

The main body of the document is a large grid of 15 columns and 20 rows of small, illegible text blocks. Each block appears to be a small exercise or data set related to the CPU instruction exerciser. The text is too small to read, but the layout is consistent across the entire page.

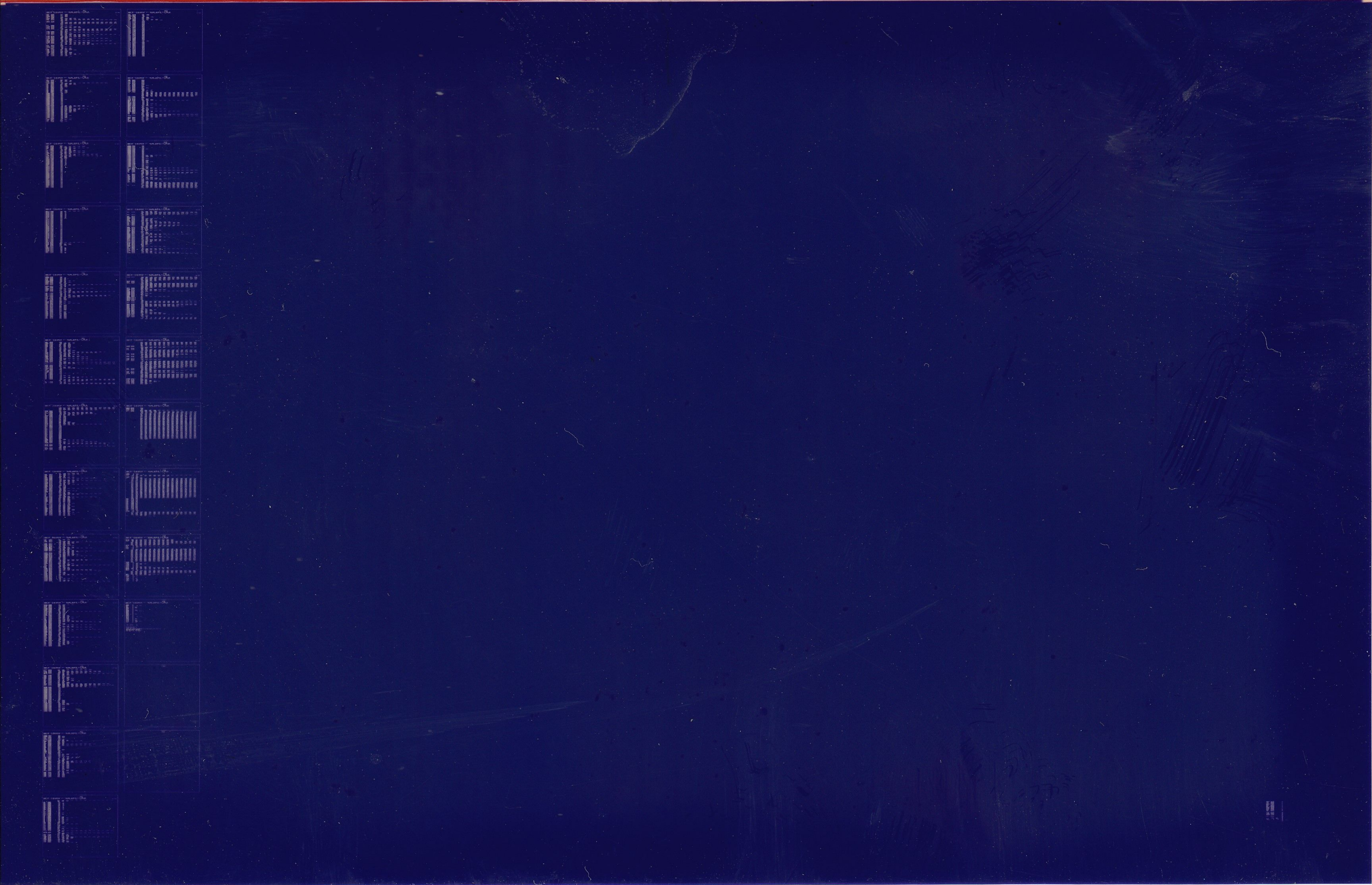
# PDP11/70

CPU INSTRUCTION EXERCISER  
CEQKCC0

AH-7996C-MC  
COPYRIGHT © 75-78  
FICHE 2 OF 2

JUN 1978  
**digital**  
MADE IN USA

Address	Instruction	Op-Code	Address	Instruction	Op-Code
000000	...	...	000000	...	...
000001	...	...	000001	...	...
000002	...	...	000002	...	...
000003	...	...	000003	...	...
000004	...	...	000004	...	...
000005	...	...	000005	...	...
000006	...	...	000006	...	...
000007	...	...	000007	...	...
000008	...	...	000008	...	...
000009	...	...	000009	...	...
000010	...	...	000010	...	...
000011	...	...	000011	...	...
000012	...	...	000012	...	...
000013	...	...	000013	...	...
000014	...	...	000014	...	...
000015	...	...	000015	...	...
000016	...	...	000016	...	...
000017	...	...	000017	...	...
000018	...	...	000018	...	...
000019	...	...	000019	...	...
000020	...	...	000020	...	...
000021	...	...	000021	...	...
000022	...	...	000022	...	...
000023	...	...	000023	...	...
000024	...	...	000024	...	...
000025	...	...	000025	...	...
000026	...	...	000026	...	...
000027	...	...	000027	...	...
000028	...	...	000028	...	...
000029	...	...	000029	...	...
000030	...	...	000030	...	...
000031	...	...	000031	...	...
000032	...	...	000032	...	...
000033	...	...	000033	...	...
000034	...	...	000034	...	...
000035	...	...	000035	...	...
000036	...	...	000036	...	...
000037	...	...	000037	...	...
000038	...	...	000038	...	...
000039	...	...	000039	...	...
000040	...	...	000040	...	...
000041	...	...	000041	...	...
000042	...	...	000042	...	...
000043	...	...	000043	...	...
000044	...	...	000044	...	...
000045	...	...	000045	...	...
000046	...	...	000046	...	...
000047	...	...	000047	...	...
000048	...	...	000048	...	...
000049	...	...	000049	...	...
000050	...	...	000050	...	...
000051	...	...	000051	...	...
000052	...	...	000052	...	...
000053	...	...	000053	...	...
000054	...	...	000054	...	...
000055	...	...	000055	...	...
000056	...	...	000056	...	...
000057	...	...	000057	...	...
000058	...	...	000058	...	...
000059	...	...	000059	...	...
000060	...	...	000060	...	...
000061	...	...	000061	...	...
000062	...	...	000062	...	...
000063	...	...	000063	...	...
000064	...	...	000064	...	...
000065	...	...	000065	...	...
000066	...	...	000066	...	...
000067	...	...	000067	...	...
000068	...	...	000068	...	...
000069	...	...	000069	...	...
000070	...	...	000070	...	...
000071	...	...	000071	...	...
000072	...	...	000072	...	...
000073	...	...	000073	...	...
000074	...	...	000074	...	...
000075	...	...	000075	...	...
000076	...	...	000076	...	...
000077	...	...	000077	...	...
000078	...	...	000078	...	...
000079	...	...	000079	...	...
000080	...	...	000080	...	...
000081	...	...	000081	...	...
000082	...	...	000082	...	...
000083	...	...	000083	...	...
000084	...	...	000084	...	...
000085	...	...	000085	...	...
000086	...	...	000086	...	...
000087	...	...	000087	...	...
000088	...	...	000088	...	...
000089	...	...	000089	...	...
000090	...	...	000090	...	...
000091	...	...	000091	...	...
000092	...	...	000092	...	...
000093	...	...	000093	...	...
000094	...	...	000094	...	...
000095	...	...	000095	...	...
000096	...	...	000096	...	...
000097	...	...	000097	...	...
000098	...	...	000098	...	...
000099	...	...	000099	...	...



B01

EOF1CZDE00580411

00010000

780519

IDENTIFICATION

0#HDR1CEQKCCSEQ

00010000

780519  
SEQ 0001

PRODUCT CODE: AC-7994C-MC  
PRODUCT NAME: CEQKCCD PDP11/70 CPU INSTRUCTION EXERCISER  
DATE CREATED: 15-MAR-78  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR(S): DONALD W. MONROE-REV B  
           JOHN ADAMS-REV A  
MODIFIED BY: BILL SCHLITZKUS

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

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright (C) 1975, 1978 by Digital Equipment Corporation

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION

DIGITAL           PDP           UNIBUS           MASSBUS

CONTENTS  
-----

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	Equipment
2.2	Storage
2.3	Preliminary Programs
3.0	LOADING PROCEDURE
3.1	Method
4.0	STARTING PROCEDURE
4.1	Control Switch Settings
4.2	Starting Addresses
4.3	Program and Operator Action
5.0	OPERATING PROCEDURE
5.1	Operational Switch Settings
5.2	Display Register
5.3	Operator Action
6.0	ERRORS
6.1	Error Halts and Description
6.2	Error Recovery
7.0	WARNINGS AND EXCEPTIONS
7.1	Warnings
7.2	Exceptions
8.0	MISCELLANEOUS
8.1	Execution Time
8.2	Stack Pointer
8.3	Pass Count
8.4	Iterations
8.5	T Bit Trapping
8.6	ACT11 Competability
8.7	PSW and Margin Tables
8.8	I/O Device Address Modifications
8.9	Power Failure
9.0	PROGRAM DESCRIPTION
9.1	Micro Break Test
9.2	Unibus Exerciser Function
9.3	Mass Bus Tester Function
9.4	Line Clock Initialization

1.0 ABSTRACT

This program is designed to be a comprehensive check of the PDP-11/70 cpu cluster. The program executes each instruction in all address modes and includes tests for traps, interrupts, the mapping box, memory management, memory, the Unibus, and the Mass Bus. If not deselected, the program relocates the test code throughout memory (0-2m). Also, if not deselected, the program will relocate using available disks (RPO3, RK05, RPO4, RS03/4). See section 9.5 for a description of relocation.

The main differences between revision A and revision B are routines to use the UBE and MBT (manufacturing only), worst case testing occurs with all switches down, standard SYSMAC macros, and floating point processor tests.

Also, the disk driver was rewritten to make each device have a modular driver and to cause I/O to occur concurrently on the available disks. (see section 9.5.4 for a description of disk drivers)

Since worst case testing now occurs with all switches down, precautions must be taken to ensure the protection of user disks. Refer to section 7.0 for a description of warnings and exceptions.

2.0 REQUIREMENTS

2.1 Equipment

PDP-11/70 Central Processor with 16K of memory, a line clock, and an LA30 (or equivalent) console.

2.1.1 Optional Equipment Used

1. Unibus Exerciser
2. Mass Bus Tester
3. RP11/RPO3, RK11/RK05, RH70/RPO4, RH70/RS03/RS04
4. FP11-B, FP11-C

2.2 Storage

The program loads into the first 12K of memory and runs in all memory (exclusive of the XXDP monitor if running in chain mode).

2.3 Preliminary Programs

advisable that the CPU cluster (and floating point) diagnostics run first. These consist of:

DEKBA	DEKBF
DEKBB	DEKBG
DEKBC	DEMJA
DEKBD	DEFPA
CKBE	DEFPB

### 3.0 LOADING PROCEDURE

#### 3.1 Method

The program is supplied on the diagnostic media. Refer to the XXDP operating manual for further information.

### 4.0 STARTING PROCEDURE

#### 4.1 Console Switch Settings

See Section 5.1

#### 4.2 Starting Addresses

The starting address for the exerciser is 200.

By starting at address 210, the switch register and display lights can be checked. This routine just moves the switches to the display register allowing the operator to toggle the switches and see the corresponding lights in the display register.

By starting at address 214, the micro-break register can be checked. This test requires a maintenance card. See Section 9.0 for further details.

#### 4.3 Program and Operator Action

1. Load program into memory (See Section 3)
2. Check for any system disk packs or configuration exceptions as described in section 7.0.
3. Load address 200
4. Set switches (See Section 5.1)
5. Press Start
6. The program will loop and messages will be typed at the end of each sub-pass and each pass. (see section 8.3 for

5.0 OPERATING PROCEDURE

5.1 Operational Switch Settings

SW15	HALT ON ERROR	This switch when set will halt the processor when an error is detected. Pressing continue will cause an error message to be typed and the processor will again halt. Pressing continue again will resume testing.
SW14	LOOP ON TEST	This switch when set will cause the program to loop on the current subtest.
SW13	INHIBIT ERROR TYPEOUT	This switch when set inhibits the error typeout.
SW12	INHIBIT UBE	This switch when set inhibits the initialization of the Unibus Exerciser. See section 9.2 for a description of the UBE function.
SW11	INHIBIT SUB- TEST ITERATION	This switch when set inhibits subtest iteration after the first pass. Each subtest is executed 10 times before the next subtest is run. Setting SW11 causes each test to be executed once before starting the next subtest.
SW10	RING BELL ON ERROR	This switch when set will ring the bell when an error is detected.
SW9	LOOP ON ERROR	This switch when set will cause the program to loop on the first failure even if the failure is intermittent. See section 6.1 for a description of looping on relocation errors.
SW8	RELOCATE WITH CPU ONLY	This switch when set will cause relocation to be done by the CPU instead of a disk. See section 9.5 for a

- SW7      INHIBIT SYSTEM SIZE TYPEOUT      This switch when set will inhibit the typeout of the switch definitions and the disks that will be used for relocation. (Typeout only occurs when the program is dumped)
- SW6      INHIBIT RELOCATION      This switch when set will inhibit all relocation. Do not change this switch while the program is running.
- SW5      INHIBIT ROUND ROBIN      This switch when set will only relocate using the device selected by switches <2:0> rather than all available devices.
- SW4      INHIBIT RANDOM DISK ADDRESS      This switch when set will cause relocation to always start at address 0 on the disk(s).
- SW3      INHIBIT MBT      This switch when set inhibits the initialization of the Mass Bus Tester. See section 9.3 for a description of the MBT function.
- SW2-SW0      DEVICE CODES      These switches (along with SW5) cause the program to relocate the test code using the device specified below:

VALUE	DEVICE
0	RP11/rp03
1	RK05
2	Not used
3	Not used
4	RH70/RP04
5	RH70/RS03/RS04
6	Not used
7	Not used

NOTE

When relocating via a specific device, set in the value(SW<2:0>) to select the device then set switch 5.

Unit 0 of the load device is marked not



mode, and therefore will not be used to relocate.

## 5.2 Display Register

While the program is running, the low byte of the display register contains the subtest number and the high byte contains bits <14:7> of KERNEL PAR0. These bits, of kernel par0, correspond to bits <20:13> of the physical address of the relocated code. When an error is detected and loop on error is selected, the high byte contains the error count.

## 5.3 Operator Action

When the program is loaded\* and started with switch 7 on a zero the program will typeout the disks and unit numbers that will be used for relocation and then wait for the operator to type a character. This is to allow the operator to write protect any drive that is not to be used. If there are no devices available for relocation, operator action is not required.

If the program is loaded via ACT11 in QV or AA or with XXDP in chain mode no operator action is required and all disks not write protected (except for the XXDP media) will be used for relocation.

\*Except chain mode, QV(manufacturing only), or Auto Accept (manufacturing only)

## 6.0 ERRORS

### 6.1 Error Halts and Description

If an error is detected, the program will trap to the error handling routine (SERROR). If halt on error is enabled, the processor will halt. Pressing continue will cause an error message to be typed and the processor will halt again.

There are many different types of errors. No matter which type occurs a minimum set of information is typed as follows:

```
HHH:MM:SS  
ERRORPC PHYS PC   PSW   MAINT  TEST NO SUB-PASS CNT  
UUUUUU  VVVVVVVV  WWWWWW XXXXXX YYYYYY SSSSSS PPPPPP
```

where:

UUUUUU = Virtual PC of the error call.  
VVVVVVV = Physical PC of the error call.

XXXXXX = Contents of the maintenance register(1777750).  
 YYYYYY = Test number.  
 SSSSSS = Sub-pass count (0 thru 5)  
 PPPPPP = Pass count

HMM:MM:SS Represents the elapsed run time of the program, since the most previous start, where: HMM = hours, MM = minutes, and SS = seconds.

The Virtual PC is the 16 bit word that was pushed on the stack when the error call was made. The physical PC is calculated in one of two ways:

1. If memory management is off the contents of location "FACTOR" is subtracted from the Virtual PC. This generates the corresponding PC for the non-relocated code.
2. If memory management is on the contents of the appropriate PAR is shifted and added to the Virtual PC to generate a physical 22 bit address. In this case the virtual PC corresponds to the non-relocated code.

The contents of the maintenance register will indicate what memory margin was being performed when the error occurred.

Depending on the type of error additional information is typed as described below.

5.1.1 Unexpected Trap to 4

PCOFTP    PHYSPC    PSW    CPUERR  
 VVVVVV    PPPPPPP    YYYYYY    ZZZZZZ

VVVVVV = Virtual PC that was pushed on the stack when the trap occurred.  
 PPPPPPP = Physical PC calculated as described above.  
 YYYYYY = PSW that was pushed on the stack.  
 ZZZZZZ = Contents of the CPU error register(1777766).

5.1.2 Unexpected Trap to 114

PCOFTP    PHYSPC    PSW    ERRREG    ERR ADR REG  
 VVVVVV    PPPPPPP    YYYYYY    ZZZZZZ    EEEEEEE

V, P, and Y = are the same as described in 5.1.1.  
 ZZZZZZ = Contents of the memory error register (777744).  
 EEEEEEE = Contents of the error address registers combined into a 22 bit address (777740 & 777742).

5.1.3 Parity Error During Data Check

This error can only occur during the data check that is made on the relocated test code before it is executed. This check is made by comparing the unrelocated code with the relocated

destination data to the relocated code.

SRCADR	DSTADR	EADDRREG	MEM ERR REG
SSSSSS	DDDDDDDD	EEEEEEEE	ZZZZZZ

SSSSSS = Virtual address of the source data.  
DDDDDDDD = Physical address of the destination data.  
EEEEEEEE = Contents of the error address registers.  
ZZZZZZ = Contents of memory error register (777744).

