   R1           ;:*8
4534 023432 006102      ROL   R2
4535 023434 042716 177770      BIC   #'C7,(SP)    ;STRIP THE ASCII JUNK
4536 023440 002601      ADD   (SP)+,R1     ;ADD IN THIS DIGIT
4537 023442 000756      BR    2$          ;LOOP
4538 023444 005726      3$:   TST    (SP)+     ;CLEAN TERMINATOR FROM STACK
4539 023446 010166 000012      MOV    R1,12(SP)   ;SAVE THE RESULT
4540 023452 010267 000032      MOV    R2,$SHIOCT
4541 023456 012602      MOV    (SP)+,R2    ;POP STACK INTO R2
4542 023460 012601      MOV    (SP)+,R1    ;POP STACK INTO R1
4543 023462 012600      MOV    (SP)+,RO    ;POP STACK INTO RO
4544 023464 000002      RTI
4545 023466 005726      4$:   TST    (SP)+     ;CLEAN PARTIAL FROM STACK
4546 023470 105010      CLR   (RO)         ;SET A TERMINATOR
4547 023472 004567 000014      JSR   R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4548 023476 000000      5$:   .WORD  0
4549 023500 004567 000006      JSR   R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4550 023504 001200      .WORD  $QUES       ;ADDRESS OF MESSAGE TO BE TYPED
4551 023506 000723      BR    5$          ;TRY AGAIN
4552 023510 000000      SHIOCT: .WORD  0   ;HIGH ORDER BITS GO HERE
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563 023524 013746 177776      SPRINT: MOV (R5)+, 1$ ;GET THE MESSAGE VIRTUAL ADDRESS.
4564 023530 004767 000004      ADD  RELOC, 1$    ;MAKE IT PHYSICAL.
4565 023534 000000      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
4566 023536 000205      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585 023540 105767 155413      1$:   .TYPE: TSTB  $TPFLG ;IS THERE A TERMINAL?
4586 023544 100002      BPL   1$          ;BR IF YES
4587 023546 000000      HALT ;HALT HERE IF NO TERMINAL
4588 023550 000430      BR    3$          ;LEAVE
4589 023552 010046      1$:   MOV    RO,-(SP)   ;SAVE RO
4590 023554 017600 000002      MC    @2(SP),RO   ;GET ADDRESS OF ASCII STRING
4591 023560 122767 000001 155436      CMPB  #APTENV,$ENV ;RUNNING IN APT MODE
4592 023566 001011      BNE   62$         ;NO, GO CHECK FOR APT CONSOLE
4593 023570 102767 000100 155427      BITB  #APTSPOOL,$ENVM ;SPOOL MESSAGE TO APT
4594 023576 001425      BEQ   62$         ;NO, GO CHECK FOR CONSOLE
4595 023600 010067 000004      MOV   RO,61$     ;SETUP MESSAGE ADDRESS FOR APT
4596 023604 004767 000222      JSR   PC,$ATY3   ;SPOOL MESSAGE TO APT
4597 023610 000000      61$: .WORD  0      ;MESSAGE ADDRESS
4598 023612 122767 000040 155405      62$: BITB  #APTCSUP,$ENVM ;APT CONSOLE SUPPRESSED
4599 023620 001033      BNE   60$         ;YES, SKIP TYPE OUT
4600 023622 112046      2$:   MVB   (RO)+,-(SP) ;PUSH CHARACTER TO BE TYPED UNTO STACK
4601 023624 001005      BNE   4$          ;BR IF IT ISN'T THE TERMINATOR
4602 023626 005726      TST  (SP)+       ;IF TERMINATOR POP IT OFF THE STACK
4603 023630 012600      60$: MOV   (SP)+,RO   ;RESTORE RO
4604 023632 002716 000002      3$:   ADD   #2,(SP)    ;ADJUST RETURN PC
4605 023636 000002      RTI
4606 023640 122716 000011      4$:   CMPB  #'HT,(SP)  ;BRANCH IF <HT>
4607 023644 001431      BEQ   5$
4608 023646 122716 000200      CMPB  #'CRLF,(SP) ;BRANCH IF NOT <CRLF>
4609 023652 001007      BNE   5$
4610 023654 005726      TST  (SP)+       ;POP <CR><LF> EQUIV
4611 023656 004567 177630      JSR   R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4612 023662 001201      SCLRF
4613 023664 105367 000130      CLR   $CHARCNT   ;CLEAR CHARACTER COUNT
4614 023670 000754      BR    2$         ;GET NEXT CHARACTER
4615 023672 004767 000056      5$:   JSR   PC,$TYPEC  ;GO TYPE THIS CHARACTER
4616 023676 126726 155254      65$: CMPB  $FILLC,(SP)+ ;IS IT TIME FOR FILLER CHARS.?
4617 023702 001347      BNE   2$         ;IF NO GO GET NEXT CHAR.
4618 023704 016746 155244      MC   $NULL,-(SP) ;GET # OF FILLER CHARS. NEEDED
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628 023730 112716 000040      7$:   DECB  1(SP)     ;DOES A NULL NEED TO BE TYPED?
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
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
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
```