#### 3.1.4 Error During Data Check-Reloc was by CP

This error is similar to 6.1.3 except instead of a parity error, it is a data comparison error. Refer to section 9.5.3 for a description of CP relocation.

Loop on error (SW<9>) has the following effect:

1. Memory Management Off- If switch<9> is set, looping will be performed on the section relocation (see section 9.5.1). If SW<9> is not set, execution will continue at the beginning of the next section.
2. Memory Management On- If SW<9> is set, looping will be performed on the program relocation (see section 9.5.2) to the same memory space that failed. If SW<9> is not set, program relocation will be retried in the same memory space.

#### 3.1.5 Error During Data Check-Reloc was by I/O

This error is the same as 6.1.4 except relocation was performed via a disk rather than the CP. The error printout will identify which device and drive number transferred the particular word that failed. Refer to section 9.5.4 for a description of I/O relocation.

Loop on error (SW<9>) has the following effect:

1. If SW<9> is set, the device that relocated the word (that caused the data check error) is initiated to do the same transfer with the same disk address and memory addresses. This transfer will continually be initiated and checked until SW<9> is not set.

#### 3.1.6 Device Error

This error occurs if a device error occurs while the device is doing a transfer. The device and drive number are identified and the contents of the device registers are typed.

When SW<9> (loop on error) is set, the device that failed is continually restarted with the same disk address, memory address, and function that caused the error.

6.1.7 Unibus Exerciser Failed

CC	BUSADR	CR2	CR1	PHYS BUS ADR
XXXXXX	VVVVVV	WWWWWW	YYYYYY	ZZZZZZZZ

XXXXXX = Cycle count.  
 VVVVVV = Virtual bus address that the UBE failed at  
 WWWWWW = Control register number 2  
 YYYYYY = Control register number 1  
 ZZZZZZZZ = Physical memory address that the UBE failed at

The physical memory address is calculated by adding the appropriate map register to the virtual bus address, forming a real 22 bit memory address.

6.1.8 UBE Non-Existent Memory Error

This error only occurs when the "NO SLAVE SYNC" error occurs in the unibus exerciser. Only the physical address that timed out is typed. This error might indicate that there is a hole in memory or that the size register (777760) is set wrong.

6.1.9 Mass Bus Tester Failed

CS1	WRDCNT	BUSADR	BADREX	MR2	CS2	ST
AAAAAA	BBBBBB	CCCCCC	DDDDDD	EEEEEE	FFFFFF	GGGGGG

ER	CS3
HHHHHH	JJJJJJ

AAAAAA = Control and status register #1 (760100).  
 BBBBBB = Word count register (760102).  
 CCCCCC = Bus address register (760104).  
 DDDDDD = Bus address extended register (760174).  
 EEEEEEE = Maintenance register #2 (760106).  
 FFFFFFF = Control and status register #2 (760110).  
 GGGGGG = Status register (760112).  
 HHHHHH = Error register (760114).  
 JJJJJJ = Control and status register #3 (760176).

6.1.10 MBT Non-Existent Memory Error

This is the same as 6.1.7 except that it is detected by the NEXM bit in CS2 of the MBT.

6.1.11 Floating Point Error

This error will only occur if the left and right hand sides of the floating point identities do not agree within the expected tolerance. The value of the calculations are typed out.

This error should only be a function of the Floating Point Processor and the FPP diagnostics (DEFPA DEFPB) should be used

6.1.12 Device Hung

This error will occur if a device does not finish its relocation function within 2 seconds after its initiation. If a line clock is not installed, a hung device will hang the program. Refer to section 9.5.4.4 to determine which device and drive is hung.

6.2 Error Recovery

Different types of errors recover in different ways as described below.

6.2.1 Errors Within Subtests

Execution starts with the instruction following the error call.

6.2.2 Relocation with Memory Mgmt. Of

Execution starts at the beginning of the next section.

6.2.3 Device Error or CP Relocation with Memory Mgmt. On

Relocation is restarted.

6.2.4 Unexpected Traps Except Parity (4,10,250)

Execution starts at the address pointed to by location "SLPERR". This location contains the address+2 of the most recently executed "SCOPE" instruction.

6.2.5 Unexpected Parity Error

If the parity error is fatal (Bit 2 or 3 set in error reg) the program types a restart message at restarts. Otherwise, execution starts as in 6.2.4.

7.0 WARNINGS AND EXCEPTIONS

7.1 Warnings

Any drive that is not "write protected" will be written on (except unit 0 of the XXDP load device in chain mode).

When the program is dumped (see section F.3) and SW(7) is set, the devices and drives that are not write protected will be identified on the terminal. Before typing a character to continue, a drive can be write protected without causing an error because, the system is sized again.

-----  
If any of the devices is located at a non-standard address (see below), the device register address tables (in "common tags") should be changed to the correct addresses. Following is the default address of the control and status register of each device:

RPO3----176714  
RK05----177404  
RPO4----176700  
RS03/4--172040

If the system has both an RPO3 and an RPO4, the branch instruction at 1005, in the "size routine" must be replaced by a nop (240) for both devices to be used. This branch is approximately at address 4552.

## B.0 MISCELLANEOUS

-----

### B.1 Execution Time

-----

The execution time is dependent on the amount of memory on the system. Following are two typical run times:

1. Manufacturing Basic Line-32K memory, UBE, MBT, and no disks---3 minutes.
2. System-128K memory, 2 RK05's, RPO4, and 2 RS04's ---9 minutes.

### B.2 Stack Pointer

-----

The stack pointer is set to 700.

#### NOTE

When the program is running in either user or supervisor mode, the user/supervisor stack pointer is set to 700 and the Kernel stack pointer is set to 1200. The Kernel stack pointer is used only for the Error and Interrupt Service routines. routines.

### B.3 Pass Count

-----

There are two words used for effective pass count. Location "SUBPASS" and "SPASS". Subpass contains the ASCII representation of the subpass count. This is used to index

Six subpasses are executed for each pass. This allows all margins and PSW combinations to be tested before reporting end of pass.

At the end of each subpass the subpass number (that is being started) is typed followed by "THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789". If running on ACT11 QV or AA, only the sub-pass number is typed. At the end of each pass the elapsed run time and the message "END PASS X TOTAL ERRORS SINCE LAST REPORT Y" is typed.

#### B.4 Iterations

Sub-test iterations are not performed until the pass count (SPASS) is non-zero. This makes a QV pass as short as possible.

After the first pass, full 10 octal iterations are performed on each subtest.

#### B.5 T-Bit Trapping

T bit trapping is controlled by the PSW table. The default condition is to run with the T-Bit on during subpasses 2, 4, and 6.

#### B.5 ACT-11 Compatability

The program is fully ACT-11 compatible.

#### B.7 PSW and Margin Tables

At the end of the program, just before the messages, are the PSW and margin tables. These tables control what mode and register set and which memory margin will be executed on a subpass. Refer to section 9.5.2 for a description of how these tables are used by the program. These tables may be modified if desired.

#### B.8 I/O Device Address Modification

To modify the program address of the I/O devices patch the appropriate device table (in the common tags area) to the desired addresses.

If you are patching the RPO3 or RPO4 see section 7.2.

#### B.9 Power Fail

If a power fail occurs (followed by a power up), the word "POWER" is typed on the terminal and the program restarts.

9.0 PROGRAM DESCRIPTION  
-----

The program is divided into 9 sections of position independent relocatable test code. Each section is approximately 1K words long.

When the program is initially loaded and started it will identify itself and type the function of the switch register and the devices and drives that will be used for relocation, if SW7=0. It will also type the CP options available indicator word (OPT.CP). The contents of OPT.CP contain the following indicators:

Bit15	=	Not used
Bit14	=	Not used
Bit13=1/0	=	FPP available/not available
Bit12	=	Not used
Bit11	=	Not used
Bit10=1/0	=	MBT available/not available
Bit09=1/0	=	KW11-L available/not available
Bit08=1/0	=	Console tty available/not available
Bit07=1/0	=	UBE available/not available
Bits06-00	=	Not used

Following is a brief description of each section:

Section 0 This section causes a 256 word 3 Xor 9 test pattern to be relocated throughout memory 0 - 28K.

NOTE: This should not be construed to be a complete memory test.

Section 1 This section tests the unary instruction set executing each unary instruction in each address mode (excluding unary instructions using address mode 7).

Section 2 This section tests the unary instructions using address mode 7 and binaries in all address modes (excluding binary byte ops using address mode 7).

Section 3 This section tests binary byte ops using address mode 7, JMP, JSR and program trap (IOT, TRAP, and EMT) instruction.

Section 4 This section checks that each bit in the processor status word (PSW) can be set cleared, reserved instructions, and odd address traps.

Section 5 This section checks the SXT, XOR, SOB, MARK, RTT



Section 6 This section checks the ASH, ASHC, MUL, DIV, SPL instructions and the program interrupt request (PIRQ) logic.

Section 7 This section checks the stack limit register memory management abort logic, the memory management registers, and the mapping box registers.

Section 8 This section checks the floating point option, (FP11-B or FP11-C) if available.

Following section 8 are two routines to check the teletype printer logic and a routine to start the KW11-L clock. If the KW11-L is available the priority arbitration logic is tested.

#### 9.1 Micro-Break Test

-----

The micro-break test is used to test the micro break comparators and the stop on micro match function of the maintenance card. To run this test the operator must have a maintenance card installed and start the program at address 214.

The program asks the operator to turn on the stop on micro match switch. It then checks certain bit patterns in the micro break register to ensure the processor does not stop when it is not supposed to.

The processor will then stop with zero in the micro address lights. The operator then hits continue, and the processor will stop with one (1) in the lights. This sequence continues with 2, 4, 10, 20, 40, and 200 appearing in the lights. The program types done when it is finished.

#### 9.2 Unibus Exerciser(UBE)

-----

Any one of 4 UBE's will be used. The program looks for a UBE at addresses 17770004, 17770024, 17770034, and 17770044.

Test 75 will initiate the unibus exerciser if it is present. This is only done on pass 1 - subpass 1, since from that point on, the service routine takes care of restarting it.

The UBE is initially set up with a bus address of 0. The function that is loaded is "DATA IN PAUSE-DATA OUT BYTE". The word count is set for 4K bytes. It is also set to interrupt on level 5.

When an interrupt occurs a check is made to see if it was caused by an error. If there was no error, 776 is loaded as the bus address and the UBE is started again. On the next

sequence continues until a memory timeout error occurs.

When an error occurs a check is made to see if it was caused by a memory timeout. If it was, the address in the UBE bus address register is compared with the address in the system size registers. If they are the same (no holes in memory) the UBE is restarted at address 0 and the above sequence is repeated. If the addresses are not the same a memory-hole error is reported.

If the error was not due to a timeout a UBE error is reported.

### 9.3 Mass Bus Tester(MBT) -----

Any one of 4 MBT's will be used. The program looks for an MBT at addresses 17770100, 17770200, 17770300, and 17770400. If an MBT is found, the drive type register (17770X26) is checked to make sure that it really is an MBT.

Test 75 also initiates the mass bus tester. Again, this is only done on Pass 1 - subpass 1 since the service routine keeps it running.

The bus address register is initially set to 0, the word count to 2K words, and a read function is initiated.

When an interrupt occurs an error check is made. This error check is the same as that described for the UBE. If there was no error, the word count is reloaded and the function is issued. The bus address register is not changed so it will continue from where it left off.

### 9.4 Line Clock Initialization -----

Test 75 turns on the line clock. Two locations in "common tags" keep track of the elapsed run time of the program. When the clock interrupts, the low byte of location "lticks" is incremented. When this byte gets to 60(decimal) it is cleared and the high byte is incremented(seconds). When the second count gets to 60(decimal) location "mticks" is incremented and lticks is cleared. This gives the timer a 64K decimal minute range.

#### NOTE

For the UBE, MBT, and Line Clock, when an interrupt occurs, program execution returns to Kernel mode and the Kernel PAR's are mapped down to the 0-12K bank of memory. Upon returning from the interrupt the PAR's are mapped back to where they were and the previous

## 9.5 Relocation Algorithm

### 9.5.1 Section Relocation

As each section is entered the virtual start address is saved in location "FRSTAD" and the relocation factor (byte offset from non-relocated code) is calculated and saved in location "FACTOR". The test code is then executed.

At the end of each section, control is transferred to the "relocation routine". If SW<8> is set, this routine will relocate the section via the CP (see 9.5.3). If SW<8> is not set, the length of the section is calculated, saved as a word count, and control is transferred to the "I/O monitor" (see section 9.5.4) which relocates the section by using a disk.

Each section is initially relocated to the end address of the program. Subsequent relocations start at the end of the previous relocation. For example: if section 0 is 1000 bytes long and the end address of the program is 60000, the first relocation starts at address 60000, the second at 61000, the third at 62000, etc. This continues until 28K has been reached at which time execution goes to the start of the next section and the process repeats with the new section.

Each section is written in position independent code so that it can be relocated and executed without the use of memory management.

### 9.5.2 Program Relocation

When all nine sections have been relocated and executed thru 28K (see section 9.5.1), memory management is setup according to the value in location "NEXPAR". This value is initialized to 600 (or 1600 if running under the XXDP monitor), making relocation start at address 60000 (or 160000). The "I/O monitor" is then entered (see section 9.5.4) to relocate the program. When the I/O monitor completes the relocation, execution is transferred to the start of the program at the relocated position.

Each section is executed only once with memory management on. At the end of section 8, 77 is added to "NEXPAR" and relocation is performed again. This causes the next relocation to move up by 7700 bytes. For example: If nexpar=1600 the first relocation starts at address 160000, the second at address 167700, the third at 177600, etc.

This continues until the end of memory is reached and constitutes a sub-pass. The PSW and maintenance register (for memory margins) are then setup for the next sub-pass and the program restarts.

the tables (see section 8.7). The particular entry that is used is obtained by indexing the table by the sub-pass number (see section 8.3). For example, sub-pass 3 uses word 3 (the first word is counted as zero) of each table. Therefore, to change the value in the PSW or maintenance register only requires changing the value in the appropriate table.

The completion of 6 sub-passes constitutes a pass and an end of pass message is typed. The program then restarts in pass 2, sub-pass 0.

### 9.5.3 Relocation VIA CP

If SW(B) is set, both section and program relocation (see sections 9.5.1 and 9.5.2), are performed by an instruction move loop rather than a disk. For example:

```
IS: MOV (R0)+,(R2)+
      CMP R0,R3
      BNE IS
```

where R0 is the address of the code being moved, R2 is the address that it is being moved to, and R3 is the last address that is to be moved.

When this is finished, the relocated data is checked by an instruction compare loop to ensure that the relocation was performed correctly.

### 9.5.4 Relocation VIA I/O

If SW(B) is not set, both section and program relocation (see section 9.5.1 and 9.5.2), are performed by writing the data to a disk and reading it back to the relocated position. This relocation is controlled by the "I/O Monitor".

#### 9.5.4.1 Section Relocation

When the I/O monitor is entered from the "relocation routine" (see section 9.5.1) a device is selected (see 9.5.4.3), the memory addresses (from and to) and word count are passed to the device handler (see section 9.5.4.4) and the handler is called. When the handler finishes, the I/O monitor checks the relocated data with an instruction compare loop to ensure the relocated data is correct, and returns to the "relocation routine" (see 9.5.1).

#### 9.5.4.2 Program Relocation

When the I/O monitor is entered for program relocation (see section 9.5.2) the base address for the relocation is calculated from the contents of kernel par3 which was set up with memory management (see 9.5.2). If SW(B) is set,

If SW<8> is not set, a device is selected (see 9.5.4.3), the word count is set to 2K, and the memory addresses (from and to) and word count are passed to the device handler (see 9.5.4.4) and the handler is called. The I/O monitor then adds 2K to the memory addresses, selects another device, passes the addresses to the device handler, and calls the handler. This continues until all 12K has been relocated. The relocated data is then checked with an instruction compare loop. The relocated program is then executed as described in 9.5.2.

#### 9.5.4.3 Device Selection

If SW<5> is not set, an index is picked up from location "DEVINDEX". This index is used to index the system size table. The system size table consists of 8 words (one for each device type). Bits <7:0> of each word are used to indicate the drive numbers that are available on the device, and are initialized in the size routine. Bits <15:8> of each word are used to indicate whether the drive has been used for a data transfer (unit used bit).

The system size table is then searched, using the index described above, for a drive that has not been used. When a drive is found, the "unit used bit" is set, the current index is put back in location DEVINDEX, and execution continues as described in 9.5.4.1 or 9.5.4.2.

If an unused unit is not found, all the "unit used" bits are cleared and the search is restarted. If the search finds the system size table empty (no devices on the system), the message "NO I/O DEVICES" is typed and relocation is performed via the CP as described in 9.5.3.

If SW<5> is set, SW's<2:0> are used to index the system size table. In this case only one word of the table is used corresponding to the device being selected by SW's<2:0> (see section 5.1). In this mode, a round robin selection is performed on the drives of the selected device.

#### 9.5.4.4 Device Handlers

Each device that is used for relocation has a handler. These handlers are functionally the same.

The handler is called by the I/O Monitor (see section 9.5.4). It first clears the done bit (bit 7) in the handler status word. This prevents the monitor from calling this handler again before it is finished.

If a "device hung" error (see section 6.1.12) is detected, the handler status words can be examined to determine which device did not finish (set bit 7). The drive can then be determined by looking in the "device handler unit number" table. The

are located in the "common tags" area of the listing.

Then the handler calculates a disk address. This address is either generated from a random number (SW4=0) or is set to zero (SW4=1). The device ID, unit number, and cylinder address are combined and placed in the "RUN TABLE" (RONTBL). The position in the run table corresponds to which 2K block of the program is being transferred (i.e. the first 2K block is identified by word 1, the second 2K by word 2, etc.). The bit configuration of each word in the run table is as follows:

<15:13> = Device ID  
<12:10> = Unit Number  
<9> = not used  
<8:0> = Cylinder Address

The track-sector address of the transfer is saved in the "RUN TRACK TABLE" (RUNTRAK). The position in this table is as described above. The bit configuration of each word is the same as that for the disk address register for the particular device. Bit 15 is used to indicate a device error. It is set by the device service routine. (see section 9.5.4.5)

The handler then initializes the device registers with all the appropriate information and starts a write function. Execution then returns to the I/O Monitor at the point where the handler was called.

#### 9.5.4.5 Device Service Routines

Each device that is used for relocation has a service routine. These routines are all functionally the same.

The routine is entered by a device interrupt. The device is checked for any errors. If no error occurred the device registers are loaded and the next function to perform is initiated. Three functions are executed: Write, Write Check, and Read. All the necessary bus address information is calculated by the I/O Monitor, so the service routine just takes care of the device.

When the read function has been completed successfully, the done bit (bit 7) in the handler status word is set.

Upon initiation of a function, or completion of all three functions, the service routine returns execution to where it was when it was interrupted.

If an error is detected, the function that failed is retried two more times. If the error is still present the done bit and the error bit (bit 15) is set in the handler status word along with bit 15, in the appropriate entry, in the RUN TRACK



UNDELETABLE INFORMATION

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT USE
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	INHIBIT RELOCATION VIA I/O DEVICE
7	INHIBIT SYSTEM SIZE TYPEOUT
6	INHIBIT RELOCATION
5	INHIBIT ROUND ROBIN
4	INHIBIT RANDOM DISK ADDRESS
3	INHIBIT MBT
2	THESE THREE SWITCHES
1	ARE ENCODED TO SELECT RELOCATION
0	ON THE FOLLOWING DEVICES:
0...	RP11/RP03
1...	RK11/RK05
2...	NOT USED
3...	NOT USED
4...	RH70/RP04
5...	RH70/RS04
6...	NOT USED



71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125

.SBTTL BASIC DEFINITIONS

.\*INITIAL ADDRESS OF THE STACK POINTER \*\*\* 1200 \*\*\*  
STACK= 1200 ;: FIRST ADDRESS OF THE STACK  
KERSTK= STACK ;: KERNEL STACK  
SUPSTK= STACK-300 ;: SUPERVISOR STACK  
USESTK= STACK-400 ;: USER STACK  
.EQUIV EMT,ERROR ;: BASIC DEFINITION OF ERROR CALL  
.EQUIV IOT,SCOPE ;: BASIC DEFINITION OF SCOPE CALL  
PS= 177776 ;: PROCESSOR STATUS WORD  
.EQUIV PS,PSW  
STKLM= 177774 ;: STACK LIMIT REGISTER  
PIRQ= 177772 ;: PROGRAM INTERRUPT REQUEST REGISTER  
SWR= 177570 ;: SWITCH REGISTER  
DISPLAY=SWR

.\*MISCELLANEOUS DEFINITIONS

HT= 11 ;: CODE FOR HORIZONTAL TAB  
LF= 12 ;: CODE LINE FEED  
CR= 15 ;: CODE CARRIAGE RETURN  
CRLF= 200 ;: CODE FOR CARRIAGE RETURN-LINE FEED

.\*GENERAL PURPOSE REGISTER DEFINITIONS

R0= %0 ;: GENERAL REGISTER  
R1= %1 ;: GENERAL REGISTER  
R2= %2 ;: GENERAL REGISTER  
R3= %3 ;: GENERAL REGISTER  
R4= %4 ;: GENERAL REGISTER  
R5= %5 ;: GENERAL REGISTER  
R6= %6 ;: GENERAL REGISTER  
R7= %7 ;: GENERAL REGISTER  
.EQUIV R0,R10 ;: GENERAL REGISTER  
.EQUIV R1,R11 ;: GENERAL REGISTER  
.EQUIV R2,R12 ;: GENERAL REGISTER  
.EQUIV R3,R13 ;: GENERAL REGISTER  
.EQUIV R4,R14 ;: GENERAL REGISTER  
.EQUIV R5,R15 ;: GENERAL REGISTER  
SP=%6 ;: STACK POINTER  
.EQUIV SP,KSP ;: KERNEL STACK POINTER  
.EQUIV SP,SSP ;: SUPERVISOR STACK POINTER  
.EQUIV SP,USP ;: USER STACK POINTER  
PC=%7 ;: PROGRAM COUNTER

.\*PRIORITY LEVEL DEFINITIONS

PR0= 0 ;: PRIORITY LEVEL 0  
PR1= 40 ;: PRIORITY LEVEL 1  
PR2= 100 ;: PRIORITY LEVEL 2  
PR3= 140 ;: PRIORITY LEVEL 3  
PR4= 200 ;: PRIORITY LEVEL 4  
PR5= 240 ;: PRIORITY LEVEL 5  
PR6= 300 ;: PRIORITY LEVEL 6  
PR7= 340 ;: PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS

127	040000	SW14=	40000
128	020000	SW13=	20000
129	010000	SW12=	10000
130	004000	SW11=	4000
131	002000	SW10=	2000
132	001000	SW09=	1000
133	000400	SW08=	400
134	000200	SW07=	200
135	000100	SW06=	100
136	000040	SW05=	40
137	000020	SW04=	20
138	000010	SW03=	10
139	000004	SW02=	4
140	000002	SW01=	2
141	000001	SW00=	1
142		.EQUIV	SW09, SW9
143		.EQUIV	SW08, SW8
144		.EQUIV	SW07, SW7
145		.EQUIV	SW06, SW6
146		.EQUIV	SW05, SW5
147		.EQUIV	SW04, SW4
148		.EQUIV	SW03, SW3
149		.EQUIV	SW02, SW2
150		.EQUIV	SW01, SW1
151		.EQUIV	SW00, SW0

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

154	100000	BIT15=	100000
155	040000	BIT14=	40000
156	020000	BIT13=	20000
157	010000	BIT12=	10000
158	004000	BIT11=	4000
159	002000	BIT10=	2000
160	001000	BIT09=	1000
161	000400	BIT08=	400
162	000200	BIT07=	200
163	000100	BIT06=	100
164	000040	BIT05=	40
165	000020	BIT04=	20
166	000010	BIT03=	10
167	000004	BIT02=	4
168	000002	BIT01=	2
169	000001	BIT00=	1
170		.EQUIV	BIT09, BIT9
171		.EQUIV	BIT08, BIT8
172		.EQUIV	BIT07, BIT7
173		.EQUIV	BIT06, BIT6
174		.EQUIV	BIT05, BIT5
175		.EQUIV	BIT04, BIT4
176		.EQUIV	BIT03, BIT3
177		.EQUIV	BIT02, BIT2
178		.EQUIV	BIT01, BIT1
179		.EQUIV	BIT00, BIT0

.\*BASIC "CPU" TRAP VECTOR ADDRESSES

180  
181

183	000010	RESVEC= 10	:: RESERVED AND ILLEGAL INSTRUCTIONS
184	000014	TBITVEC=14	:: "T" BIT
185	000014	TRTVEC= 14	:: TRACE TRAP
186	000014	BPTVEC= 14	:: BREAKPOINT TRAP (BPT)
187	000020	IOTVEC= 20	:: INPUT/OUTPUT TRAP (IOT) **SCOPE**
188	000024	PWRVEC= 24	:: POWER FAIL
189	000030	EMTVEC= 30	:: EMULATOR TRAP (EMT) **ERROR**
190	000034	TRAPVEC=34	:: "TRAP" TRAP
191	000060	TKVEC= 60	:: TTY KEYBOARD VECTOR
192	000064	TPVEC= 64	:: TTY PRINTER VECTOR
193	000114	CACHVEC=114	:: CACHE ERROR INTERRUPT VECTOR
194	000240	PIRQVEC=240	:: PROGRAM INTERRUPT REQUEST VECTOR
195	000250	MMVEC= 250	:: MEMORY MANAGEMENT VECTOR

.SBTTL CACHE REGISTER DEFINITIONS

200	177740	LOADRS = 177740	:: LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
201	177742	HIADRS = 177742	:: UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
202	177744	MEMERR = 177744	:: CACHE ERROR REGISTER
203	177746	CONTRL = 177746	:: MEMORY CONTROL REGISTER
204	177750	MAINT = 177750	:: MEMORY MAINTENANCE REGISTER
205	177752	HITMIS = 177752	:: HIT MISS REGISTER "1" IMPLIES HIT IN CACHE

.SBTTL CPU REGISTER DEFINITIONS

211	177760	SIZELO = 177760	:: MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR TO GET TO THE LAST 32 WORDS OF MEMORY
212	177762	SIZEHI = 177762	:: HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
213			:: CURRENTLY ALL ZERO
214			:: SYSTEM ID REGISTER
215	177764	SYSTID = 177764	:: SYSTEM ID REGISTER
216	177766	CPUERR = 177766	:: CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED THE TRAP TO ERRVEC (000004)

.SBTTL MEMORY MANAGEMENT DEFINITIONS

;\*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

227	177572	MMR0= 177572
228	177574	MMR1= 177574
229	177576	MMR2= 177576
230	172516	MMR3= 172516
231		.EQUIV MMR0,SR0
232		.EQUIV MMR1,SR1
233		.EQUIV MMR2,SR2
234		.EQUIV MMR3,SR3

;\*USER "I" PAGE DESCRIPTOR REGISTERS

235  
236  
237

239	177602	UIPDR1= 177602
240	177604	UIPDR2= 177604
241	177606	UIPDR3= 177606
242	177610	UIPDR4= 177610
243	177612	UIPDR5= 177612
244	177614	UIPDR6= 177614
245	177616	UIPDR7= 177616

;\*USER "D" PAGE DESCRIPTOR REGISTORS

248	177620	UDPDR0= 177620
249	177622	UDPDR1= 177622
250	177624	UDPDR2= 177624
251	177626	UDPDR3= 177626
252	177630	UDPDR4= 177630
253	177632	UDPDR5= 177632
254	177634	UDPDR6= 177634
255	177636	UDPDR7= 177636

;\*USER "I" PAGE ADDRESS REGISTERS

258	177640	UIPAR0= 177640
259	177642	UIPAR1= 177642
260	177644	UIPAR2= 177644
261	177646	UIPAR3= 177646
262	177650	UIPAR4= 177650
263	177652	UIPAR5= 177652
264	177654	UIPAR6= 177654
265	177656	UIPAR7= 177656

;\*USER "D" PAGE ADDRESS REGISTERS

268	177660	UDPAR0= 177660
269	177662	UDPAR1= 177662
270	177664	UDPAR2= 177664
271	177666	UDPAR3= 177666
272	177670	UDPAR4= 177670
273	177672	UDPAR5= 177672
274	177674	UDPAR6= 177674
275	177676	UDPAR7= 177676

;\*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS

278	172200	SIPDR0= 172200
279	172202	SIPDR1= 172202
280	172204	SIPDR2= 172204
281	172206	SIPDR3= 172206
282	172210	SIPDR4= 172210
283	172212	SIPDR5= 172212
284	172214	SIPDR6= 172214
285	172216	SIPDR7= 172216

;\*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS

288	172220	SDPDR0= 172220
-----	--------	----------------

295	172224	SDPOR2= 172224
296	172226	SDPOR3= 172226
297	172230	SDPOR4= 172230
298	172232	SDPOR5= 172232
299	172234	SDPOR6= 172234
300	172236	SDPOR7= 172236

;\*SUPERVISOR "I" PAGE ADDRESS REGISTERS

304	172240	SIPAR0= 172240
305	172242	SIPAR1= 172242
306	172244	SIPAR2= 172244
307	172246	SIPAR3= 172246
308	172250	SIPAR4= 172250
309	172252	SIPAR5= 172252
310	172254	SIPAR6= 172254
311	172256	SIPAR7= 172256

;\*SUPERVISOR "D" PAGE ADDRESS REGISTERS

315	172260	SDPAR0= 172260
316	172262	SDPAR1= 172262
317	172264	SDPAR2= 172264
318	172266	SDPAR3= 172266
319	172270	SDPAR4= 172270
320	172272	SDPAR5= 172272
321	172274	SDPAR6= 172274
322	172276	SDPAR7= 172276

;\*KERNEL "I" PAGE DESCRIPTOR REGISTERS

326	172300	KIPDR0= 172300
327	172302	KIPDR1= 172302
328	172304	KIPDR2= 172304
329	172306	KIPDR3= 172306
330	172310	KIPDR4= 172310
331	172312	KIPDR5= 172312
332	172314	KIPDR6= 172314
333	172316	KIPDR7= 172316

;\*KERNEL "D" PAGE DESCRIPTOR REGISTERS

337	172320	KDPDR0= 172320
338	172322	KDPDR1= 172322
339	172324	KDPDR2= 172324
340	172326	KDPDR3= 172326
341	172330	KDPDR4= 172330
342	172332	KDPDR5= 172332
343	172334	KDPDR6= 172334
344	172336	KDPDR7= 172336

;\*KERNEL "I" PAGE ADDRESS REGISTERS

348	172340	KIPAR0= 172340
349	172342	KIPAR1= 172342

351 172346  
352 172350  
353 172352  
354 172354  
355 172356

KIPAR3 = 172346  
KIPAR4 = 172350  
KIPAR5 = 172352  
KIPAR6 = 172354  
KIPAR7 = 172356

;\*KERNEL "D" PAGE ADDRESS REGISTERS

358 172360  
359 172362  
360 172364  
361 172366  
362 172370  
363 172372  
364 172374  
365 172376

KDPAR0 = 172360  
KDPAR1 = 172362  
KDPAR2 = 172364  
KDPAR3 = 172366  
KDPAR4 = 172370  
KDPAR5 = 172372  
KDPAR6 = 172374  
KDPAR7 = 172376

.SBTTL UNIBUS MAP REGISTER DEFINITIONS

;\*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'  
;\*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'

378 170200  
379 170202  
380 170204  
381 170206  
382 170210  
383 170212  
384 170214  
385 170216  
386 170220  
387 170222  
388 170224  
389 170226  
390 170230  
391 170232  
392 170234  
393 170236  
394 170240  
395 170242  
396 170244  
397 170246  
398 170250  
399 170252  
400 170254  
401 170256  
402 170260  
403 170262  
404 170264  
405 170266

MAPL00 = 170200  
MAPH00 = 170202  
MAPL01 = 170204  
MAPH01 = 170206  
MAPL02 = 170210  
MAPH02 = 170212  
MAPL03 = 170214  
MAPH03 = 170216  
MAPL04 = 170220  
MAPH04 = 170222  
MAPL05 = 170224  
MAPH05 = 170226  
MAPL06 = 170230  
MAPH06 = 170232  
MAPL07 = 170234  
MAPH07 = 170236  
MAPL10 = 170240  
MAPH10 = 170242  
MAPL11 = 170244  
MAPH11 = 170246  
MAPL12 = 170250  
MAPH12 = 170252  
MAPL13 = 170254  
MAPH13 = 170256  
MAPL14 = 170260  
MAPH14 = 170262  
MAPL15 = 170264  
MAPH15 = 170266

407	170272	MAPH16	=	170272
408	170274	MAPL17	=	170274
409	170276	MAPH17	=	170276
410	170300	MAPL20	=	170300
411	170302	MAPH20	=	170302
412	170304	MAPL21	=	170304
413	170306	MAPH21	=	170306
414	170310	MAPL22	=	170310
415	170312	MAPH22	=	170312
416	170314	MAPL23	=	170314
417	170316	MAPH23	=	170316
418	170320	MAPL24	=	170320
419	170320	MAPH24	=	170320
420	170324	MAPL25	=	170324
421	170326	MAPH25	=	170326
422	170330	MAPL26	=	170330
423	170332	MAPH26	=	170332
424	170334	MAPL27	=	170334
425	170336	MAPH27	=	170336
426	170340	MAPL30	=	170340
427	170342	MAPH30	=	170342
428	170344	MAPL31	=	170344
429	170346	MAPH31	=	170346
430	170350	MAPL32	=	170350
431	170352	MAPH32	=	170352
432	170354	MAPL33	=	170354
433	170356	MAPH33	=	170356
434	170360	MAPL34	=	170360
435	170362	MAPH34	=	170362
436	170364	MAPL35	=	170364
437	170366	MAPH35	=	170366
438	170370	MAPL36	=	170370
439	170372	MAPH36	=	170372
440	170374	MAPL37	=	170374
441	170376	MAPH37	=	170376
442		.EQUIV		MAPL00, MAPL0
443		.EQUIV		MAPH00, MAPH0
444		.EQUIV		MAPL01, MAPL1
445		.EQUIV		MAPH01, MAPH1
446		.EQUIV		MAPL02, MAPL2
447		.EQUIV		MAPH02, MAPH2
448		.EQUIV		MAPL03, MAPL3
449		.EQUIV		MAPH03, MAPH3
450		.EQUIV		MAPL04, MAPL4
451		.EQUIV		MAPH04, MAPH4
452		.EQUIV		MAPL05, MAPL5
453		.EQUIV		MAPH05, MAPH5
454		.EQUIV		MAPL06, MAPL6
455		.EQUIV		MAPH06, MAPH6
456		.EQUIV		MAPL07, MAPL7
457		.EQUIV		MAPH07, MAPH7
458				
459				
460				
461				

E03

463 000001  
464 000002  
465 000003  
466 000004  
467 000005  
468  
469 172540  
470 172542

AC1= %1  
AC2= %2  
AC3= %3  
AC4= %4  
AC5= %5

;LINE CLOCK AND PROGRAMMABLE LINE CLOCK REGISTERS  
PLKCSR=172540  
FLKCSB=172542



472 177546  
 473 000100  
 474  
 475  
 476 170000  
 477 170002  
 478 170004  
 479 170006  
 480 170010  
 481 170014  
 482 170016  
 483 000510  
 484  
 485  
 486 160100  
 487 160102  
 488 160104  
 489 160106  
 490 160110  
 491 160112  
 492 160114  
 493 160116  
 494 160120  
 495 160124  
 496 160126  
 497 160174  
 498 160176  
 499 000774  
 500 000776  
 501  
 502  
 503 100000  
 504 040000  
 505 020000  
 506 002000  
 507 001000  
 508 000400  
 509 000200  
 510  
 511  
 512  
 513  
 514 000010  
 515 000000  
 516 140000  
 517 000000  
 518 030000  
 519 177770  
 520  
 521  
 522  
 523 000000  
 524  
 525  
 526