```
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585 023540 105767 155413      $TYPE: TSTB  $TPFLG ;IS THERE A TERMINAL?
4586 023544 100002      BPL   1$          ;BR IF YES
4587 023546 000000      HALT ;HALT HERE IF NO TERMINAL
4588 023550 000430      BR    3$          ;LEAVE
4589 023552 010046      1$:   MOV    RO,-(SP)   ;SAVE RO
4590 023554 017600 000002      MC    @2(SP),RO   ;GET ADDRESS OF ASCII STRING
4591 023560 122767 000001 155436      CMPB  #APTENV,$ENV ;RUNNING IN APT MODE
4592 023566 001011      BNE   62$         ;NO, GO CHECK FOR APT CONSOLE
4593 023570 102767 000100 155427      BITB  #APTSPOOL,$ENVM ;SPOOL MESSAGE TO APT
4594 023576 001425      BEQ   62$         ;NO, GO CHECK FOR CONSOLE
4595 023600 010067 000004      MOV   RO,61$     ;SETUP MESSAGE ADDRESS FOR APT
4596 023604 004767 000222      JSR   PC,$ATY3   ;SPOOL MESSAGE TO APT
4597 023610 000000      61$: .WORD  0      ;MESSAGE ADDRESS
4598 023612 122767 000040 155405      62$: BITB  #APTCSUP,$ENVM ;APT CONSOLE SUPPRESSED
4599 023620 001033      BNE   60$         ;YES, SKIP TYPE OUT
4600 023622 112046      2$:   MVB   (RO)+,-(SP) ;PUSH CHARACTER TO BE TYPED UNTO STACK
4601 023624 001005      BNE   4$          ;BR IF IT ISN'T THE TERMINATOR
4602 023626 005726      TST  (SP)+       ;IF TERMINATOR POP IT OFF THE STACK
4603 023630 012600      60$: MOV   (SP)+,RO   ;RESTORE RO
4604 023632 002716 000002      3$:   ADD   #2,(SP)    ;ADJUST RETURN PC
4605 023636 000002      RTI
4606 023640 122716 000011      4$:   CMPB  #'HT,(SP)  ;BRANCH IF <HT>
4607 023644 001431      BEQ   5$
4608 023646 122716 000200      CMPB  #'CRLF,(SP) ;BRANCH IF NOT <CRLF>
4609 023652 001007      BNE   5$
4610 023654 005726      TST  (SP)+       ;POP <CR><LF> EQUIV
4611 023656 004567 177630      JSR   R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4612 023662 001201      SCLRF
4613 023664 105367 000130      CLR   $CHARCNT   ;CLEAR CHARACTER COUNT
4614 023670 000754      BR    2$         ;GET NEXT CHARACTER
4615 023672 004767 000056      5$:   JSR   PC,$TYPEC  ;GO TYPE THIS CHARACTER
4616 023676 126726 155254      65$: CMPB  $FILLC,(SP)+ ;IS IT TIME FOR FILLER CHARS.?
4617 023702 001347      BNE   2$         ;IF NO GO GET NEXT CHAR.
4618 023704 016746 155244      MC   $NULL,-(SP) ;GET # OF FILLER CHARS. NEEDED
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628 023730 112716 000040      7$:   DECB  1(SP)     ;DOES A NULL NEED TO BE TYPED?
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
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
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
```

```
4629 023734 004767 000014 9S: JSR PC,STYPEC ;;TYPE A SPACE
4630 023740 122767 000007 000052 BITB #7,SCHARCNT ;;BRANCH IF NOT AT
4631 023746 001372 BNE 9S ;;TAB STOP
4632 023750 005726 TST (SP)+ ;;POP SPACE OFF STACK
4633 023752 000773 BR 25 ;;GET NEXT CHARACTER
4634 023754 105777 155170 $TYPEC: TSTB @STPB ;;WAIT UNTIL PRINTER IS READY
4635 023760 100375 BPL $TYPEC
4636 023762 116677 000002 155162 MOVB 2(SP),STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4637 023770 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
4638 023776 001003 BNE 1S ;;BRANCH IF NO
4639 024000 105067 000014 CLR B SCHARCNT ;;YES--CLEAR CHARACTER COUNT
4640 024004 000406 BR $TYPEX ;;EXI:
4641 024005 122766 000012 000002 1S: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
4642 024014 001402 BEQ $TYPEX ;;BRANCH IF YES
4643 024016 105227 INCB (PC)+ ;;COUNT THE CHARACTER
4644 024020 000000 SCHARCNT: WORD 0 ;;CHARACTER COUNT STORAGE
4645 024022 000207 $TYPEX: RTS PC
4646
4647 .SBTTL APT COMMUNICATIONS ROUTINE
4648
4649 *****
4650 024024 112767 000001 000376 SATY1: MOVB #1,SFFLG ;;TO REPORT FATAL ERROR
4651 024032 112767 000001 000366 SATY3: MOVB #1,SMFLG ;;TO TYPE A MESSAGE
4652 024040 0004C3 BR SATYC
4653 024042 112767 000001 000360 SATY4: MOVB #1,SFFLG ;;TO ONLY REPORT FATAL ERROR
4654 024050 S:TYC:
4655 024050 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
4656 024052 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
4657 024054 105767 000346 TSTB SMFLG ;;SHOULD TYPE A MESSAGE?
4658 024060 001450 BEQ 5S ;;IF NOT: BR
4659 024062 1.2767 000001 155134 CMPB #APTENV,SENV ;;OPERATING UNDER APT?
4660 024070 001031 BNE 3S ;;IF NOT: BR
4661 024072 132767 000100 155125 BITB #APTSPOOL,SFNVM ;;SHOULD SPCDL MESSAGES?
4662 024100 001425 BEQ 3S ;;IF NOT: BR
4663 024102 017660 000004 MOVB @4(SP),RO ;;GET MESSAGE ADDR.
4664 024106 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4665 024114 005767 155064 1S: TST SMSGTYP ;;SEE IF DONE W/ LAST XMISSION?
4666 024120 001375 BNE 1S ;;IF NOT: WAIT
4667 024122 010067 155072 MOV RO,SMSGAD ;;PUT ADDR IN MAILBOX
4668 024126 105720 2S: TSTB (RO)+ ;;FIND END OF MESSAGE
4669 024130 001376 BNE 2S
4670 024132 166700 155062 SUB SMSGAD,RO ;;SUB START OF MESSAGE
4671 024136 006200 ASR RO ;;GET MESSAGE LGTH IN WORDS
4672 024140 010067 155056 MOV RO,SMSGLTG ;;PUT LENGTH IN MAILBOX
4673 024144 012767 000004 155032 MOVB #4,SMSGTYP ;;TELL APT TO TAKE MSG.
4674 024152 000413 BR 5S
4675 024154 017667 000004 000016 3S: MOV @4(SP),4S ;;PUT MSG ADDR IN JSR LINKAGE
4676 024162 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
4677 024170 016746 153602 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
4678 024174 004767 177340 JSR PC,$TYPE ;;CALL TYPE MACRO
4679 024200 000000 4S: .WORD 0
4680 024202 5S:
4681 024202 105767 000221 TSTB SLFLG ;;SHOULD LOG AN ERROR?
4682 024206 001422 BEQ 10S ;;IF NOT: BR
4683 024210 017667 000004 MOVB @4(SP),RO ;;GET ERROR #
4684 024214 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
```

```
4685 024222 012701 001344 MOV #SASTAT,R1 ;;POINT TO TABLE START
4686 024226 005711 6S: TST (R1) ;;END OF TABLE?
4687 024230 100404 BMI 8S ;;IF SO: BP
4688 024232 0200C1 CMP RO,(R1)+ ;;PROPER ENTRY?
4689 024234 001406 BEQ 9S ;;IF SO: BR
4690 024236 005721 TST (R1)+ ;;MOVE PAST COUNTER WORD
4691 024240 000772 BR 6S ;;KEEP LOOKING
4692 024242 026701 155244 8S: C: SAPTR,R1 ;;TABLE FULL?
4693 024246 001402 BEQ 10S ;;IF SO: BP -- NO MORE ROOM
4694 024250 010021 MOV RO,(R1)+ ;;SET UP NEW ENTRY
4695 024252 005211 9S: INC (R1) ;;BUMP ERROR COUNT
4696 024254 105767 000150 10S: TSTB SFFLG ;;SHOULD REPORT FATAL ERROR?
4697 024260 001416 BEQ 12S ;;IF NOT: BR
4698 024262 005767 154736 TST SENV ;;RUNNING UNDER APT?
4699 024266 001413 BEQ 12S ;;IF NOT: BR
4700 024270 005767 154710 11S: TST SMSGTYP ;;FINISHED LAST MESSAGE?
4701 024274 001375 BNE 11S ;;IF NOT: WAIT
4702 024276 017667 000004 154702 MOVB @4(SP),SFATAL ;;GET ERROR #
4703 024304 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4704 024312 005267 154666 INC SMSGTYP ;;TELL APT TO TAKE ERROR
4705 024316 105067 000106 12S: CLR B SFFLG ;;CLEAR FATAL FLAG
4706 024322 105067 000C01 CLR B SLFLG ;;CLEAR LOG FLAG
4707 024326 105067 000074 CLR B SMFLG ;;CLEAR MESSAGE FLAG
4708 024332 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
4709 024334 012600 MOV (SP)+,RO ;;POP STACK INTO RO
4710 024338 000207 RTS PC ;;RETURN
4711 024340 SATY6:
4712 024340 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
4713 024342 016700 155144 MOV SAPTR,RO
4714 024346 1E2700 001344 SUB #SASTAT,RO ;;GET SIZE OF STAT TABLE
4715 024352 005767 154626 1S: TST SMSGTYP ;;SEE IF DONE LAST COMMUNICATION
4716 024356 001375 BNE 1S ;;IF NOT: WAIT
4717 024360 010067 154636 MOV RO,SMSGLG ;;SET MESSAGE LENGTH
4718 024364 012767 001375 154626 MOVB #SASTAT,SMEGAD ;;SET MESSAGE ADDR.
4719 024372 012767 000002 154604 MOV #2,SMSGTY ;;TELL APT TO TAKE STATS.
4720 024400 012600 MCV (SP)+,RO ;;POP STACK INTO RO
4721 024402 000207 RTS PC ;;RETURN
4722 024404 SATY7:
4723 024404 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
4724 024406 012701 001344 MOVB #SASTAT,R1 ;;GET START OF TABLE
4725 024412 005721 1S: TST (R1)+ ;;END OF TABLE?
4726 024414 100402 BMI 2S ;;IF SO: BP
4727 024416 005021 CLR (R1)+ ;;CLEAR ERROR COUNT
4728 024420 000774 BR 1S ;;KEEP CLEARING
4729 024422 2S:
4730 024422 012600 MOV (SP)+,RO ;;POP STACK INTO RO
4731 024424 000207 RTS PC ;;RETURN
4732 024426 00C SMFLG: .BYTE 0 ;;MESSG. FLAG
4733 024427 00C SLFLG: .BYTE 0 ;;LOG FLAG
4734 024430 00C SFFLG: .BYTE 0 ;;FATAL FLAG
4735 .EVEN
4736 024432 APTSIZE=200
4737 000001 APTENV=001
4738 000100 APTSPOOL=100
4739 000000 APTCSUP=040
4740 *****
```

```

4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753 024432
4754 024432 010046
4755 024434 010146
4756 024436 010246
4757 024440 010346
4758 024442 010546
4759 024444 012746 020200
4760 024450 016605 000020
4761 024454 010004
4762 024456 005405
4763 024460 112765 000055 000001
4764 024465 016700 154106
4765 024472 012703 024654
4766 024476 060003
4767 024500 112723 000040
4768 024504 005002
4769 024506 016001 024644
4770 024512 010105
4771 024514 002402
4772 024516 005202
4773 024520 000774
4774 024522 000105
4775 024524 005702
4776 024526 001002
4777 024530 105716
4778 024532 100407
4779 024534 106316
4780 024536 103003
4781 024540 116603 000001 177777
4782 024546 052702 000060
4783 024552 022702 000040
4784 024556 101223
4785 024560 005720
4786 024562 020067 155054
4787 024566 103746
4788 024570 101002
4789 024572 010502
4790 024574 000764
4791 024576 105726
4792 024600 100003
4793 024602 116663 177777 177776
4794 024610 105013
4795 024612 012605
4796 024614 012603

```

```

4797 024616 012602
4798 024620 012601
4799 024622 012600
4800 024624 005672 176662
4801 024630 024654
4802 024632 016666 000002 000004
4803 024640 012616
4804 024642 000002
4805 024644 023420
4806 024646 001750
4807 024650 000144
4808 024652 000012
4809 024654 000004
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835 024664 017646 000000
4836 024670 116667 000001 000213
4837 024676 112667 000211
4838 024702 002716 000002
4839 024706 000406
4840 024710 112707 000001 000173
4841 024716 112767 000006 000167
4842 024724 112767 000005 000156
4843 024732 010346
4844 024734 010446
4845 024736 010546
4846 024740 116704 000147
4847 024744 005404
4848 024746 002704 000006
4849 024752 110467 000134
4850 024756 116704 000127
4851 024762 016605 000012
4852 024766 005003

```