LKS=177546  
 LKVEC=100  
 ;UNIBUS EXERCISOR REGISTER  
 UBEOB= 170000 ; DATA BUFFER  
 UBECCL= 170002 ; CYCLE COUNT  
 UBEBAL= 170004 ; BUS ADDRESS  
 UBECR1= 170006 ; CONTROL REGISTER 1  
 UBECRL= 170010 ; ERROR CLEAR  
 UBEGO= 170014 ; MULTI-EXERCISOR GO  
 UBECR2= 170016 ; CONTROL REGISTER 2  
 UBVECL= 510 ; INTERRUPT VECTOR  
 ; MASS BUS TESTER REGISTERS  
 MBTCS1= 160100  
 MBTWC= 160102  
 MBTBA= 160104  
 MBTMR2= 160106  
 MBTCS2= 160110  
 MBTST= 160112  
 MBTER= 160114  
 MBTAS= 160116  
 MBTDB= 160120  
 MBTMR1= 160124  
 MBTDT= 160126  
 MBTBAE= 160174  
 MBTCS3= 160176  
 MBTVEC= 774  
 MBTPSW= 776  
 ; MISCELLANEOUS BIT ASSIGNMENTS (USED IN OPT.CP)  
 KTOPT= 100000 ; BELOW BIT ASSIGNMENTS ARE USED  
 EISOPT= 040000 ; IN THE CPCHK ROUTINE  
 FPOPT= 020000 ; A BIT FOR EACH OPTION PRESENT  
 MBTOPT= 002000  
 LKOPT= 001000  
 TTOPT= 000400  
 UBEOPT= 000200  
 .EQUIV ERROR,HLT  
 .EQUIV BIT14,SM  
 .EQUIV BIT12,PSM  
 .EQUIV BIT11,REG  
 CALLHANDLER=10  
 KM=0  
 UM=140000  
 PKM=0  
 PUM=30000  
 UBREAK=177770  
 .SBTTL TRAP CATCHER  
 . = 0  
 ; \*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ". +2,HALT"  
 ; \*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS  
 ; \*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

```

528
529
530
531 000200 000137 003212
532 000210
533 000210 000137 002464
534 000214 000137 002474
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555 000220
556 000046 000046
557 000052 036752
558 000052 000052
559 000052 040000
560 000220

```

```

.SBTTL STARTING ADDRESS(ES)
.=200
JMP @START ;;JUMP TO STARTING ADDRESS OF PROGRAM
.=210
JMP @START1
JMP @START2
;*****
.SBTTL ACT11 HOOKS
;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
;*
;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL
;*END OF THE PROGRAM.
;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
;*
;* BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
;* =0 NO POWER FAIL DESIRED
;*
;* BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
;* =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
;*
;* BITS 13-0 MUST BE ZERO'S
;*
$SVPC=. ;;SAVE LOCATION COUNTER
.=46 ;;SET LOCATION COUNTER
.WORD SENDAD ;;SET LOC.46 TO ADDRESS SENDAD
.=52 ;;SET LOCATION COUNTER
.WORD 40000 ;;SET LOC.52 TO 40000
.$SVPC ;;RESTORE LOCATION COUNTER

```

```

562 ;*****
563
564 .SBTTL COMMON TAGS
565
566 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
567 ;*USED IN THE PROGRAM.
568
569 001200 .=1200
570
571 001200 SCMTAG: ; START OF COMMON TAGS
572 001200 000000 $PASS: .WORD 0 ; CONTAINS PASS COUNT
573 001202 000000 $TSTNM: .WORD 00 ; CONTAINS THE TEST NUMBER
574 001204 000 SERFLG: .BYTE 0 ; CONTAINS ERROR FLAG
575 001206 .EVEN
576 001206 000000 $ICNT: .WORD 0 ; CONTAINS SUBTEST ITERATION COUNT
577 001210 000000 $LPAOR: .WORD 00 ; CONTAINS SCOPE LOOP 1200
578 001212 000000 $LPERR: .WORD 00 ; CONTAINS SCOPE RETURN FOR ERRORS
579 001214 000000 $ERTTL: .WORD 00 ; CONTAINS TOTAL ERRORS DETECTED
580 001216 000 $ITEMB: .BYTE 0 ; CONTAINS ITEM CONTROL BYTE
581 001217 001 $ERMAX: .BYTE 1 ; CONTAINS MAX. ERRORS PER TEST
582 001220 000000 $ERRPC: .WORD 0 ; CONTAINS PC OF LAST ERROR INSTRUCTION
583 001222 000000 $GDADR: .WORD 00 ; CONTAINS 1200 OF 'GOOD' DATA
584 001224 000000 $BDAOR: .WORD 00 ; CONTAINS 1200 OF 'BAD' DATA
585 001226 000000 $GDDAT: .WORD 00 ; CONTAINS 'GOOD' DATA
586 001230 000000 $BDDAT: .WORD 00 ; CONTAINS 'BAD' DATA
587 001232 000000 000000 000000 $RESV: .WORD 0,0,0 ; RESERVED--NOT TO BE USED
588 001240 177560 $TKS: 177560 ; TTY KBD STATUS
589 001242 177562 $TKB: 177562 ; TTY KBD BUFFER
590 001244 177564 $TPS: 177564 ; TTY PRINTER STATUS REG. 1200
591 001246 177566 $TPB: 177566 ; TTY PRINTER BUFFER REG. 1200
592 001250 000 $NULL: .BYTE 0 ; CONTAINS NULL CHARACTER FOR FILLS
593 001251 002 $FILLS: .BYTE 2 ; CONTAINS # OF FILLER CHARACTERS REQUIRED
594 001252 012 $FILLC: .BYTE 12 ; INSERT FILL CHARS. AFTER A "LINE FEED"
595 001253 000 $TPFLG: .BYTE 0 ; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
596 001254 000000 $REGAD: .WORD 0 ; CONTAINS THE 1200 FROM
597 ; WHICH ($REGO) WAS OBTAINED
598 001256 000000 $REG0: .WORD 0 ; CONTAINS (($REGAD)+0)
599 001260 000000 $REG1: .WORD 00 ; CONTAINS (($REGAD)+2)
600 001262 000000 $REG2: .WORD 00 ; CONTAINS (($REGAD)+4)
601 001264 000000 $REG3: .WORD 00 ; CONTAINS (($REGAD)+6)
602 001266 000000 $REG4: .WORD 00 ; CONTAINS (($REGAD)+10)
603 001270 000000 $REG5: .WORD 00 ; CONTAINS (($REGAD)+12)
604 001272 000000 $REG6: .WORD 00 ; CONTAINS (($REGAD)+14)
605 001274 000000 $REG7: .WORD 00 ; CONTAINS (($REGAD)+16)
606 001276 000000 $TMP0: .WORD 00 ; USER DEFINED
607 001300 000000 $TMP1: .WORD 00 ; USER DEFINED
608 001302 000000 $TMP2: .WORD 00 ; USER DEFINED
609 001304 000000 $TMP3: .WORD 00 ; USER DEFINED
610 001306 000000 $TMP4: .WORD 00 ; USER DEFINED
611 001310 000000 $TMP5: .WORD 00 ; USER DEFINED
612 001312 000000 $TMP6: .WORD 00 ; USER DEFINED
613 001314 000000 $TMP7: .WORD 0 ; USER DEFINED
614 001316 000000 $TIMES: 0 ; MAX. NUMBER OF ITERATIONS
615 001320 000000 $ESCAPE: 0 ; ESCAPE ON ERROR 1200
616 001322 177607 000377 $BELL: .ASCIZ \207\377\377 ; CODE FOR BELL

```

618	001327	015				\$CRLF:	.ASCII	<15>		::CARRIAGE RETURN
619	001330	000012				\$LF:	.ASCII	<12>		::LINE FEED
620	001332	000000				ERRRTN:	.WORD			
621	001334	000044				\$FLBUFF:	.BLKB	44		::BUFFER FOR FLOATING POINT CONVERSION
622	001400	000000				\$BUFF:	.WORD			
623	001402	000000				\$ACO:	.WORD			::EXTENDED EXPONENT VALUES
624	001404	000000				\$AC1:	.WORD			::FOR THE SIX FLOATING POINT
625	001406	000000				\$AC2:	.WORD			::ACCUMULATORS
626	001410	000000				\$AC3:	.WORD			
627	001412	000000				\$AC4:	.WORD			
628	001414	000000				\$AC5:	.WORD			
629	001416	000000				\$STMP4:	.WORD			
630	001420	000000				\$STMP6:	.WORD			
631	001422	000004				FLTMP0:	.BLKW	4		::FLOATING POINT DBL PREC BUFFER
632	001432	000004				FLTMP1:	.BLKW	4		
633	001442	001444				TKBFRP:	.WORD	TKBFR		::POINTER FOR KEYBOARD BUFFER
634	001444	000011				TKBFR:	.BLKW	11		::KEYBOARD BUFFER
635	001466	000000				NOTYPE:	.WORD			::NO TYPEOUT FLAG (INHIBIT WHEN SET)
636	001470	000000				OPT_CP:	.WORD			::CPU OPTION FLAGS
637	001472	000006				\$RTN:	.RTT			::RETURN FOR T-BIT TRAP
638	001474	000000				VADR:	.WORD			::BUFFER FOR VIRTUAL ADDRESS
639	001476	000000				PA1500:	.WORD			::BUFFER FOR PHYSICAL ADDRESS BITS<15:00>
640	001500	000000				PA2116:	.WORD			::PHYSICAL ADDRESS BITS<21:16>
641	001502	000				NEXEC:	.BYTE			::NO EXECUTE FLAG(NO TEST EXECUTION WHEN SET)
642	001503	000				MMON:	.BYTE			::MEMORY MGMT FLAG(MGMT IS ON WHEN NON-ZERO)
643	001504	000				QV:	.BYTE			::QV FLAG(QV PASS WHEN SET)
644	001505	000				AA:	.BYTE			::AUTO ACCEPT FLAG (AA PASS WHEN SET)
645	001506	000000				FACTOR:	.WORD			::RELOCATION FACTOR(NUMBER OF
646	001510	000000				\$FACTOR:	.WORD			::BYTES ABOVE BASE CODE)
647	001512	000000				FRSTAD:	.WORD			::FIRST ADDRESS OF SECTION BEING EXECUTED
648	001514	000000				FRSTMEM:	.WORD			::ADDRESS OF FIRST FREE MEMORY
649	001516	000000				LSTMEM:	.WORD			::ADDRESS OF LAST FREE MEMORY(IN 2BK)
650	001520	000000				NEXPAR:	.WORD			::NEXT VALUE TO PUT IN PAR0
651	001522	123456				\$LNUM:	.WORD	123456		::LOW 16 BITS OF RANDOM NUMBER
652	001524	065432				\$HNUM:	.WORD	65432		::HIGH 16 BITS OF RANDOM NUMBER
653	001526	001	001	001		NULLS:	.BYTE	1,1,1,0		::BUFFER FOR PRINTER TEST
654	001531	000								
655	001532	000060				SUBPASS:	.WORD	60		::SUB-PASS COUNT IN ASCII
656	001534	000000				\$ERPSW:	.WORD			::ERROR PSW FOR TYPEOUT
657	001536	000000				EXITFL:	.WORD			
658	001540	000000				OLDBASE:	.WORD			::SOURCE BASE ADDRESS FOR DEVICE RELOCATION
659	001542	000000				NWBASL:	.WORD			::DEST ADDRESS FOR DEVICE RELOC BITS<15:00>
660	001544	000000				NWBASH:	.WORD			::DEST ADDRESS FOR DEVICE RELOC BITS<21:16>
661	001546	000000				IOWC:	.WORD			::TWO'S COMPLIMENT WORD COUNT FOR DEVICE RELOC
662	001550	000000				DEVICE:	.WORD			
663	001552	000000				DEVINOX:	.WORD			::DEVICE INDEX (0 TO 7)
664	001554	000000				UNITNO:	.WORD			::DEVICE UNIT NUMBER
665	001556	000000				RNTBINX:	.WORD			::INDEX TO RUN TABLE
666	001560	000000				MXMMHI:	.WORD			::BITS<21:16> OF LAST MEM ADDRESS ON SYSTEM
667	001562	000000				MXMML0:	.WORD			::BITS<15:00> OF LAST MEM ADDRESS ON SYSTEM
668	001564	000000				RP310:	.WORD			::DATA TO LOAD INTO RPO3 CS REGISTER
669	001566	000000				RP311:	.WORD			::RPO3 FLAG FOR FIRST 2K OF PROGRAM
670	001570	000000				RK10:	.WORD			::DATA TO LOAD INTO RK05 CS REGISTER
671	001572	000000				RK11:	.WORD			::RK05 FLAG FOR FIRST 2K OF PROGRAM
672	001574	000000				RP411:	.WORD			::RPO4 FLAG FOR FIRST 2K OF PROGRAM

```

674 001600 000000 MTICKS: .WORD ;ELAPSED RUN TIME IN MINUTES
675 001602 000000 LTICKS: .WORD ;LOW BYTE=NUMBER OF CLOCK INTERRUPTS (0 TO 59)
676 ;HIGH BYTE=ELAPSED RUN TIME IN SECONDS(0 TO 59)
677 001604 000000 $MAINT: .WORD ;CURRENT VALUE IN MAINTENANCE REGISTER
678 001606 000010 SYSSIZE: .BLKW 10 ;SYSTEM SIZE TABLE(ONE ENTRY FOR EACH DEVICE)
679 001626 000006 RUNTBL: .BLKW 6 ;RUN TIME TABLE(ONE ENTRY FOR EACH 2K BLOC<)
680 001642 000006 RUNTRAK: .BLKW 6 ;RUN TRACK TABLE(ONE ENTRY FOR EACH 2K BLOCK)
681 001656 177777 MAPTBL: .WORD -1 ;MAP TABLE(ONE BYTE FOR EACH UNIBUS DEVICE)
682 001660 177777 ;UNUSED=377, USED=LOW 5 BITS OF MAP ADDRESS
683 001662 000002 UBESAV: .BLKW 2 ;BASE ADDRESS OF UBE TRANSFER IN PROGRESS
684 001666 000002 UBEADR: .BLKW 2 ;ADDRESS THAT GETS LOADED INTO UBE BA REG
685 001672 000002 ERBA: .BLKW 2 ;18 BIT UNIBUS ADDRESS WHEN DEVICE DETECTED AN ERROR
686 .SBTTL DEVICE HANDLER STATUS WORDS
687 ;* EACH WORD HAS THE FOLLOWING BIT ASSIGNMENTS:
688 ;* 7 HANDLER READY
689 ;* 8 REPEAT LAST FUNCTION
690 ;* 15 ERROR
691 001676 000200 RP3HSTAT: .WORD 200 ;RPO3
692 001700 000200 RKHSTAT: .WORD 200 ;RK05
693 001702 000200 SPARE0: .WORD 200
694 001704 000200 SPARE1: .WORD 200
695 001706 000200 RP4HSTAT: .WORD 200 ;RPO4
696 001710 000200 RSHSTAT: .WORD 200 ;RS04
697 001712 000200 ;WORD 200 ;SPARE
698 001714 000200 ;WORD 200 ;SPARE
699
700 .SBTTL DEVICE HANDLER WORD COUNTS
701 ;* THIS TABLE GETS LOADED BY THE I/O
702 ;* RELOCATION ROUTINE WITH THE TWO'S COMPLIMENT WORD
703 ;* COUNT FOR THE TRANSFER FOR THE PARTICULAR DEVICE.
704 001716 000000 RP3HWC: .WORD ;RPO3
705 001720 000000 RKHWC: .WORD ;RK05
706 001722 000000 ;WORD ;SPARE
707 001724 000000 ;WORD ;SPARE
708 001726 000000 RP4HWC: .WORD ;RPO4
709 001730 000000 RSHWC: .WORD ;RS04
710
711 .SBTTL DEVICE HANDLER OLD BASE ADDRESS
712 ;* THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE
713 ;* WITH THE BASE ADDRESS OF THE SOURCE DATA FOR THE
714 ;* DEVICE THAT IS GOING TO TRANSFER THE DATA.
715 001732 000000 RP3OLD: .WORD ;RPO3
716 001734 000000 ;WORD
717 001736 000000 RKOLD: .WORD ;RK05
718 001740 000000 ;WORD
719 001742 000000 ;WORD ;SPARE
720 001744 000000 ;WORD ;SPARE
721 001746 000000 ;WORD
722 001750 000000 ;WORD ;SPARE
723 001752 000000 RP4OLD: .WORD ;RPO4
724 001754 000000 ;WORD
725 001756 000000 RSOLD: .WORD ;RS04
726 001760 000000 ;WORD
727
728 .SBTTL DEVICE HANDLER NEW BASE ADDRESSES
    
```