```

4853 024770 006105 1S: ROL R5 ;;ROTATE MSB INTO "C"
4854 024772 004044 BR R3 ;;GO DO MSB
4855 024774 006105 2S: ROL R5 ;;FORM THIS DIGIT
4856 024776 006105 ROL R5
4857 025000 006105 ROL R5
4858 025002 010503 MOV R5,R3
4859 025004 006103 3S: ROL R3 ;;GET LSB OF THIS DIGIT
4860 025006 105367 000100 DECB $OMODE ;;TYPE THIS DIGIT?
4861 025012 100017 BPL R5 ;;BR IF NO
4862 025014 042703 177770 BIC #17770,R3 ;;GET RID OF JUNK
4863 025020 001002 BNE 4S ;;TEST FOR C
4864 025022 005704 TST R4 ;;SUPPRESS THIS 0?
4865 025024 001403 BEQ 5S ;;BR IF YES
4866 025026 005204 4S: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
4867 025030 052703 000060 BIS #'0,R3 ;;MAKE THIS DIGIT ASCII
4868 025034 052703 000040 5S: BIS #'1,R3 ;;MAKE ASCII IF NOT ALREADY
4869 025040 110367 000042 MOV R3,R5 ;;SAVE FOR TYPING
4870 025044 004567 176442 JET R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4871 025050 025106 .WORD 8S ;;ADDRESS OF MESSAGE TO BE TYPED
4872 025052 105367 000032 7S: DECB $OCNT ;;COUNT BY 1
4873 025056 003346 BGT 2S ;;BR IF MORE TO DO
4874 025060 002402 BLT 6S ;;BR IF DONE
4875 025062 005204 INC R4 ;;INSURE LAST DIGIT ISN'T A BLANK
4876 025064 0007-3 BR 2S ;;GO DO THE LAST DIGIT
4877 025036 012605 6S: MOV (SP)+,R5 ;;RESTORE R5
4878 025070 012604 MOV (SP)+,R4 ;;RESTORE R4
4879 025072 012603 MOV (SP)+,R3 ;;RESTORE R3
4880 025074 016666 000002 000004 MCV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING
4881 025102 012616 MOV (SP)+,(SP)
4882 025104 000002 RTI ;;RETURN
4883 025106 000000 .BYTE 0 ;;STORAGE FOR ASCII DIGIT
4884 025107 000000 .BYTE 0 ;;TERMINATOR FOR TYPE ROUTINE
4885 025110 000000 $OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
4886 025111 000000 $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
4887 025112 000000 $OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
4888 .ERROR TRAP SERVICE ROUTINE
4889 025114 005727 ERRTRP: TST (PC)+ ;;CHECK IF PREV TRAP TO 4 REPORTED
4890 025116 000000 1S: .WORD 0 ;;CONTAINS ERROR REPORTED FLAG
4891 025120 001010 BNE 2S ;;BRANCH IF NOT REPORTED
4892 025122 005267 INC 1S ;;SET DOUBLE TRAP FLAG.
4893 025126 011667 154034 MOV (SP), STMP3 ;;SAVE THE BAD PC FOR TYPOUT.
4894 025132 004767 174302 JSR PC, ERROR ;;**ERROR** (GO TYPE A MESSAGE)
4895 025136 000031 .WORD 31 ;;ERROR TYPE CODE.
4896 025140 000041 BR 3S ;;SKIP HALT
4897 025142 000000 2S: HALT ;;ERROR! SECOND TRAP TO 4 OCCURRED
4898 BEFORE FIRST WAS PRINTED
4899 025144 005067 177746 3S: CLR 1S
4900 025150 000002 RTI ;;RETURN TO PROGRAM AND TRY TO RECOVER

.SBTTL PHYSICAL ADDRESS TYPE ROUTINE
;* ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
STYPAD:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK

```

```

4909 025162 016602 000012 MOV 12(SP), R2 ;;GET BASE ADDRESS
4910 025166 005003 CLR R3 ;;WORKING & INDEX REGISTER
4911 025170 005767 153412 TST MMAVA ;;CHECK FOR MEM MGMT AVAILABLE
4912 025174 0014-0 BEQ 1S ;;BRANCH IF NO MEM MGMT
4913 025176 002737 000001 177572 BIT #1, @#SR0 ;;CHECK IF MEM MGMT ENABLED
4914 025184 001404 BEQ 1S ;;BRANCH IF MEM MGMT NOT ENABLED
4915 025206 010201 MOV R2, R1 ;;COPY VIRTUAL ADR
4916 025210 006101 RCL R1 ;;SHUFFLE BITS 13,14,15 INTO 1,2,3
4917 025212 006101 ROL R1
4918 025214 006101 ROL R1
4919 025216 006101 ROL R1
4920 025220 006101 ROL R1
4921 025222 042701 177761 BIC #177761, R1 ;;CLR ALL EXCEPT BITS 1,2,3
4922 025226 052701 172340 ADD #KIPAR0, R1 ;;SET TO APPROPRIATE PAR
4923 025232 011101 MOV (R1), R1 ;;GET CONTENTS OF PAR
4924 025234 012700 000006 MOV #6, R0 ;;SET UP COUNTER
4925 025240 006301 4S: ASL R1 ;;SHIFT PAR
4926 025242 006103 ROL R3 ;;SAVE OVERFLOW BITS
4927 025244 077003 SOB R0, 4S ;;COUNT SIX SHIFTS
4928 025246 042702 160000 BIC #160000, R2 ;;SAVE BANK BITS
4929 025252 000102 ADD R1, R2 ;;COMPUTE PHYSICAL ADDRESS
4930 025254 005503 ADC R3 ;;MAKE SURE CARRY ISN'T LOST!
4931 025256 006302 1S: ASL R2 ;;FIRST DIGIT TO R3
4932 025260 006103 ROL R3
4933 025262 012700 000006 MOV #6,R0 ;;DIGIT COUNT
4934 025268 004004 BR 2S ;;PRINT FIRST DIGIT
4935 025270 006302 2S: ASL R2
4936 025272 006103 ROL R3
4937 025274 005301 DEC R1
4938 025276 001374 BNE 2S
4939 025300 012701 000003 3S: MOV #3,R1 ;;DIGIT SHIFT COUNT
4940 025304 002703 000060 ADD #60, R3 ;;MAKE IT AN ASCII DIGIT
4941 025310 110367 000036 MOV R3, 8S ;;LOAD DIGIT INTO MESSAGE
4942 025314 004567 1761-3 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4943 025320 025352 .WORD 8S ;;ADDRESS OF MESSAGE TO BE TYPED
4944 025322 005003 CLR R3 ;;CLEAR INDEX
4945 025324 005300 DEC R0 ;;DEC DIGIT COUNT
4946 025326 001360 BNE 2S
4947 025330 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
4948 025332 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
4949 025334 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
4950 025336 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
4951 025340 012616 MOV (SP)+,(SP) ;;ADJUST THE STACK TO CLEAR DATA
4952 025342 004567 176144 JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4953 025346 027005 .WORD FILL2 ;;ADDRESS OF MESSAGE TO BE TYPED
4954 025350 000207 RTS PC ;;RETURN
4955 025352 000000 .BYTE 0 ;;ONE DIGIT MESSAGE BUFFER
4956 025353 000000 .BYTE 0 ;;MESSAGE TERMINATOR
4957
4958 .SBTTL STANDARD PROGRAM MESSAGES
4959 *****
4960 ;VARIOUS MESSAGE PRINTOUTS USED THROUGH
4961 ;THE PROGRAM
4962 *****
4963 025354 005015 052113 030461 MMAMES: .ASCII <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
4964 025362 024040 042515 047515

```

4965 025370 054522 046440 047101
4966 025376 043501 046705 047105
4967 025404 024524 040440 040526
4968 025412 046111 041101 042574
4969 025420 000
4970 025421 015 046412 046305 MEMMES: .ASCIZ <15><12>'MEMORY MAP:'
4971 025426 051117 020131 040515
4972 025434 025120 000
4973 025437 015 041012 052131 BYTIMES: .ASCIZ <15><12>'BYTE MEMORY MAP:'
4974 025444 020105 042515 047515
4975 025452 054522 046440 050101
4976 025460 000072
4977 025462 005015 040520 044522 MTMAP: .ASCIZ <15><12>'PARITY MEMORY MAP:'
4978 025470 054524 046440 046505
4979 025476 051117 020131 040515
4980 025504 025120 000
4981 025507 015 043012 047522 FROM: .ASCIZ <15><12>'FROM '
4982 025514 020115 000
4983 025517 040 047524 00040 TO: .ASCIZ ' TO
4984 025524 050515 047111 052523 INSUFF: .ASCIZ <15><12>'INSUFFICIENT MEMORY,..FIRST 16K NOT ALL THERE!'
4985 025532 043106 041511 042511
4986 025540 052116 046440 046505
4987 025546 051117 027131 027556
4988 025554 044506 051522 020124
4989 025562 053061 020113 047516
4990 025570 020124 046101 020114
4991 025576 044124 051105 020305
4992 025604 000
4993 025605 015 047012 020117 MTR: .ASCIZ <15><12>'NO PARITY REGISTERS FOUND'
4994 025612 040520 044522 054524
4995 025620 051040 043505 051511
4996 025626 042524 051522 043040
4997 025634 052517 042116 000
4998 025641 015 051012 051505 PWRMSG: .ASCIZ <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
4999 025646 040524 052122 047111
5000 025656 020107 043101 042524
5001 025663 020122 020101 047520
5002 025670 022527 020122 040506
5003 025676 046111 051125 006505
5004 025704 030012
5005 025706 005015 047516 050740 NOPER: .ASCIZ <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
5006 025714 051101 052111 020131
5007 025722 051105 047522 051522
5008 025730 043000 052517 042116
5009 025736 047440 020116 042515
5010 025744 047515 054522 051440
5011 025752 040503 006516 000012
5012 025760 005015 051120 043517 PROREL: .ASCIZ <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO BK'
5013 025766 040522 020115 047516
5014 025774 020127 042522 044523
5015 026002 02504 020123 040502
5016 026010 035503 040440 020124
5017 026016 020060 047524 034040
5018 026024 113
5019 026025 015 044012 052111 .ASCIZ <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
5020 026032 041440 047117 044524

5021 026040 052516 020105 047509
5022 026046 020122 047516 046522
5023 026054 046101 051040 047125
5024 026062 044516 043516 005015
5025 026070 000
5026 026071 015 051012 043705 MX1: .ASCIZ <15><12>'REGISTER AT '
5027 026076 051511 042524 020122
5028 026104 052101 000040
5029 026110 041440 047117 051124 MX2: .ASCIZ ' CONTROLS '
5030 026115 046117 020123 000
5031 026123 015 041412 051177 MX3: .ASCIZ <15><12>'CORE PARITY '
5032 026130 020115 040520 044522
5033 026136 045100 000040
5034 026142 0000 047515 020123 MX4: .ASCIZ <15><12>'MOS PARITY '
5035 026150 040503 044522 054524
5036 026156 000040
5037 026160 000015 051515 030461 MX5: .ASCIZ <15><12>'MS11-K CSR '
5038 026166 045456 041440 051123
5039 026174 000040
5040 026176 051515 0461 045455 MX6: .ASCIZ 'MS11-K MEMORY PRESENT!! TO COMPLETELY TEST RUN DZMML...'
5041 026204 046440 04505 051117
5042 026212 020131 051122 051505
5043 026220 047105 020524 020741
5044 026226 047524 041440 046517
5045 026234 046120 052105 04675
5046 026242 020131 042524 052743
5047 026250 051040 047125 042040
5048 026256 046502 046115 027056
5049 026264 000056
5050 026266 005015 047516 046440 NOMEM: .ASCIZ <15><12>'NO MEMORY FOUND.'
5051 026274 046505 051117 020131
5052 026302 047505 047125 027104
5053 026310 000
5054 026311 015 005012 044412 FADMES: .ASCIZ <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
5055 026316 050116 052125 040440
5056 026324 046114 050040 051101
5057 026332 046501 052105 051105
5058 026340 020123 047111 047440
5059 026346 052103 046101 000
5060 026353 015 040012 051111 .ASCIZ <15><12>'FIRST ADDRESS: '
5061 026360 052123 050440 042104
5062 026366 042522 051323 020772
5063 026374 000040
5064 026376 005015 040514 03113 LADMES: .ASCIZ <15><12>'LAST ADDRESS: '
5065 026404 040440 042104 042522
5066 026412 051523 020072 0200
5067 026420 000
5068 026421 015 037412 042101 BADADR: .ASCIZ <15><12>'ADDRESS IN UNMAPPED BANK?'
5069 026426 041104 041505 020123
5070 026434 047111 052440 046516
5071 026442 050101 042520 020104
5072 026450 040502 046516 000077
5073 026456 005015 042524 042514 CONST: .ASCIZ <15><12>'SELECT CONSTANT.'
5074 026464 052103 041440 047117
5075 026472 052123 047101 035124
5076 026500 000

5077 026501 015 052412 042516 UNEXPT: .ASCIZ <15><12>'UNEXPECTED MEMORY PARITY ERROR'
5078 026506 050130 041505 042524
5079 026514 020104' 042515 047515
5080 026522 054523 050040 051101
5081 026530 022111 020131 051105
5082 026536 047522 000122
5083 026542 005015 051120 043517 PKELOC: .ASCIZ <15><12>'PROGRAM RELOCATED TO '
5084 026550 040522 020115 042522
5085 026556 047514 040503 042524
5086 026564 020104 047524 000140
5087 026572 005015 047515 042522 MTOE' .ASCIZ <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
5088 026600 052040 040510 020116
5089 026606 047117 020105 040520
5090 026614 044522 054524 042440
5091 026622 051122 051117 043740
5092 026630 052517 042116 000756
5093 026636 005015 041523 047101 SCANM: .ASCIZ <15><12>'SCANNING MEMORY FOR BAD PARITY.'
5094 026644 044516 043516 046440
5095 026652 046505 051117 020131
5096 026660 047506 020122 040502
5097 026666 020104 040520 044522
5098 026674 054524 000756
5099 026700 005015 040520 044522 PEWNC: .ASCIZ <15><12>'PARITY ERROR WILL NOT CLEAR.'