# K03

```

730      ;*      WITH THE BASE ADDRESS OF THE DESTINATION FOR THE
731      ;*      PARTICULAR DEVICE THAT IS GOING TO DO THE TRANSFER.
732      001762 000000      RP3NWL: .WORD      ;RPO3
733      001764 000000      RP3NWH: .WORD
734      001766 000000      RKNEWL: .WORD      ;RK05
735      001770 000000      RKNEWH: .WORD
736      001772 000000      .WORD      ;SPARE
737      001774 000000      .WORD
738      001776 000000      .WORD      ;SPARE
739      002000 000000      .WORD
740      002002 000000      RP4NWL: .WORD      ;RPO4
741      002004 000000      RP4NWH: .WORD
742      002006 000000      RSNEWL: .WORD      ;RSO4
743      002010 000000      RSNEWH: .WORD
744
745      .SBTTL  DEVICE HANDLER UNIT NUMBER
746      ;*      THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE.
747      ;*      IT TELLS THE DEVICE HANDLER WHICH UNIT NUMBER IS
748      ;*      TO DO THE TRANSFER.
749      002012 000000      RP3UNIT: .WORD      ;RPO3
750      002014 000000      RKUNIT:  .WORD      ;RK05
751      002016 000000      .WORD      ;SPARE
752      002020 000000      .WORD      ;SPARE
753      002022 000000      RP4UNIT: .WORD      ;RPO4
754      002024 000000      RSUNIT:  .WORD      ;RSO4
755
756      .SBTTL  ADDRESS OF THE DEVICE HANDLERS
757      ;*      THIS TABLE CONTAINS THE ADDRESS OF THE DEVICE HANDLER
758      ;*      ROUTINES. IT IS USED BY THE I/O RELOCATION ROUTINE
759      ;*      TO TRANSFER CONTROL TO THE DEVICE HANDLER.
760      002026 036772      RP3HANA: .WORD      RP3DRV  ;RPO3
761      002030 037410      RKHANA:  .WORD      RKDRV   ;RK05
762      002032 000000      .WORD      ;SPARE
763      002034 000000      .WORD      ;SPARE
764      002036 040004      RP4HANA: .WORD      RP4DRV  ;RPO4
765      002040 040354      RSHANA:  .WORD      RSDRV   ;RSO4
766
767      .SBTTL  DEVICE HANDLER DISK ADDRESS TABLE
768      ;*      THIS TABLE GETS LOADED BY THE DEVICE HANDLER WITH THE
769      ;*      DISK ADDRESS(SECTOR AND CYLINDER) OF THE CURRENT
770      ;*      TRANSFER.
771      002042 000000      RP3HDA: .WORD      ;RPO3 DISK ADDRESS
772      002044 000000      RP3HDC: .WORD      ;RPO3 DESIRED CYLINDER
773      002046 000000      RKHDA:  .WORD      ;RK05 DISK ADDRESS
774      002050 000000      .WORD      ;SPARE
775      002052 000000      RP4HDA: .WORD
776      002054 000000      RP4HDC: .WORD      ;RPO4 DESIRED CYLINDER
777      002056 000000      RSHDA:  .WORD      ;RSO4 DISK ADDRESS
778
779      .SBTTL  DEVICE HANDLER FUNCTION TABLE
780      ;*      THIS TABLE GETS LOADED BY THE DEVICE HANDLERS
781      ;*      AND THE DEVICE SERVICE ROUTINES. IT TELLS THE ROUTINES
782      ;*      WHICH FUNCTION TO DO NEXT.
783      002060 000000      RP3FUN: .WORD      ;RPO3
784      002062 000000      RKFUN:  .WORD      ;RK05
    
```

786 002066 000000  
 787 002070 000000  
 788  
 789  
 790  
 791  
 792  
 793  
 794  
 795 002072 000  
 796 002073 000  
 797 002074 000  
 798 002075 000  
 799 002076 000  
 800 002100

RP4FUN: .WORD ;RP04  
 RSFUN: .WORD ;RS04  
 .SBTTL DEVICE HANDLER RETRY COUNT  
 \* THIS TABLE GETS LOADED BY THE DEVICE HANDLERS AND IS USED  
 \* BY THE DEVICE SERVICE ROUTINES. IF AN ERROR OCCURS  
 \* THE DEVICE SERVICE ROUTINE WILL RETRY THE FUNCTION UNTIL  
 \* THE BYTE IN THIS TABLE GOES TO ZERO. IT IS INITIALIZED  
 \* TO A 3.  
 RP3TRY: .BYTE ;RP03  
 RKTRY: .BYTE ;RK05  
 .BYTE ;SPARE  
 RP4TRY: .BYTE ;RP04  
 RSTRY: .BYTE ;RS04  
 .EVEN

801  
 802  
 803  
 804  
 805  
 806  
 807  
 808  
 809  
 810 002100 176710  
 811 002102 176712  
 812 002104 176714  
 813 002106 176716  
 814 002110 176720  
 815 002112 176724  
 816 002114 176722  
 817 002116 000254  
 818 002120 000256  
 819  
 820

.SBTTL DEVICE REGISTER TABLES  
 \* THE FOLLOWING TABLES CONTAIN THE STANDARD ADDRESS FOR  
 \* THE DEVICES USED BY THIS PROGRAM. IF A DEVICE IS PLACED  
 \* AT A NON-STANDARD ADDRESS THE APPROPRIATE TABLE CAN BE  
 \* CHANGED AND THE PROGRAM WILL OPERATE THAT DEVICE.  
 \* EXCEPTION--SEE DOCUMENTATION FOR RP03 AND RP04 PROBLEMS.  
 .SBTTL RP11/RP03 REGISTERS  
 RP3DS: .WORD 176710 ;DRIVE STATUS  
 RP3ER: .WORD 176712 ;ERROR REGISTER  
 RP3CS: .WORD 176714 ;CONTROL AND STATUS  
 RP3WC: .WORD 176716 ;WORD COUNT  
 RP3BA: .WORD 176720 ;BUS ADDRESS  
 RP3DA: .WORD 176724 ;DISK ADDRESS  
 RP3DC: .WORD 176722 ;DESIRED CYLINDER  
 RP3VEC: .WORD 254 ;INTERRUPT VECTOR  
 RP3PSW: .WORD 256 ;INTERRUPT VECTOR+2

821 002122 177400  
 822 002124 177402  
 823 002126 177404  
 824 002130 177406  
 825 002132 177410  
 826 002134 177412  
 827 002136 000220  
 828 002140 000222  
 829  
 830

.SBTTL RK11/RK05 REGISTERS  
 RKDS: .WORD 177400 ;DRIVE STATUS  
 RKER: .WORD 177402 ;ERROR REGISTER  
 RKCS: .WORD 177404 ;CONTROL AND STATUS  
 RKWC: .WORD 177406 ;WORD COUNT  
 RKBA: .WORD 177410 ;BUS ADDRESS  
 RKDA: .WORD 177412 ;DISK ADDRESS  
 RKVEC: .WORD 220 ;INTERRUPT VECTOR  
 RKPSW: .WORD 222 ;INTERRUPT VECTOR+2

831 002142 176700  
 832 002144 176702  
 833 002146 176704  
 834 002150 176750  
 835 002152 176706  
 836 002154 176710  
 837 002156 176752  
 838 002160 176712  
 839 002162 176714  
 840 002164 176734

.SBTTL RH70/RP04 REGISTERS  
 RP4CS1: .WORD 176700 ;CONTROL AND STATUS #1  
 RP4WC: .WORD 176702 ;WORD COUNT  
 RP4BA: .WORD 176704 ;BUS ADDRESS  
 RP4BAE: .WORD 176750 ;BUS ADDRESS EXTENDED  
 RP4DA: .WORD 176706 ;DISK ADDRESS  
 RP4CS2: .WORD 176710 ;CONTROL AND STATUS #2  
 RP4CS3: .WORD 176752 ;CONTROL AND STATUS #3  
 RP4DS: .WORD 176712 ;DRIVE STATUS  
 RP4ER1: .WORD 176714 ;ERROR REG #1  
 RP4DC: .WORD 176734 ;DESIRED CYLINDER

842 002170 176742  
 843 002172 176736  
 844 002174 176732  
 845 002176 000254  
 846 002200 000256

RP4ER3: .WORD 176742 :ERROR REG #3  
 RPCC: .WORD 176736 :CURRENT CYLINDER  
 RP4OF: .WORD 176732 :OFFSET REGISTER  
 RP4VEC: .WORD 254 :INTERRUPT VECTOR  
 RP4PSW: .WORD 256 :INTERRUPT VECTOR+2

848  
 849 002202 172040  
 850 002204 172042  
 851 002206 172044  
 852 002210 172070  
 853 002212 172046  
 854 002214 172050  
 855 002216 172072  
 856 002220 172052  
 857 002222 172054  
 858 002224 000204  
 859 002226 000206

.SBTTL RH70/R504 REGISTERS  
 RSCS1: .WORD 172040 :CONTROL AND STATUS #1  
 RSWC: .WORD 172042 :WORD COUNT  
 RSBA: .WORD 172044 :BUS ADDRESS  
 RSBAE: .WORD 172070 :BUS ADDRESS EXTENDED  
 RSDA: .WORD 172046 :DISK ADDRESS  
 RSCS2: .WORD 172050 :CONTROL AND STATUS #2  
 RSCS3: .WORD 172072 :CONTROL AND STATUS #3  
 RSDS: .WORD 172052 :DRIVE STATUS  
 RSER: .WORD 172054 :ERROR REG  
 RSVEC: .WORD 204 :INTERRUPT VECTOR  
 RSPSW: .WORD 206 :INTERRUPT VECTOR+2

860  
 861  
 862  
 863  
 864  
 865  
 866 002230 170002  
 867 002232 170004  
 868 002234 170016  
 869 002236 170006  
 870 002240 170010  
 871 002242 000510  
 872 002244 000512

.SBTTL UNIBUS EXERCISER REGISTER ADDRESS TABLE  
 THIS TABLE IS ASSEMBLED FOR UBE #0. IF THE UBE  
 ADDRESSES ARE CUT FOR OTHER THAN UNIT #0, THE PROGRAM  
 WILL CHANGE THIS TABLE. THE PROGRAM LOOKS FOR A  
 UBE AT ADDRESSES 770002, 770022, 770032, AND 770042.  
 UBETBL: .WORD UBEC :CYCLE COUNT  
 .WORD UBEBA :BUS ADDRESS REG  
 .WORD UBECR2 :CONTROL REGISTER #2  
 .WORD UBECR1 :CONTROL REGISTER #1  
 .WORD UBECLE :UBE CLEAR ADDRESS  
 .WORD UBEVEC :INTERRUPT VECTOR  
 .WORD UBEVEC+2 :INTERRUPT VECTOR +2

873  
 874  
 875  
 876  
 877  
 878  
 879 002246 160100  
 880 002250 160102  
 881 002252 160104  
 882 002254 160174  
 883 002256 160106  
 884 002260 160110  
 885 002262 160112  
 886 002264 160114  
 887 002266 160176  
 888 002270 000774  
 889 002272 000776  
 890 002274 160126  
 891 002276 160200  
 892 002300 160300  
 893 002302 160400

.SBTTL MASS BUS TESTER REGISTER ADDRESSES  
 THE PROGRAM IS ASSEMBLED WITH ADDRESSES FOR A MBT  
 AT 770100. IF THE MBT IS AT ANOTHER ADDRESS THE PROGRAM  
 WILL CHANGE THIS TABLE. THE PROGRAM LOOKS FOR A UBE  
 AT ADDRESSES 770100, 770200, 770300, AND 770400.  
 MBTTBL: .WORD MBTCS1 :CONTROL AND STATUS #1  
 .WORD MBTWC :WORD COUNT  
 .WORD MBTBA :BUS ADDRESS  
 .WORD MBTBAE :BUS ADDRESS EXTENDED  
 .WORD MBTMR2 :MAINTENANCE REGISTER #2  
 .WORD MBTCS2 :CONTROL REGISTER #2  
 .WORD MBTST :STATUS REGISTER  
 .WORD MBTER :ERROR REGISTER  
 .WORD MBTCS3 :CONTROL REGISTER #3  
 .WORD MBTVEC :INTERRUPT VECTOR  
 .WORD MBTVECP :INTERRUPT VECTOR+2  
 .WORD MBTDT :DRIVE TYPE REGISTER  
 MBTN2: .WORD 160200 :MASS BUS TESTER #2  
 MBTN3: .WORD 160300 :MASS BUS TESTER #3  
 MBTN4: .WORD 160400 :MASS BUS TESTER #4



895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949

002304

002304 054774  
002306 055021  
002310 055066  
002312 055060  
002314 055100  
002316 055126  
002320 055154  
002322 055060  
002324 055164  
002326 055221  
002330 055270  
002332 055060  
002334 055304  
002336 055333  
002340 055270  
002342 055411  
002344 055416  
002346 055455  
002350 055526  
002352 055522  
002354 055540  
002356 055610  
002360 055630  
002362 055411  
002364 000000  
002366 000000  
002370 000000  
002372 000000  
002374 055636  
002376 055707

\*\*\*\*\*

.SBTTL ERROR POINTER TABLE

\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
\*LOCATION \$ITEMB, THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
\*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

\* EM :POINTS TO THE ERROR MESSAGE  
\* DH :POINTS TO THE DATA HEADER  
\* DT :POINTS TO THE DATA  
\* DF :POINTS TO THE DATA FORMAT

\$ERRTB:

: ITEM 1  
EM1 :UNEXPECTED TRAP TO 4  
DH1 :PCOFTP PHYSPC PSW CPUERR  
DT1 :VADR,VADR,\$TMPO,\$TMP2  
DF1 :0,1,0,0,0  
: ITEM 2  
EM2 :UNEXPECTED TRAP TO 10  
DH2 :PCOFTP PHYSPC PSW  
DT2 :VADR,VADR,\$TMPO  
DF1 :  
: ITEM 3  
EM3 :UNEXPECTED TRAP TO 250(MGMT)  
DH3 :PCOFTP PHYSPC PSW MMR0 MMR2  
DT3 :VADR,VADR,\$TMPO,,\$TMP2,,\$TMP3  
DF1 :  
: ITEM 4  
EM4 :UNEXPECTED TRAP TO 114  
DH4 :PCOFTP PHYSPC PSW ERADREG MEMERRREG  
DT3 :VADR,VADR,\$TMPO,\$TMP3,\$TMP2  
DF4 :0,1,0,2,0  
: ITEM 5  
EM5 :PARITY ERROR DURING DATA CHECK  
DH5 :SRCADR DSTADR ERRADREG MEM ERR REG  
DT5 :\$TMPO,PA1500,\$TMP3,\$TMP2  
DF5 :  
: ITEM 6  
EM6 :ERROR DURING CHECK OF RELOCATED DATA  
DH6 :SRCADR DSTADR  
DT6 :\$TMPO,PA1500  
DF4 :  
: ITEM 7  
0  
0  
0  
0  
: ITEM 10  
EM10 :ERROR DURING DATA CHECK-RELOC WAS BY I/O  
DH10 :SRCADR DSTADR DEVICE THAT DID XFER

951	002402	055756	DF10	:0,1,3,0
952			:ITEM 11	
953	002404	055774	EM11	:BIT(S) STUCK IN MICRO-BREAK REG
954	002406	056041	DH11	:GOOD DAT BAD DAT
955	002410	056064	DT11	:\$TMP0,\$TMP1
956	002412	056062	DF11	:0,0
957			:ITEM 12	
958	002414	056072	EM12	:UNIBUS EXERCISOR NON-EXISTANT MEMOREY
959	002416	056130	DH12	:PHYSICAL ADDRESS
960	002420	056146	DT12	:PA1500
961	002422	056144	DF12	:2
962			:ITEM 13	
963	002424	056152	EM13	:MASS BUS TESTER NON-EXISTANT MEMORY
964	002426	056210	DH13	:PHYSICAL ADDRESS
965	002430	056146	DT12	
966	002432	056144	DF12	
967			:ITEM 14	
968	002434	056225	EM14	:FLOATING POINT ERROR
969	002436	056252	DH14	:DATA1 DATA2
970	002440	056272	DT14	:\$TMP4,\$REG2,\$TMP6,\$REG3
971	002442	056304	DF14	:4,0,4,0
972			:ITEM 15	
973	002444	056310	EM15	:DEVICE HUNG
974	002446	000000	0	
975	002450	000000	0	
976	002452	000000	0	
977			:ITEM 16	
978	002454	056225	EM14	:FLOATING POINT ERROR
979	002456	056324	DH16	
980	002460	056356	DT16	:FLTMP0,\$REG2,FLTMP1,\$REG3

```

982 002464 013737 177570 177570
983 002472 000774
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001 002474 012706 001100
1002 002500 012737 051374 000034
1003 002506 012737 044774 000030
1004 002514 012700 000377
1005 002520 010037 177770
1006 002524 020037 177770
1007 002530 001024
1008 002532 005000
1009 002534 010037 177770
1010 002540 020037 177770
1011 002544 001016
1012 002546 012700 000125
1013 002552 010037 177770
1014 002556 020037 177770
1015 002562 001007
1016 002564 012700 000252
1017 002570 010037 177770
1018 002574 020037 177770
1019 002600 001411
1020 002602 010067 176470
1021 002606 013737 177770 001300
1022 002614 012737 002474 001212
1023 002622 104011
1024
1025 002624 012737 000100 177770
1026 002632 104400 002640
1027 002636 000421
1028
1029 002702
1030 002702 104400 002710
1031 002706 000407
1032
1033 002726
1034 002726 000000
1035 002730 012737 000012 000010
1036 002736 012737 000002 000012

```

```

START1: MOV    J#SWR,J#SWR
BR      START1
.SBTTL  PROGRAM INITIALIZATION

:*****
:SBTTL  MICRO-BREAK REGISTER TEST
:THIS TEST IS EXECUTED BY STARTING THE PROGRAM AT ADDRESS 214.
:THIS TEST REQUIRES A MAINTENANCE CARD AND OPERATOR INTERVENTION.
:THE PROCESSOR SHOULD STOP 8 TIMES. FOLLOWING IS THE DATA
:THAT SHOULD BE IN THE MICRO-ADRESS DATA LIGHTS EACH TIME:
:
: 1      000
: 2      001
: 3      002
: 4      004
: 5      010
: 6      020
: 7      040
: 8      200
:*****
START2: MOV    #100,SP ;SETUP THE SP
MOV    #TRAP,J#TRAPVEC ;SETUP TRAP VECTOR
MOV    #ERROR,J#EMTVEC ;SETUP EMT VECTOR
MOV    #377,R0 ;PUT MICRO-BREAK DATA IN R0
MOV    R0,J#UBREAK ;LOAD U BREAK REG
CMP    R0,J#UBREAK ;LOAD OK?
BNE    UBRERR ;BRANCH IF NO
CLR    R0
MOV    R0,J#UBREAK
CMP    R0,J#UBREAK
BNE    UBRERR
MOV    #125,R0
MOV    R0,J#UBREAK
CMP    R0,J#UBREAK
BNE    UBRERR
MOV    #252,R0
MOV    R0,J#UBREAK
CMP    R0,J#UBREAK
BEQ    UBRK2
UBRERR: MOV    R0,$TMP0
MOV    J#UBREAK,J#STMP1
MOV    #START2,J#SLPERR
ERROR  11
:TEST TO ENSURE U BREAK COMPARATORS DO NOT COME ON.
UBRK2: MOV    #100,J#UBREAK ;PUT SAFE VALUE IN REG
TYPE   65$ ;:TYPE ASCIZ STRING
BR     64$ ;:GET OVER THE ASCIZ
:65$: .ASCIZ /SET MAINT TO STOP ON MICRO-BREAK/<<CRLF>
64$:
:67$: .ASCIZ /HIT CONTINUE/<<CRLF>
66$:
:67$: .ASCIZ /HIT CONTINUE/<<CRLF>
66$:
HALT
MOV    #12,J#RESVEC
MOV    #2,J#RESVEC+2

```

```

1038 002750 012701 000010          MOV      #10,R1          ;SET SOB COUNT
1039 002754 012702 003161          MOV      @UBRTBL+1,R2   ;GET ADRS OF UBREAK DATA TABLE
1040 002760 112237 177770          1$:     MOVVB   (R2)+,@#UBREAK ;LOAD MICRO-BREAK FROM TABLE
1041 002764 000010          10      ;EXEC RES INSTR (ROM ADRS 000)
1042 002766 005037 177770          CLR      @#UBREAK
1043 002772 077105          SOB      R1,1$         ;CONTINUE
1044 002774 012737 000125 177770    MOV      #125,@#UBREAK ;SET MICRO-BREAK DATA PATTERN
1045 003002 006400          MARK    0              ;EXEC MARK (ROM ADRS 252)
1046 003004 005037 177770          2$:     CLR      @#UBREAK
1047 003010 012706 001100          MOV      #1100,SP      ;RESTORE SP
1048 003014 012737 000006 000004    MOV      #6,@#ERRVEC
1049 003022 012737 000002 000006    MOV      #2,@#ERRVEC+2
1050 003030 052737 040000 177776    BIS      #1114,@#PSW   ;GO TO SUPER MODE
1051 003036 012706 000700          MOV      #700,SP      ;SET SUPER SP
1052 003042 012746 003064          MOV      #3$,-(SP)    ;SETUP STACK FOR JSR INSTR
1053 003046 005000          CLR      R0           ;SETUP R0
1054 003050 012701 000007          MOV      #7,R1        ;SET SOB COUNT
1055 003054 012702 003174          MOV      @INSTBL+2,R2 ;GET ADRS OF TABLE OF INSTRUCTIONS
1056 003060 012217          4$:     MOV      (R2)+,(PC)  ;GET INSTRUCTION
1057 003062 000000          .WORD   ;EXECUTE INSTRUCTION
1058 003064 077103          3$:     SOB      R1,4$    ;CONTINUE
1059 003066 012737 000100 177770    MOV      #100,@#UBREAK ;PUT SAFE VALUE IN UBREAK REG
1060 003074 005000          CLR      R0
1061 003076 012702 003160          MOV      @UBRTBL,R2
1062 003102 012703 003172          MOV      @INSTBL,R3
1063 003106 012701 000010          MOV      #10,R1
1064 003112 012746 003126          MOV      #5$,-(SP)
1065 003116 112237 177770          6$:     MOVVB   (R2)+,@#UBREAK ;LOAD UBREAK REG FROM TABLE
1066 003122 012317          MOV      (R3)+,(PC)  ;GET INSTR FROM TABLE
1067 003124 000000          .WORD   ;EXECUTE INSTR. PROCESSOR SHOULD STOP
1068          ;WITH THE CORRECT ROM ADR IN THE LIGHTS
1069 003126 077105          5$:     SOB      R1,6$    ;CONTINUE
1070 003130 111237 177770          MOVVB   (R2),@#UBREAK ;PUT SAFE VALUE IN UBREAK REG
1071 003134 005037 177776          CLR      @#PSW       ;GO BACK TO KERNEL MODE
1072 003140 104400 003146          TYPE    69$          ;TYPE ASCIZ STRING
1073 003144 000403          BR      68$          ;GET OVER THE ASCIZ
1074          ;69$: .ASCIZ /DONE/<CRLF>
1075          68$:
1076 003154 000000          HALT
1077 003156 000415          BR      START
1078 003160 000 001 002  UBRTBL: .BYTE 0,1,2,4,10,20,40,200,100
1079 003163 004 010 020
1080 003166 040 200 100
1081 003172
1082 003172 000010 005010 005020  INSTBL: .EVEN
1083 003200 005040 000000 005200  .WORD 10,5010,5020,5040,0,5200,207,5010

```

```

1085 003212 012706 001200 START: MOV #KERSTK, SP ;SET KERNEL STACK PTR
1086 003216 012737 076543 001524 MOV #76543, #SHINUM ;INITIALIZE RANDOM NUM GEN
1087 003224 012737 123456 001522 MOV #123456, #SLONUM
1088
1089 ; DETERMINE HOW PROGRAM WAS LOADED AND WHAT MODE (IF ACT11)
1090 ; AND SET MEMORY PROTECTION.
1091 003232 005037 001504 CLR #QV ;SET NOT QV NOR AA MODE
1092 003236 005027 CLR (PC)+ ;SET NOT XXDP
1093 003240 000 XXDP: .BYTE 0 ;XXDP INDICATOR
1094 003241 000 XXDPC: .BYTE 0 ;XXDP CHAIN MODE INDICATOR
1095 003242 005027 CLR (PC)+ ;CLEAR MEMORY PROTECTION LIMIT
1096 003244 000000 PROT: .WORD 0 ;WILL CONTAIN MEM PROT LIMIT
1097 003246 005737 036756 TST #SENDAD+4 ;BRANCH IF NOT QV
1098 003252 100003 BPL 1$
1099 003254 110637 001504 MOVB SP, #QV ;SET ACT11 QV MODE
1100 003260 000411 BR 3$
1101
1102 003262 001003 1$: BNE 2$
1103 003264 110637 001505 MOVB SP, #AA ;SET ACT11 AA MODE
1104 003270 000405 BR 3$
1105
1106 003272 005737 000042 2$: TST #42 ;BRANCH IF NOT IN CHAIN MODE
1107 003276 001402 BEQ 3$
1108 003300 110637 003241 MOVB SP, #XXDPC ;SET CHAIN MODE INDICATOR
1109
1110 ;SET MEMORY PROTECTION LIMITS
1111 003304 005737 001504 3$: TST #QV ;BRANCH IF QV OR AA
1112 003310 001006 BNE MEMSIZ
1113 003312 005737 003240 TST #XXDP ;BRANCH IF NOT VIA XXDP
1114 003316 001403 BEQ MEMSIZ
1115 003320 012737 005700 003244 MOV #5700, #PROT ;PROTECT XXDP MONITOR
1116 003326 012737 157776 001516 MEMSIZ: MOV #157776, #LSTMEM ;SET VALUE INTO LSTMEM
1117 003334 163737 003244 001516 SUB #PROT, #LSTMEM ;SET PROTECTION
1118 003342 012737 056372 001514 MOV #ENDTAG+2, #FRSTMEM ;SET FIRST RELOCATION ADDRESS
1119
1120 ;GET ADDRESS OF THE LAST MEMORY LOCATION ON THE SYSTEM
1121 003350 005002 CLR R2
1122 003352 013703 177760 MOV #SIZELO, R3 ;GET ADDRESS OF SIZE REG LO
1123 003356 073227 000006 ASHC #6, R2 ;SHIFT TO FORM ADDRESS
1124 003362 052703 000077 BIS #7, R3 ;ENSURE LOWER SIX BITS SET
1125 003366 062703 000001 ADD #1, R3
1126 003372 005502 ROC R2
1127 003374 010237 001560 MOV R2, #MXMMHI ;SAVE UPPER SIX BITS
1128 003400 010337 001562 MOV R3, #MXMMLO ;SAVE LOWER 16 BITS
1129
1130 003404 012706 001200 MOV #KERSTK, SP ;SET STACK PTR
1131 003410 005037 001200 CLR #SPASS ;CLEAR PASS COUNT
1132 003414 105037 001503 CLRB #MMON ;SET MEM MGMT ON IND=NOT ON
1133 003420 012737 000600 001520 MOV #600, #NEXPAR ;SET FIRST 'PAR' VALUE
1134 003426 005737 003244 TST #PROT
1135 003432 001403 BEQ 1$
1136 003434 012737 001600 001520 MOV #1600, #NEXPAR
1137 003442 1$: MOV #27, R0 ;SET SOB COUNT
1138 003442 012700 000027 CLR R1 ;SETUP INDEX
1139 003446 005001

```

```

1141 003454 062701 000002          ADD    #2,R1
1142 003460 077005          SOB    R0,2$          ;CONTINUE
1143 003462 012737 177777 001656      MOV    #-1,2$MAPTBL  ;INITIALIZE MAP TABLE
1144 003470 012737 177777 001660      MOV    #-1,2$MAPTBL+2
1145 003476 012700 000010          MOV    #10,R0        ;SET SOB COUNT
1146 003502 012701 001676          MOV    #RPNHSTAT,R1  ;GET ADDRESS OF HANDLER STAT
1147 003506 012721 000200          MOV    #200,(R1)+    ;INITIALIZE STATUS TABLE
1148 003512 077003          SOB    R0,3$          ;CONTINUE
1149 003514 012737 000060 001532      MOV    #60,2$SUBPASS ;INIT SUBPASS TO ASCII 0
1150 003522 012700 047106          MOV    #TIMEBUF,R0   ;GET ADR OF TIME BUFFER
1151 003526 012701 000012          MOV    #12,R1        ;SET SOB COUNT
1152 003532 112720 000060          MOV    #60,(R0)+     ;INIT TIME BUFFER
1153 003536 077103          SOB    R1,4$
1154 003540 105040          CLRB   -(R0)         ;INSERT TERMINATOR
1155 003542 112737 000072 047111      MOV    #72,2$TIMEBUF+3 ;INSERT COLON
1156 003550 112737 000072 047114      MOV    #72,2$TIMEBUF+6
1157 003556 012737 000340 177776      MOV    #340,2$PS     ;LOCK OUT ALL INTERRUPTS
1158 003564 012706 001200          MOV    #SCMTAG,R6    ;FIRST LOCATION TO BE CLEARED
1159 003570 005026          CLR    (R6)+         ;CLEAR MEMORY LOCATION
1160 003572 022706 001240          CMP    #STKS,R6     ;DONE?
1161 003576 001374          BNE    -6           ;LOOP BACK IF NO
1162 003600 012706 001200          MOV    #STACK,SP    ;SETUP THE STACK POINTER
1163 003604 012737 044534 000020      MOV    #SSCOPE,2$IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
1164 003612 012737 000340 000022      MOV    #340,2$IOTVEC+2 ;LEVEL 7
1165 003620 012737 044774 000030      MOV    #SERIOR,2$EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
1166 003626 012737 000340 000032      MOV    #340,2$EMTVEC+2 ;LEVEL 7
1167 003634 012737 051374 000034      MOV    #STRAP,2$TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
1168 003642 012737 000340 000036      MOV    #340,2$TRAPVEC+2 ;LEVEL 7
1169 003650 012737 051230 000024      MOV    #SPWRON,2$PWAVEC ;POWER FAILURE VECTOR
1170 003656 012737 000340 000026      MOV    #340,2$PWAVEC+2 ;LEVEL 7
1171 003664 016767 032726 032716      MOV    #ENDCT,SEOPCT ;SETUP END-OF-PROGRAM COUNTER
1172 003672 005067 175420          CLR    #TIMES       ;INITIALIZE NUMBER OF ITERATIONS
1173 003676 005067 175416          CLR    #ESCAPE      ;CLEAR THE ESCAPE ON ERROR ADDRESS
1174 003702 112767 000001 175307      MOV    #1,SERMAX    ;ALLOW ONE ERROR PER TEST
1175 003710 012767 003710 175272      MOV    #.,$LPADR    ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1176 003716 012767 003716 175266      MOV    #.,$LPERA    ;SETUP THE ERROR LOOP ADDRESS
1177
1178 ;CLEAR PROGRAM INDICATORS
1179 003724 052777 000100 175306      BIS    #100,2$TKS   ;SET IE BIT IN KEYBOARD STATUS REG
1180 003732 012737 052400 000060      MOV    #TKISR,2$TKVEC ;SETUP KEYBOARD VECTOR
1181 003740 012737 000200 000062      MOV    #PR4,2$TKVEC+2
1182 003746 012737 052612 000064      MOV    #TPISR,2$TPVEC
1183 003754 012737 000200 000066      MOV    #PR4,2$TPVEC+2
1184 003762 005037 001466          CLR    #NOTYPE     ;CLEAR 'NO TYPING' INDICATOR
1185
1186 ;THE BELOW ROUTINE ASCERTAINS WHICH CP & CP OPTIONS THE PROGRAM IS RUN-
1187 ;NING ON AND SETS AN INDICATOR IN OPT.CP ACCORDINGLY.
1188 003766 012737 000006 000004      CPCHK: MOV    #ERRVEC+2,2$ERRVEC ;SET UP ERROR TRAP TO RETURN
1189 003774 012737 000002 000006      MOV    #2,2$ERRVEC+2
1190 004002 012737 000012 000010      MOV    #RESVEC+2,2$RESVEC ;AND ALSO RESERVED INST TRAP
1191 004010 012737 000002 000012      MOV    #2,2$RESVEC+2
1192 004016 012702 144006          MOV    #144006,R2   ;SET 11/70 NON-OPTION BITS
1193 004022 000261          SEC
1194 004024 170500          TSTF  R0            ;WILL CLEAR CARRY IF 11/45 FLOATING POINT
1195 004026 170000          CFCC              ;IS AVAIL. COPY FLOATING CC'S INTO PSW