5100 026706 054524 042440 051172
5101 026714 051117 053440 046111
5102 026722 020114 047516 020124
5103 026730 046103 040505 027122
5104 026736 000
5105 026737 015 047012 020117 NDMTST: .ASCIZ <15><12>'NO MEMORY TESTED.'
5106 026744 042515 047515 054522
5107 026752 052040 051505 042524
5108 026760 027104 000
5109 026763 015 051412 044513 SKPMES: .ASCIZ <15><12>'SKIPPING TEST #'
5110 026770 050120 047117 020107
5111 026776 042524 052123 021440
5112 027004 000
5113 027005 377 000377 FILL2: .ASCIZ <377><377>
5114
5115
5116
5117
5118
5119 027010 040520 044522 054524
5120 027016 051040 043505 051511
5121 027024 042524 020122 040504
5122 027032 040524 042440 051122
5123 027040 051117 000056
5124 027044 042101 051104 051505 DM2: .ASCIZ 'ADDRESS TEST ERROR(TST1-5).'5125 027052 020123 042524 052123
5126 027060 042440 051122 051117
5127 027066 052050 052123 026461
5128 027074 024465 000056
5129 027100 047503 051516 040524 DM4: .ASCIZ 'CONSTANT DATA ERROR(TST6-10).'5130 027106 052116 042040 052101
5131 027114 020101 051105 047522
5132 027122 024122 051524 033124

5133 027130 030455 024460 000056
5134 027136 047522 040524 044524 DM5: .ASCIZ 'ROTATING BIT ERROR(TST11-12).'5135 027144 043515 041000 052111
5136 027152 042440 051122 051117
5137 027160 052050 052123 030461
5138 027166 030455 024462 000756
5139 027174 047515 020123 042522 DM6: .ASCIZ 'MDS REFRESH TEST ERROR (TST 30-31).'5140 027202 051106 051505 020110
5141 027210 042524 052123 042440
5142 027216 051122 051117 024740
5143 027224 051524 020124 030763
5144 027232 031455 024461 000756
5145 027240 020063 047530 020122 DM7: .ASCIZ '3 XOR 9 PATTERN ERROR(TST13-16).'5146 027246 020071 040507 052124
5147 027254 051105 020116 051105
5148 027262 047522 024122 051524
5149 027270 030524 026433 033061
5150 027276 027051 000
5151 027301 115 051101 044103 DM10: .ASCIZ 'MARCHING 1'S AND 0'S ERROR(TST 27).'5152 027306 047111 020107 023461
5153 027314 020123 047101 020104
5154 027322 023460 020123 051105
5155 027330 047522 024122 051524
5156 027336 020124 033462 027051
5157 027344 000
5158 027345 120 051101 052111 DM11: .ASCIZ 'PARITY MEMORY ADDRESS ERROR(TST17).'5159 027352 020131 042515 042115
5160 027360 054522 040440 042104
5161 027366 042522 051523 042440
5162 027374 051122 051117 052050
5163 027402 052123 033461 027951
5164 027410 000
5165 027411 104 052101 050111 DM12: .ASCIZ 'DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).'5166 027416 053440 052111 020110
5167 027424 051127 047117 020107
5168 027432 040520 044522 054524
5169 027440 042040 042111 023516
5170 027446 020124 051124 050101
5171 027454 052050 052123 033161
5172 027462 027051 000
5173 027465 127 047522 043516 DM13: .ASCIZ 'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
5174 027472 050040 051101 052111
5175 027500 020131 051124 050101
5176 027506 042507 026104 041240
5177 027514 052125 047040 020117
5178 027522 042522 044507 052123
5179 027530 051105 051440 047510
5180 027536 051527 042440 051122
5181 027544 051117 043040 040514
5182 027552 027107 000
5183 027555 120 051101 052111 DM14: .ASCIZ 'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).'5184 027562 020131 042522 044507
5185 027570 052123 051105 047040
5186 027576 052117 046440 050101
5187 027604 042520 020104 051501
5188 027612 041440 047117 051124

5189	027620	046117	044514	043516						
5190	027626	052040	044510	020123						
5191	027634	042101	051104	051505						
5192	027642	024123	351524	030524						
5193	027650	024467	000056							
5194	027654	047515	042522	052140	DM16:	.ASCIZ	'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'			
5195	027662	040510	020116	047117						
5196	027670	020105	042522	044507						
5197	027676	052123	051105	044440						
5198	027704	042116	041511	052101						
5199	027712	042105	050040	051101						
5200	027720	052111	020131	151105						
5201	027725	047522	027122	100						
5202	027733	104	052101	020101	DM17:	.ASCIZ	'DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST17).'			
5203	027740	044123	052517	042114						
5204	027746	023516	020124	040510						
5205	027754	042526	041440	040510						
5206	027762	043516	042105	053140						
5207	027770	042510	020116	040520						
5208	027776	044522	054524	042440						
5209	030004	051122	051117	052140						
5210	030012	040522	050120	042105						
5211	030020	052050	052123	033461						
5212	030026	027011	000							
5213	030031	122	047101	047504	DM20:	.ASCIZ	'RANDOM DATA ERROR(TST20).'			
5214	030036	020115	040504	040524						
5215	030044	042440	051122	051117						
5216	030052	052050	052123	030062						
5217	030060	027051	000							
5218	030063	111	051516	051124	DM21:	.ASCIZ	'INSTRUCTION EXECUTION ERROR(TST21-26).'			
5219	030070	01525	044524	047117						
5220	030075	042440	042530	052503						
5221	030104	044524	047117	042140						
5222	030112	051122	051117	052150						
5223	030120	052123	030462	031055						
5224	030126	044466	000056							
5225	030132	051120	043517	040522	DM23:	.ASCIZ	'PROGRAM CODE CHANGED WHEN RELOCATED.'			
5226	030140	020115	047503	042504						
5227	030146	041440	040510	043516						
5228	030154	042105	053440	042510						
5229	030162	020116	042522	047514						
5230	030170	040503	042424	027104						
5231	030176	000								
5232	030177	124	040522	050110	DM24:	.ASCIZ	'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'			
5233	030204	042105	020054	052502						
5234	030212	020124	047516	051040						
5235	030220	043505	051511	042524						
5236	030226	020122	040510	020104						
5237	030234	051105	047522	020122						
5238	030242	044502	020124	042523						
5239	030250	027124	000							
5240	030253	124	040522	050120	DM25:	.ASCIZ	'TRAPPED TO 114.'			
5241	030260	042105	052040	020117						
5242	030268	030461	027004	100						
5243	030273	106	044501	042514	DM26:	.ASCIZ	'FAILED TO TRAP.'			
5244	030300	020104	047524	052040						

5245	030306	040522	027120	000						
5246	030313	050	041501	044524	DM27:	.ASCIZ	'(ACTION ENABLE WASN'T SET).'			
5247	030320	047117	042440	040516						
5248	030326	046102	020105	040527						
5249	030334	047123	052047	051440						
5250	030342	052105	027051	000						
5251	030347	051123	052012	040522	DM31:	.ASCIZ	<15><12>'TRAPPED TO 4'			
5252	030354	050120	042105	052040						
5253	030362	020117	020064	000						
5254										
5255										
5256										
5257										
5258										
5259	030367	120	044503	042522	DH1:	.ASCIZ	'PC REG S/B WAS'			
5260	030374	044507	027523	044502						
5261	030402	040527	000123							
5262	030406	027526	041520	050011	DH2:	.ASCIZ	'V/PC P/PC MA S/B WAS'			
5263	030414	050057	044503	040515						
5264	030422	051411	041057	053411						
5265	030430	051501	000							
5266	030433	126	050057	044503	DH12:	.ASCIZ	'V/PC P/PC MA S/B'			
5267	030440	027520	041520	046411						
5268	030446	044501	027523	000102						
5269	030454	027526	041520	050011	DH14:	.ASCIZ	'V/PC P/PC REG MA'			
5270	030462	050057	044503	042522						
5271	030470	044507	040515	000						
5272	030475	126	050057	044503	DH15:	.ASCIZ	'V/PC P/PC MAUT REG S/B WAS'			
5273	030502	027520	041520	046411						
5274	030510	042501	044524	042522						
5275	030516	044507	027523	040502						
5276	030524	040527	000123							
5277	030530	027526	041520	050011	DH21:	.ASCIZ	'V/PC P/PC IUT MA S/B WAS'			
5278	030536	050057	044503	052511						
5279	030544	044524	040515	051411						
5280	030552	041057	053411	051501						
5281	030560	000								
5282	030561	126	050057	044503	DH23:	.ASCIZ	'V/PC P/PC SRC MA DST MA S/B WAS'			
5283	030566	027520	041520	051411						
5284	030574	041522	046440	040501						
5285	030602	051504	020124	040515						
5286	030610	051411	041057	053411						
5287	030616	051501	000							
5288	030621	126	050057	044503	DH24:	.ASCIZ	'V/PC P/PC TRP/PC'			
5289	030626	027520	041520	052011						
5290	030634	050122	050057	000103						
5291	030642	027526	041520	051111	DH25:	.ASCIZ	'V/PC P/PC TRP/PC REG WAS'			
5292	030650	050057	044503	051124						
5293	030656	027520	041520	051011						
5294	030664	043505	053411	051501						
5295	030672	000								
5296	030673	126	050057	044503	DH26:	.ASCIZ	'V/PC P/PC REG WAS'			
5297	030700	027520	041520	051011						
5298	030706	043505	053411	051501						
5299	030714	000								
5300	030715	122	043505	053411	DH30:	.ASCIZ	'REG WAS MA WAS'			