```

1197	004032	052702	020000		BIS	#FPOPT,R2	;SET FP OPTION AVAIL INDICATOR
1198	004036	000261		6\$:	SEC		
1199	004040	005737	177546		TST	#LKS	;BRANCH IF NO KW11-L
1200	004044	103402			BCS	7\$	
1201	004046	052702	001000		BIS	#LKOPT,R2	;SET OPTION INDICATOR
1202	004052	000261		7\$:	SEC		
1203	004054	005777	175164		TST	#STPS	;BRANCH IF NO CONSOLE TTY
1204	004060	103402			BCS	9\$	
1205	004062	052702	000400		BIS	#TTOPT,R2	
1206	004066	005003		9\$:	CLR	R3	
1207	004070	000261			SEC		
1208	004072	005737	170000		TST	#UBED8	; IS UBE1 THERE?
1209	004076	103410			BCS	12\$	; BRANCH IF NO
1210	004100	105037	170006		CLRB	#UBECR1	; IS THIS A TESTER OR EXERCISOR?
1211	004104	105737	170006		TSTB	#UBECR1	
1212	004110	100045			BPL	15\$	; BRANCH IF TESTER
1213	004112	052702	000200	16\$:	BIS	#UBEOPT,R2	; SET INDICATOR
1214	004116	000425			BR	17\$	
1215	004120	000261		12\$:	SEC		
1216	004122	005737	170020		TST	#UBED8+20	; IS UBE2 THERE?
1217	004126	103403			BCS	13\$	; BRANCH IF NO
1218	004130	012703	000020		MOV	#20,R3	; SET OFFSET IN R3
1219	004134	000766			BR	16\$	
1220	004136	000261		13\$:	SEC		
1221	004140	005737	170040		TST	#UBED8+40	; IS UBE3 THERE?
1222	004144	103403			BCS	14\$	; BRANCH IF NO
1223	004146	012703	000040		MOV	#40,R3	; PUT OFFSET IN R3
1224	004152	000757			BR	16\$	
1225	004154	000261		14\$:	SEC		
1226	004156	005737	170060		TST	#UBED8+60	; IS UBE4 THERE?
1227	004162	103420			BCS	15\$	; BRANCH IF NO
1228	004164	012703	000060		MOV	#60,R3	; PUT OFFSET IN R3
1229	004170	000750			BR	16\$	
1230	004172	005227	177777	17\$:	INC	#-1	
1231	004176	001012			BNE	15\$	
1232	004200	012704	002230		MOV	#UBETBL,R4	; GET ADDRESS OF UBE TABLE
1233	004204	012705	000005		MOV	#5,R5	; SET SOB COUNT
1234	004210	060324		18\$:	ADD	R3,(R4)+	; ADJUST UBE TABLE ENTRIES
1235	004212	077502			SOB	R5,18\$	; CONTINUE
1236	004214	006003			ROR	R3	
1237	004216	006003			ROR	R3	; ADJUST OFFSET FOR UBE VECTOR
1238	004220	060324			HOD	R3,(R4)+	; ADJUST UBEVEC ENTRY
1239	004222	060314			ADD	R3,(R4)	; ADJUST UBEVEC PSW ENTRY
1240	004224	005003		15\$:	CLR	R3	; INIT R3
1241	004226	000261			SEC		
1242	004230	005777	176012		TST	#MBTTBL	; IS MASS BUS TESTER THERE?
1243	004234	103403			BCS	20\$	; BRANCH IF NO
1244	004236	052702	002000	21\$:	BIS	#MBTOPT,R2	; SET OPTION AVAILABLE
1245	004242	000422			BR	24\$	
1246	004244	005777	176026	20\$:	TST	#MBTN2	; IS MBT2 THERE?
1247	004250	103403			BCS	22\$	; BRANCH IF NO
1248	004252	012703	000100		MOV	#100,R3	; SETUP R3
1249	004256	000767			BR	21\$	
1250	004260	005777	176014	22\$:	TST	#MBTN3	; IS MBT3 THERE?
1251	004264	103403			BCS	23\$	; BRANCH IF NO

```

1253 004272 000761          BR      21$
1254 004274 005777 176002 23$:  TST    2MBTN4          ; IS MBT4 THERE?
1255 004300 103427          BCS    30$              ; BRANCH IF NO
1256 004302 012703 000300          MOV    200,R3
1257 004306 000753          BR      21$
1258 004310 005227 177777 24$:  INC    2-1
1259 004314 001021          BNE    30$
1260 004316 012704 002246          MOV    2MBTTBL,R4      ; GET ADDRESS OF MBT TABLE
1261 004322 012705 000011          MOV    211,R5          ; SET SOB COUNT
1262 004326 060324          ADD    R3,(R4)+        ; ADJUST MBT TABLE
1263 004330 077502          SOB    R5,25$          ; CONTINUE
1264 004332 060337 002274          ADD    R3,2MBTTBL+26  ; ADJUST DRIVE TYPE ADDRESS
1265 004336 112777 000007 175714          MOVB  27,2MBTTBL+12   ; SET UNIT NUMBER
1266 004344 122777 009040 175722          CMPB  240,2MBTTBL+26  ; IS THIS REALLY A MBT?
1267 004352 001402          BEQ    30$              ; BRANCH IF YES
1268 004354 042702 002000          BIC    2MBTOPT,R2      ; CLEAR OPTION AVAILABLE BIT
1269 004360 012737 053442 000004 30$:  MOV    2ERRPT,2ERRVEC ; RESTORE ERROR TRAP
1270 004366 012737 053370 000010          MOV    2RESERR,2RESVEC ; AND ALSO RESERVED INST TRAP
1271 004374 010237 001470          MOV    R2,2OPT.CP     ; LOAD INDICATOR
1272 004400 005227 177777          INC    2-1
1273 004404 001031          BNE    64$              ; FIRST TIME?
1274 004406 022737 036752 000042          CMP    2SENDAD,242    ; ACT-11?
1275 004414 001425          BEQ    64$              ; BRANCH IF YES
1276 004416 104400 004424          TYPE  65$              ; TYPE ASCIZ STRING
1277 004422 000422          BR      64$              ; GET OVER THE ASCIZ
1278          ;:65$: .ASCIZ <CRLF>"CEQKCC...POP 11/70 CPU EXERCISOR"<CRLF>
1279 004470          64$:
1280          ;*****
1281          ;SBTTL SYSTEM SIZER
1282          ;THIS ROUTINE DETERMINES WHAT DRIVES ARE AVAILABLE ON
1283          ;THE FOLLOWING DEVICES: RK05, RPO3, RPO4, AND RSO4. THE
1284          ;INFORMATION IS STORED IN THE TABLE "SYSSIZE" IN THE FOLLOWING FORMAT:
1285          ;A. EACH DEVICE IS ASSIGNED A WORD
1286          ;B. THE LOW BYTE OF THIS WORD INDICATES WHICH DRIVES ARE AVAILABLE
1287          ;C. THE HIGH BYTE INDICATES WHICH DRIVES HAVE BEEN USED
1288          ;BY THE RELOCATION ROUTINE.
1289          ;*****
1290 004470 012737 004602 000004 SIZE: MOV    221,2ERRVEC      ; SETUP TIMEOUT VECTOR
1291 004476 005037 001276          CLR    2$TMP0          ; ENSURE $TMP0 CLEAR
1292 004502 005000          CLR    R0              ; USED TO SET THE UNIT AVAIL BITS
1293 004504 012701 000010          MOV    210,R1          ; SOB COUNT
1294 004510 013777 001276 175416 9$:  MOV    2$TMP0,2RKDA    ; SET UNIT NUMBER
1295 004516 012777 000015 175402          MOV    215,2RKCS      ; SEND DRIVE RESET
1296 004524 032777 000200 175372          BIT    2BIT7,2RKER    ; NOW EXISTANT DISK?
1297 004532 001011          BNE    7$              ; BRANCH IF YES
1298 004534 017702 175362          MOV    2RKDS,R2        ; GET DRIVE STATUS
1299 004540 042702 177537          BIC    2177537,R2     ; GET BITS 5 & 7 ONLY
1300 004544 022702 000200          CMP    2200,R2         ; IS DRIVE READY?
1301 004550 001002          BNE    7$              ; BRANCH IF NO
1302 004552 052700 000400          BIS    2BIT8,R0        ; SET UNIT AVAILABLE
1303 004556 006000          ROR    R0
1304 004560 012777 000001 175340          MOV    21,2RKCS       ; CLEAR THE ERRORS
1305 004566 062737 020000 001276          ADD    220000,2$TMP0  ; SELECT NEXT UNIT
1306 004574 077133          SOB    R1,9$          ; CONTINUE
1307 004576 110037 001610          MOVB  R0,2SYSSIZE+2  ; STORE IN TABLE

```



```

1309          ;*****
1310          ;THIS CODE DETERMINES IF THERE IS AN RPO3 OR AN RPO4 OR BOTH.
1311          ;IF BOTH ARE ON THE SYSTEM, THE OPERATOR MUST CHANGE THE RPO4
1312          ;ADDRESSES IN THE TABLE IN "COMMON TAGS" AND "NOP" THE BRANCH
1313          ;AT "100$".
1314
1315 004602 012737 005042 000004 21$:  MOV    #11$,@ERRVEC      ;SET THE ERROR VECTOR
1316 004610 005737 176710          TST    @#176710      ;IS THERE AN RP ON THE SYSTEM?
1317          ;STAY HERE IF YES
1318 004614 012737 004630 000004          MOV    #1$,@ERRVEC
1319 004622 005777 175314          TST    @RP4CS1      ;IS THERE AN RPO4 ON SYSTEM?
1320 004626 000441          BR     100$        ;BRANCH IF YES
1321          ;*****
1322
1323          ;*****
1324 004630 012737 004732 000004 1$:  MOV    #10$,@ERRVEC  ;SETUP TIMEOUT VEC FOR RPO3 TEST
1325 004636 012737 000001 001276          MOV    #1,@STMP0    ;SETUP TEMPO
1326 004644 005000          CLR    R0           ;USED TO SET UNIT AVAILABLE BITS
1327 004646 012701 000010          MOV    #10,R1       ;SOB COUNT
1328 004652 013777 001276 175224 3$:  MOV    @STMP0,@RP3CS ;SET FUNCTION IDLE WITH UNIT NO
1329 004660 005777 175220          TST    @RP3CS       ;WAS THERE AN ERROR?
1330 004664 100006          BPL   6$           ;BRANCH IF NO
1331 004666 006000          ROR    R0           ;UNIT NOT AVAILABLE
1332 004670 062737 000400 001276          ADD    #400,@STMP0  ;SELECT NEXT UNIT
1333 004676 077113          SOB   R1,3$        ;CONTINUE
1334 004700 000412          BR     5$
1335 004702 017702 175172          MOV    @RP305,R2    ;GET STATUS REGISTER
1336 004706 042702 136377          BIC   #136377,R2   ;GET BITS 14 9 & 8 ONLY
1337 004712 022702 041400          CMP   #41400,R2    ;IS DRIVE READY?
1338 004716 001363          BNE   4$           ;BRANCH IF NO
1339 004720 052700 000400          BIS   #BIT8,R0     ;SET DRIVE AVAILABLE BIT
1340 004724 000760          BR     4$           ;CONTINUE
1341 004726 110037 001606          MOVB  R0,@SYSSIZE  ;STORE IN TABLE
1342
1343          ;*****
1344 004732 012737 005042 000004 10$: MOV    #11$,@ERRVEC  ;SETUP ERROR VEC FOR RPO4 TEST
1345 004740 005037 001276          CLR   @STMP0
1346 004744 005000          CLR   R0           ;UNIT AVAILABLE WORD
1347 004746 012701 000010          MOV   #10,R1       ;SOB COUNT
1348 004752 113777 001276 175174 14$: MOVB  @STMP0,@RP4CS2 ;SET UNIT NUMBER
1349 004760 012777 000021 175154          MOV   #21,@RP4CS1 ;TRY READ-IN-PRESET
1350 004766 032777 010000 175160          BIT   #BIT12,@RP4CS2 ;NON EXISTANT DRIVE?
1351 004774 001011          BNE   12$          ;BRANCH IF YES
1352 004776 017702 175156          MOV   @RP405,R2    ;GET DRIVE STATUS
1353 005002 042702 163277          BIC   #163277,R2   ;GET BITS 12, 11, 8, & 6 ONLY
1354 005006 022702 010500          CMP   #10500,R2   ;IS DRIVE READY?
1355 005012 001002          BNE   12$          ;BRANCH IF NO
1356 005014 052700 000400          BIS   #BIT8,R0     ;SET UNIT AVAILABLE
1357 005020 006000          ROR   R0
1358 005022 052777 000040 175124 12$: BIS   #BITS,@RP4CS2 ;CLEAR ERROR BITS
1359 005030 005237 001276          INC   @STMP0       ;SELECT NEXT DRIVE
1360 005034 077132          SOB   R1,14$      ;CONTINUE
1361 005036 110037 001616          MOVB  R0,@SYSSIZE+10 ;STORE IN TABLE
1362
1363          ;*****

```

```

1365 005050 005037 001276          CLR      @#STMP0
1366 005054 005030          CLR      RO
1367 005056 012701 000010          MOV      @10,R1          ;SOB COUNT
1368 005062 113777 001276 175124 18$: MOVB     @#STMP0,@RSCS2  ;SET UNIT NUMBER
1369 005070 012777 000001 175104  ;MOV      @1,@RSCS1      ;TRY NOP OPERATION
1370 005076 032777 010000 175110  ;BIT      @BIT12,@RSCS2  ;NON EXISTANT DRIVE?
1371 005104 001011          BNE      16$            ;BRANCH IF YES
1372 005106 017702 175106          MOV      @RSDS,R2      ;GET DRIVE STATUS
1373 005112 042702 163577          BIC      @163577,R2    ;GET BITS 12, 11, & 7 ONLY
1374 005116 022702 010200          CMP      @10200,R2    ;IS DRIVE READY?
1375 005122 001002          BNE      16$            ;BRANCH IF NO
1376 005124 052700 000400          BIS      @BIT8,RO     ;SET DRIVE AVAILABLE BIT
1377 005130 006000          ROR      RO
1378 005132 052777 000040 175054 16$: BIS      @BITS,@RSCS2  ;CLEAR ANY ERROR BITS
1379 005140 005237 001276          INC      @#STMP0      ;SELECT NEXT UNIT
1380 005144 077132          SOB      R1,18$      ;CONTINUE
1381 005146 110037 001620          MOVB     RO,@#SYSSIZE+12 ;STORE IN TABLE
1382
1383 ;NEXT, DELETE XXDP UNIT 0 FROM TABLE
1384 005152 122737 000002 000041 15$: CMPB     @2,@#41      ;RK?
1385 005160 001004          BNE      19$            ;BRANCH IF NO
1386 005162 042737 000001 001610          BIC      @BIT0,@#SYSSIZE+2 ;MAKE UNIT ZERO NOT AVAILABLE
1387 005170 000420          BR      20$
1388 005172 113700 000041          19$: MOVB     @#41,RO     ;GET LOCATION 41
1389 005176 042700 177770          BIC      @177770,RO   ;GET LEAST SIG 3 BITS
1390 005202 000241          CLC
1391 005204 006100          ROL      RO           ;ENSURE C CLEAR
1392 005206 122700 000002          CMPB     @2,RO        ;ADJUST
1393 005212 002404          SLT      40$
1394 005214 042737 000001 001606          BIC      @BIT0,@#SYSSIZE ;BRANCH IF NO
1395 005222 000403          BR      20$
1396 005224 042760 000001 001612 40$: BIC      @BIT0,SYSSIZE+4(RO)
1397 005232 005227 177777          20$: INC      #-1
1398 005236 001055          BNE      LOOP          ;;BRANCH IF NOT FIRST TIME
1399 005240 104400 054764          TYPE     MSG25
1400 005244 013746 001470          MOV      @#OPT.CP,-(SP)
1401 005250 104402          TYPCC
1402 005252 104400 001327          TYPE     $CRLF
1403 005256 005737 001504          TST      @#OV          ;ACT11?
1404 005262 001043          BNE      LOOP          ;BRANCH IF YES
1405 005264 105737 003241          50$: TSTB     @#XXDPC    ;XXDP CHAIN MODE?
1406 005270 001040          BNE      LOOP          ;;BRANCH IF YES
1407 005272 105737 001200          TSTB     @#SPASS      ;FIRST PASS?
1408 005276 001035          BNE      LOOP          ;;BRANCH IF NO
1409 005300 032737 000200 177570          BIT      @SW7,@#SWR   ;INHIBIT SIZE TYPEOUT?
1410 005306 001031          BNE      LOOP          ;;BRANCH IF YES
1411 005310 004767 041604          JSR      PC,TYPSIZ    ;GO TYPE SYSTEM SIZE
1412 005314 104400 005322          TYPE     65$          ;;TYPE ASCIZ STRING
1413 005320 000417          BR      64$          ;;GET OVER THE ASCIZ
1414 ;65$: .ASCIZ /TYPE A CHARACTER TO CONTINUE/<CRLF>
1415 64$: CLR      @#PSW
1416 005360 005037 177776          WAIT
1417 005364 000001          JMP      @#SIZE       ;GO CHECK SYSTEM AGAIN
1418 005366 000137 004470          ;PROGRAM RESTARTS HERE AFTER RELOCATION ABOVE 28K IS COMPLETE.
1419

```

• 441

```

1421 005372 012706 000700 LOOP: MOV #SUPSTK,SP ;SET THE STACK...WILL BE DIFFERENT
1422 ;THAN KERN STACK WHEN IN OUTER MODE
1423 005376 012700 000004 MOV #ERRVEC,RO
1424 005402 013701 177776 MOV @#PSW,R1 ;GET CURRENT PSW
1425 005406 012720 053442 MOV #ERRPT,(RO)+ ;SET ERROR VEC
1426 005412 052701 000340 BIS #PR7,R1 ;SET PRIORITY 7 IN CURRENT PSW
1427 005416 042701 000020 BIC #BIT4,R1 ;CLEAR T BIT
1428 005422 010120 MOV R1,(RO)+
1429 005424 012720 053370 MOV #RESERR,(RO)+ ;SET RESERVED INST TRAP VECTOR
1430 005430 010120 MOV R1,(RO)+
1431 005432 012720 001472 MOV #SATRN,(RO)+ ;SET T BIT VEC
1432 005436 042701 000340 BIC #PR7,R1
1433 005442 005020 CLR (RO)+ ;SET TBIT VEC+2
1434 005444 005720 TST (RO)+ ;BUMP RO TO SCOPE VEC+2
1435 005446 005020 CLR (RO)+ ;SET SCOPE VEC+2
1436 005450 062700 000006 ADD #6,RO ;SET RO TO ERROR TRAP VEC
1437 005454 012720 000340 MOV #PR7,(RO)+ ;SET ERROR VEC
1438 005460 005720 TST (RO)+
1439 005462 012720 000340 MOV #PR7,(RO)+ ;SET TRAP VEC+2
1440 005466 012737 052642 000114 MOV #,PARSRV,@#CACHVEC ;SET PARITY ERROR VECTOR
1441 005474 052701 000340 BIS #PR7,R1
1442 005500 010137 000116 MOV R1,@#CACHVEC+2
1443 005504 012737 053274 000250 MOV #KTABRT,@#MMVEC ;SET KT11 ABORT VECTOR
1444 005512 010137 000252 MOV R1,@#MMVEC+2
1445 005516 042737 000340 177776 BIC #PR7,@#PSW
1446 ;*****
1447 ;*TEST 1 MEMORY VERIFICATION TEST
1448 ;*****
1449 005524 112737 000001 001202 ST1: MOVB #1,@#STSTNM ;LOAD TEST NUMBER
1450 005532 012767 000001 173556 MOV #1,$TIMES ;;DO 1 ITERATION
1451 005540 000004 SCOPE
1452
1453 .SBTTL START OF SECTION 0
1454 :0000000000000000 FIRST ADDRESS TO BE RELOCATED 00000000
1455 RELO: MOV PC,RO ;GET PC
1456 005544 005740 TST -(RO) ;RO CONTAINS THE ADDRESS OF RELO
1457 005546 010037 001512 MOV RO,@#FRSTAD ;SAVE
1458 005552 010700 MOV PC,RO ;GET CURRENT PC
1459 005554 162700 005554 SUB #,RO ;SUBTRACT RELOCATION FACTOR
1460 005560 010037 001506 MOV RO,@#FACTOR ;SAVE RELOCATION FACTOR
1461 005564 010737 001212 MOV PC,@#SLPERR ;SET LOOP ADDRESS
1462 005570 062737 000030 001212 ADD #30,@#SLPERR ;ADJUST
1463 005576 013737 001212 001210 MOV @#SLPERR,@#SLPADR
1464 005604 105737 001502 TSTB @#NEXEC ;BR IF TEST CODE TO BE EXECUTED
1465 005610 001402 BEQ .+6
1466 005612 000167 000720 JMP RELO
1467 ;MEMORY AND DISK (IF SELECTED) VERIFICATION TEST.
1468 005616 000167 000714 JMP 1$
1469 005622 177777 177777 177777 .WORD -1,-1,-1,-1,0,0,0,0
1470 005630 177777 000000 000000 .WORD -1,-1,-1,-1,0,0,0,0
1471 005636 000000 000000 000000 .WORD -1,-1,-1,-1,0,0,0,0
1472 005642 177777 177777 177777 .WORD -1,-1,-1,-1,0,0,0,0
1473 005650 177777 000000 000000 .WORD -1,-1,-1,-1,0,0,0,0
1474 005656 000000 000000 000000 .WORD -1,-1,-1,-1,0,0,0,0
1475 005662 177777 177777 177777 .WORD -1,-1,-1,-1,0,0,0,0