```

5301 030722 051501 046411 004501
5302 030730 040527 000123
5303
5304 ;*****
5305 ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
5306 ;*****
5307 030734 000 377 000 DF1: .BYTE 0,-1,0,0
5308 030737 000
5309 030740 000 377 377 DF2: .BYTE 0,-1,-1,0,0
5310 030743 000 000
5311 030745 000 377 DF3: .BYTE 0,-1,-1,-2,-2
5312 030750 376 376
5313 030752 000 377 377 DF14: .BYTE 0,-1,-1,-1,0,0
5314 030755 377 000
5315 030760 000 377 000 DF21: .BYTE 0,-1,0,-1,0,0
5316 030763 377 000 000
5317 030766 377 C30 377 DF30: .BYTE -1,0,-1,-2
5318 030771 376
5319 .EVEN
5320
5321 032110 . = 32110 ;THE LOADERS ARE SAVE HERE TO END OF BK
5322
5323 000001 .END
    
```

```

ABASE = 000000 384 425
ACDW1 = 000000 384 427
ACDW2 = 000000 384 428
ACPUOP = 000000 384 399
ACDW0 = 000000 384 429
ADW1 = 000000 384 430
ADW10 = 000000 384 439
ADW11 = 000000 384 440
ADW12 = 000000 384 441
ADW13 = 000000 384 442
ADW14 = 000000 384 443
ADW15 = 000000 384 444
ADDW2 = 000000 384 431
ADDW3 = 000000 384 432
ADDW4 = 000000 384 433
ADDW5 = 000000 384 434
ADDW6 = 000000 384 435
ADDW7 = 000000 384 436
ADDW8 = 000000 384 437
ADDW9 = 000000 384 438
ADEVCT = 000000 384 390
ADEVM = 000000 384 426
AE = 000001 179# 2433 2519 3750
AENV = 000000 384 395
AENVM = 000000 384 396
AFATAL = 000000 384 387
AMADR1 = 000000 384 412
AMADR2 = 000000 384 416
AMADR3 = 000000 384 419
AMADR4 = 000000 384 422
AMAMS1 = 000000 384 406
AMAMS2 = 000000 384 414
AMAMS3 = 000000 384 417
AMAMS4 = 000000 384 420
AMSGAD = 000000 384 392
AMSGLG = 000000 384 393
AMSGTY = 000000 384 386
AMTYP1 = 000000 384 407
AMTYP2 = 000000 384 415
AMTYP3 = 000000 384 418
AMTYP4 = 000000 384 421
APASS = 000000 304 389
APRIOR = 000000 384
APICSU = 000040 4598 473 #
APIENV = 000001 4187 4531 4659 4737#
APTSIZ = 000000 676 4736#
APTSPO = 000100 4593 4651 4738#
AS&REG = 000000 384 397
ATESTN = 000000 384 398
AUNIT = 000000 384 391
AUS&R = 000000 384 393
AVECT1 = 000000 384 423
AVECT2 = 000000 384 424
BADADR 025421 1446 5063#
BANKNO 016130 1632 1641
BITPT 001544 522# 1134* 1135* 1171 1172 1173 1174 1182* 1183* 1192 1194 1197 1200*
    
```


FADMSK	001566	538#	3368	4134*	4136*															
FILL2	027005	4953	5113#																	
FLAGBK	001557	532#	3207*	3297*	3300*	3318*														
FROM	025507	1048	3951	4881#																
FSTADR	001562	535#	1357	1455*	3744	4065														
GMPR	004072	1013	1033#																	
GMPRA	004120	1098#	1110																	
GMPRB	004134	1097	1104#																	
GMPRC	004142	1102	1106#																	
GMPRD	004202	1095	1118#																	
GNS	= ***** U	194	905																	
HT	= 000011	36#	4606	4647																
IMPCH	= 177746	186#	939*																	
IMPCK	003272	929	927	936#																
INITDN	015064	1528	1564	1591	1666	1676	3:22#													
INITEX	015176	3218	3220	3239	3241#															
INITMM	014426	1511	1554	1602	1631	1640	1708	1722	1758	1779	1802	1827	1837	1880						
		1929	1940	1983	2032	2043	2091	2217	2228	2276	2408	2555	2616	2665						
		2714	2764	2813	2862	2906	2958	2977	2990	3010	3023	3148#	3417							
INSERT	021124	4056#																		
INSUFF	025524	1015	4984#																	
IOIVEC	= 000020	11#																		
KIPARO	= 172340	166#	242	261*	3133*	3554*	3620*	4922												
KIPARI	= 172342	167#	26*	3134*	3555*	3621*														
KIPAR2	= 172344	168#	975	995*	1072*	1181*	1185	2484	3135*	3154*	3160*	3171	3228*	3258*						
		3276	3349*	3381	3536*	3539*	3554	3615*	3809	3810*	3849*	3854*	3528*	3536	3555					
KIPAR3	= 172346	169#	1069	1072	1073*	3136*	3171*	3176*												
		3616*																		
KIPAR4	= 172350	170#	3137*																	
KIPAR5	= 172352	171#	3138*																	
KIPAR6	= 172354	172#	3139*																	
KIPAR7	= 172355	173#	3140*																	
KIPDR0	= 172300	155#	237	3125*																
KIPDR1	= 172302	156#	3126*																	
KIPDR2	= 172304	157#	3127*																	
KIPDR3	= 172306	158#	3128*																	
KIPDR4	= 172310	159#	3129*																	
KIPDR5	= 172312	160#	3130*																	
KIPDR6	= 172314	161#	3131*																	
KIPDR7	= 172316	162#	3132*																	
LADMAP	001602	546#	1443*	1450*	1473*	1474*	3165	3167	3184	3186	3199	3208	3236	3238						
		3250	3252	3266	3268	3285	3287	3307	3317	3319	4107	4108	4109	4111						
		1401	5064#																	
LADMSK	001600	544#	3169	3188	3210	3270	3269	3309	3321	4139*	4141*									
LDDISP	001522	508#	830*																	
LF	= 000012	37#	4641	4647																
LMAD	001520	507#	3639	3670*	3677	3691*														
LSTADR	001574	541#	1353*	1412*	1417	1428	1467*	1468*	1469*	1470*	1471	3240	4074	4081						
MAMF	017536	1488	2546	3736#																
MAMF1	017622	3750#	3752																	
MAMF2	017634	3737	3739	3745	3754#	1448														
MANUAL	005366	1119	1274	1343	1353#															
MANUL1	005424	1362	1369#																	
MANUL2	006026	1364	1465#																	
MAPMEM	003324	938	952#																	
MAPRB	004250	1136	1143#																	

MASK4K	= 017777	183#	966	991	1065	1419	1471	3212	3323	3842	4098	4105	4134	4139						
MEMMAP	001524	200	509#	953	954	1192	1194	1197	1394	1396	1435	1437	1440	1442						
		3532	3534	3543	3545	3565	3570	3812	3814											
MEMMES	025421	962	497*																	
MMAMES	025354	932	4963*																	
MMAVA	000606	233	230#	924*	930*	972	993	1024*	1030*	1067	1137	1179	1451	1465						
		1489	2481	3152	3224	3255	3347	3378	3523	3613	3662	3671	3807	3847						
		4911																		
MIDDOWN	015674	1540	1576	1598	1672	1688	3234	3340#												
MILNIT	014300	929	1029	1140	1493	3124#														
MMDRE	001554	528#	3232*	3233*	3241*	3326	3369													
MWUP	015204	1524	1560	1616	1636	1651	1712	1733	1769	1792	1815	1831	1874	1917						
		1933	1977	2020	2036	2084	2204	2221	2269	2389	2545	2580	2630	2679						
		2725	2778	2827	2877	2959	2983	3003	3016	3036	3250#	3421								
MMVEC	= 000250	144#																		
MMMEM	004210	1114	1133#																	
MPEND	002074	617#																		
MPPATS	002050	585	636#																	
MFRX	002276	563	637*	1093	1109	1113	1176	1248	1261	1272	1288	1334	1342	2401						
		3329	3736	3759																
MPR0	002076	562	632*	1096	1150	1212	1283	1328	1337											
MPR1	002106	636#																		
MPR10	002216	672#																		
MPR11	002226	676#																		
MPR12	002236	680#																		
MPR13	002246	684#																		
MPR14	002256	688#	1187																	

CZQMCF0 0-124K MEMORY EXERCISER, 16K VER MACY*1 30A(1052) 20-FEB-78 07:56 PAGE 124
CZQMCF.P11 14-FEB-78 08:19 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0204

ERRORS DETECTED: 0

CZQMCF.SIN,CZQMCF.LST/CRF/SOL/NL:TOC=CZQMCF.SML,CZQMCF.P11
RUN-TIME: 21 28 1 SECONDS
RUN-TIME RATIO: 130/51=2.5
CORE USED: 39K (77 PAGES)