```

1477	005676	000000	000000						
1478	005702	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1479	005710	177777	000000	000000					
1480	005716	000000	000000						
1481	005722	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1482	005730	177777	000000	000000					
1483	005736	000000	000000						
1484	005742	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1485	005750	177777	000000	000000					
1486	005756	000000	000000						
1487	005762	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1488	005770	177777	000000	000000					
1489	005776	000000	000000						
1490	006002	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1491	006010	177777	000000	000000					
1492	006016	000000	000000						
1493	006022	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1494	006030	177777	000000	000000					
1495	006036	000000	000000						
1496	006042	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1497	006050	177777	000000	000000					
1498	006056	000000	000000						
1499	006062	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1500	006070	177777	000000	000000					
1501	006076	000000	000000						
1502	006102	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1503	006110	177777	000000	000000					
1504	006116	000000	000000						
1505	006122	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1506	006130	177777	000000	000000					
1507	006136	000000	000000						
1508	006142	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1509	006150	177777	000000	000000					
1510	006156	000000	000000						
1511	006162	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1512	006170	177777	000000	000000					
1513	006176	000000	000000						
1514	006202	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1515	006210	177777	000000	000000					
1516	006216	000000	000000						
1517	006222	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1518	006230	177777	000000	000000					
1519	006236	000000	000000						
1520	006242	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1521	006250	177777	000000	000000					
1522	006256	000000	000000						
1523	006262	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1524	006270	177777	000000	000000					
1525	006276	000000	000000						
1526	006302	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1527	006310	177777	000000	000000					
1528	006316	000000	000000						
1529	006322	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0			
1530	006330	177777	000000	000000					
1531	006336	000000	000000						

1533	006350	177777	000000	000000	
1534	006351	000000	000000	000000	
1535	006352	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
1536	006353	177777	000000	000000	
1537	006376	000000	000000	000000	
1538	006402	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
1539	006410	177777	000000	000000	
1540	006416	000000	000000	000000	
1541	006422	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
1542	006430	177777	000000	000000	
1543	006436	000000	000000	000000	
1544	006442	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
1545	006450	177777	000000	000000	
1546	006456	000000	000000	000000	
1547	006462	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
1548	006470	177777	000000	000000	
1549	006476	000000	000000	000000	
1550	006502	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
1551	006510	177777	000000	000000	
1552	006516	000000	000000	000000	
1553	006522	177777	177777	177777	.WORD -1,-1,-1,-1,0,0
1554	006530	177777	000000	000000	
1555	006536				
1556	006536	000004			15: RELO: SCOPE
1557	006540	010702			MOV PC,R2
1558	006542	062702	000012		ADD #12,R2
1559	006546	012707	034242		MOV #RELOC,PC ;GO RELOCATE PROGRAM CODE
1560	006552	000000			RELOO: .WORD 0
1561					;000000000000 LAST ADDRESS OF CODE TO BE RELOCATED 0000000000
1562					
1563					*****
1564					TEST 2 CHECK BRANCH INSTRUCTIONS
1565					*****
1566	006554	112737	000002	001202	TEST2: MOVB #2,\$STNM ;LOAD TEST NUMBER
1567	006562	012767	000001	172526	MOV #1,\$TIMES ;;DO 1 ITERATION
1568	006570	000004			SCOPE
1569					
1570					.SBTTL START OF SECTION 1
1571					:1111111111111 FIRST ADDRESS TO BE RELOCATED 1111111111
1572	006572	010700			REL1: MOV PC,RO ;GET PC
1573	006574	005740			TST -(RO) ;RO CONTAINS THE ADDRESS OF REL1
1574	006576	010037	001512		MOV RO,\$FRSTAD ;SAVE
1575	006602	010700			MOV PC,RO ;GET CURRENT PC
1576	006604	162700	006604		SUB #,RO ;SUBTRACT RELOCATION FACTOR
1577	006610	010037	001506		MOV RO,\$FACTOR ;SAVE RELOCATION FACTOR
1578	006614	010737	001212		MOV PC,\$SLPERR ;SET LOOP ADDRESS
1579	006620	062737	000030	001212	ADD #30,\$SLPERR ;ADJUST
1580	006626	013737	001212	001210	MOV \$SLPERR,\$SLPADR
1581	006634	105737	001502		TSTB \$NEXEC ;BR IF TEST CODE TO BE EXECUTED
1582	006640	001402			BEQ +6
1583	006642	000167	004052		JMP RELE1
1584					
1585	006646	000257			;CC'S=0000
1586	006650	103407			CCC ;SAME AS BLO
1587	006652	102406			BVS CCO CCO CCO

1589	006656	100404	BMI	CC0
1590	006660	002403	BLT	CC0
1591	006662	003402	BLE	CC0
1592	006664	101401	BLQS	CC0
1593	006666	101001	BHI	.+4
1594	006670	104000	CC0:	HLT
1595				
1596			:CONTINUE	
1597	006672	000270	SEN	
1598	006674	100003	BPL	CC1
1599	006676	002002	BGE	CC1
1600	006700	003001	BGT	CC1
1601	006702	002401	BLT	.+4
1602	006704	104000	CC1:	HLT
1603				
1604			:CONTINUE	
1605	006706	000262	SEV	
1606	006710	102003	BVC	CC2
1607	006712	002402	BLT	CC2
1608	006714	003401	BLE	CC2
1609	006716	002001	BGE	.+4

;ONE OF THE ABOVE BRANCHES FAILED

;CC'S=1000

;ONE OF THE ABOVE BRANCHES FAILED

;CC'S=1010

```

1611
1612 ;CONTINUE
1613 006722 000261 SEC ;CC'S=1011
1614 006724 103002 BCC CC3
1615 006726 101001 BHI CC3
1616 006730 003001 BGT .+4
1617 006732 104000 CC3: HLT ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
1618
1619 ;CONTINUE
1620 006734 000264 SEZ ;CC'S=1111
1621 006736 001003 BNE CC4
1622 006740 003002 BGT CC4
1623 006742 101001 BHI CC4
1624 006744 003401 BLE .+4
1625 006746 104000 CC4: HLT ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
1626
1627 ;*****
1628 ;*TEST 3 TEST UNIARY CONDITION CODES
1629 006750 112737 000003 001202 TST3: MOVB #3,2#STSTNM ;LOAD TEST NUMBER
1630 006756 000004 SCOPE
1631 ;CLR RO
1632 006760 000277 SCC
1633 006762 000244 CLZ
1634 006764 005000 CLR RO ;RO=0,CC'S=0100
1635 006766 103404 BCS CLRO
1636 006770 102403 BVS CLRO
1637 006772 001002 BNE CLRO
1638 006774 100401 BMI CLRO
1639 006776 003401 BLE .+4
1640 007000 104000 CLRO: HLT ;ERROR! INCORRECT CC'S AFTER CLR
1641
1642 007002 000277 SCC
1643 007004 000244 CLZ
1644 007006 005700 TST RO ;RO=0,CC'S=0100
1645 007010 103404 BCS TSTO
1646 007012 102403 BVS TSTO
1647 007014 001002 BNE TSTO
1648 007016 100401 BMI TSTO
1649 007020 101401 BLOS .+4
1650 007022 104000 TSTO: HLT ;ERROR! INCORRECT CC'S AFTER TST
1651
1652 007024 000257 CCC
1653 007026 000266 +SEZ!SEV
1654 007030 005100 COM RO ;RO=-1,CC'S=1001
1655 007032 103004 BCC COMO
1656 007034 102403 BVS COMO
1657 007036 001402 BEQ COMO
1658 007040 100001 BPL COMO
1659 007042 002401 BLT .+4
1660 007044 104000 COMO: HLT ;ERROR! INCORRECT CC'S AFTER COM
1661
1662 007046 000261 SEC
1663 007050 005500 ADC RO ;RO=000000,CC'S=0101
1664 007052 103003 BCC ADCO
1665 007054 102402 BVS ADCO

```

1667	007060	002001		BGE	.+4	
1668	007062	104000	ADCO:	HLT		;ERROR! INCORRECT CC'S AFTER ADC
1669						
1670	007064	000261		SEC		
1671	007066	006300		ROR	RO	;RO=100000,CC'S=1010
1672	007070	103404		BCS	RORO	
1673	007072	102003		BVC	RORC	
1674	007074	001402		BEQ	RORO	
1675	007076	100001		BPL	RORO	
1676	007100	003001		BGT	.+4	
1677	007102	104000	RORO:	HLT		;ERROR! INCORRECT CC'S AFTER ROR
1678	007104	000277		SCC		
1679	007106	000242		CLV		
1680	007110	005300		DEC	RO	;RO=077777,CC'S=0011
1681	007112	103004		BCC	DECO	
1682	007114	102003		BVC	DECO	
1683	007116	001402		BEQ	DECO	
1684	007120	100401		BMI	DECO	
1685	007122	003401		BLE	.+4	
1686	007124	104000	DECO:	HLT		;ERROR! INCORRECT CC'S AFTER DEC
1687						
1688	007126	000257		CCC		
1689	007130	005200		INC	RO	;RO=100000,CC'S=1010
1690	007132	103404		BCS	INCO	
1691	007134	102003		BVC	INCO	
1692	007136	001402		BEQ	INCO	
1693	007140	100001		BPL	INCO	
1694	007142	003001		BGT	.+4	
1695	007144	104000	INCO:	HLT		;ERROR! INCORRECT CC'S AFTER INC
1696						
1697	007146	000277		SCC		
1698	007150	000242		CLV		
1699	007152	005400		NEG	RO	;RO=100000,CC'S=1011
1700	007154	103003		BCC	NEGO	
1701	007156	102002		BVC	NEGO	
1702	007160	001401		BEQ	NEGO	
1703	007162	002001		BGE	.+4	
1704	007164	104000	NEGO:	HLT		;ERROR! INCORRECT CC'S AFTER NEG
1705						
1706	007166	000261		SEC		
1707	007170	006300		ASL	RO	;RO=000000,CC'S=0111
1708	007172	103004		BCC	ASLO	
1709	007174	102003		BVC	ASLO	
1710	007176	001002		BNE	ASLO	
1711	007200	100401		BMI	ASLO	
1712	007202	101401		BLOS	.+4	
1713	007204	104000	ASLO:	HLT		;ERROR! INCORRECT CC'S AFTER ASL
1714						
1715	007206	006100		ROL	RO	;RO=000001,CC'S=0000
1716	007210	103402		BCS	ROLO	
1717	007212	003401		BLE	ROLO	
1718	007214	002001		BGE	.+4	
1719	007216	104000	ROLO:	HLT		;ERROR! INCORRECT CC'S AFTER ROL
1720						
1721	007220	006200		ASR	RO	;RO=000000,CC'S=0111



```

1723 007224 102002          BVC  ASRO
1724 007226 001001          BNE  ASRO
1725 007230 002401          BLT  .+4
1726 007232 104000          ASRO: HLT          ;ERROR! INCORRECT CC'S AFTER ASR
1727
1728 007234 000277          SCC
1729 007236 005600          SBC  RO          ;RO=-1,CC'S=1001
1730 007240 103002          BCC  SBCO
1731 007242 102401          BVS  SBCO
1732 007244 003401          BLE  .+4
1733 007246 104000          SBCO: HLT          ;ERROR! INCORRECT CC'S AFTER SBC
1734
1735 007250 005400          NEG  RO          ;RO=000001,CC'S=00001
1736 007252 000300          SWAB RO          ;RO=000400 ,CC'S=0100
1737 007254 103403          BCS  SWABO
1738 007256 102402          BVS  SWABO
1739 007260 001001          BNE  SWABO
1740 007262 002001          BGE  .+4
1741 007264 104000          SWABO: HLT          ;ERROR! INCORRECT CC'S AFTER SWAB
1742
1743
1744
1745 007266 112737 000004 001202  ;*****
1746 007274 000004          ;*TEST 4 CHECK REGISTER SELECTION
1747 007276 012737 000005 001316  ;*****
1748 007304 005000          ;LOAD TEST NUMBER
1749 007306 000277          ;*****
1750 007310 006100          ;*****
1751 007312 010002          ;*****
1752 007314 006302          ;*****
1753 007316 010203          ;*****
1754 007320 006303          ;*****
1755 007322 010304          ;*****
1756 007324 006304          ;*****
1757 007326 010405          ;*****
1758 007330 006305          ;*****
1759 007332 010546          ;*****
1760 007334 050416          ;*****
1761 007336 050316          ;*****
1762 007340 050216          ;*****
1763 007342 050016          ;*****
1764 007344 022726 000037          ;*****
1765 007350 001401          ;*****
1766 007352 104000          ;*****
1767
1768
1769
1770 007354 000257          ;CHECK THAT ALL BITS CAN BE SET & CLEARED IN ALL REGISTERS
1771 007356 112700 000377          CCC
1772 007362 006100          MOVB #377,RO    ;SET ALL BITS (MOVB EXTENDS SIGN)
1773 007364 103776          ROL  RO        ;ROTATE A 0 THROUGH ALL BIT
1774 007366 005200          BCS  1$       ;POSITIONS
1775 007370 001401          INC  RO        ;FINAL RESULT IS -1
1776 007372 104000          BEQ  .+4
1777

```

# E05

CEQKCC PDP 11/70 CPU EXERCISOR MACY11 30A(1052) 03-MAR-78 13:15 PAGE 37  
 CEQKCC.P11 03-MAR-78 13:13 T4 CHECK REGISTER SELECTION

SEQ 0056

```

1779 007400 005002
1780 007402 000261          25:  CLR      R2
1781 007404 006002          SEC      R2          ; ROTATE 1 THROUGH ALL BIT POSITS
1782 007406 005300          ROR      R2          ; DECREMENT SHIFT COUNT
1783 007410 001374          DEC      R0
1784 007412 005102          BNE     R2          ; R2 SHOULD CONTAIN -1
1785 007414 001401          COM     R2          ; ERROR! CHECK R2 SHOULD = 0
1786 007416 104000          BEQ     .+4
1787
1788 007420 012703 100000          HLT
1789 007424 006203          35:  MOV     #100000,R3
1790 007426 103376          ASR     R3          ; EXTEND 1 BIT THROUGH ALL POSITIONS
1791 007430 005203          BCC     R3
1792 007432 001401          INC     R3
1793 007434 104000          BEQ     .+4
1794
1795 007436 112704 177401          HLT          ; ERROR!
1796 007442 060404          45:  MOV     #177401,R4
1797 007444 103376          ADD     R4,R4      ; R4=1
1798 007446 005704          BCC     R4          ; HAS THE AFFECT OF SHIFTING A BIT
1799 007450 001401          TST     R4          ; THROUGH ALL POSITIONS
1800 007452 104000          BEQ     .+4          ; RESULT SHOULD BE 0
1801
1802 007454 012705 000001          HLT
1803 007460 006305          55:  MOV     #1,R5
1804 007462 102376          ASL     R5
1805 007464 006305          BVC     R5
1806 007466 103002          ASL     R5
1807 007470 005705          BCC     R5
1808 007472 001401          TST     R5
1809 007474 104000          BEQ     .+4
1810
1811          65:  HLT
1812          ; CHECK REGISTER VOLITILITY
1813 007476 005002          CLR     R2
1814 007500 005102          COM     R2          ; R2=-1
1815 007502 010203          MOV     R2,R3
1816 007504 000257          CCC
1817 007506 006002          ROR     R2          ; R2=LOOP COUNT
1818 007510 006202          ASR     R2
1819 007512 010304          75:  MOV     R3,R4
1820 007514 005302          DEC     R2          ; DECREMENT LOOP COUNT
1821 007516 001375          BNE     R2
1822 007520 005203          INC     R3          ; CHECK R3
1823 007522 001002          BNE     R3
1824 007524 005204          INC     R4          ; CHECK R4
1825 007526 001401          BEQ     .+4
1826 007530 104000          85:  HLT
1827
1828 007532 032737 000020 177776          ; CHECK TRANSFER OF REGISTER DATA BETWEEN THE G5 AND G0 REGISTERS
1829 007540 001050          GSTST: BIT #20,0#PSW ; CHECK IF 'T' BIT IS SET
1830 007542 010627          BNE     R5          ; SKIP TEST IF 'T' BIT SET
1831 007544 000000          MOV     SP,(PC)+ ; SAVE STACK PTR
1832 007546 010727          .WORD 0           ; CONTAINS SAVED STACK PTR
1833 007550 000000          MOV     PC,(PC)+ ; LOAD DATA. THE CURRENT PC IS USED AS
                                ; DATA. IF THIS TEST FAILS 25 CON-
    
```

# F05

```

1835 007552 005267 177772          INC      25          ;MAKE ODD TO CHECK BIT 0
1836 007556 016700 177766          MOV      25,R0      ;LOAD GD REGISTER 0
1837 007562 010001          MOV      R0,R1      ;TRANSFER GS REG 0 TO GD REG 1
1838 007564 010102          MOV      R1,R2      ;AND GS REG 1 TO GD REG 2
1839 007566 010203          MOV      R2,R3      ;ETC...
1840 007570 010304          MOV      R3,R4
1841 007572 010405          MOV      R4,R5
1842 007574 152737 000340 177776      BISB    #340,2#PSW   ;SET PRIORITY LEVEL 7
1843 007602 010506          MOV      R5,SP      ;TRANSFER GS REG 5 TO GD STK PTR
1844 007604 010627          MOV      SP,(PC)+   ;TRANSFER GS STK PTR TO MEMORY
1845 007606 000000          .WORD   0           ;CONTAINS GS STACK PTR
1846 007610 016706 177730          MOV      15,SP      ;RESTORE STK PTR NEEDED FOR HLT/SCOPE
1847 007614 142737 000340 177776      BICB    #340,2#PSW   ;SET PRIORITY LEVEL 0
1848 007622 026700 177760          CMP      45,R0      ;COMPARE GS/GD STACK WITH GS REG 0
1849 007626 001004          BNE      55         ;BRANCH IF THEY WERE NOT =
1850 007630 006367 177714          ASL      25         ;SHIFT TEST DATA UNTIL = 000000
1851 007634 001350          BNE      35
1852 007636 000411          BR       65
1853 007640          55:  MOV      R0,-(SP)   ;GET GS REG 0
1854 007642          MOV      R1,-(SP)   ;ETC...
1855 007644          MOV      R2,-(SP)
1856 007646          MOV      R3,-(SP)
1857 007650          MOV      R4,-(SP)
1858 007652          MOV      R5,-(SP)
1859 007654 104000          HLT
1860          ;ERROR! DATA IN GS STK PTR NOT = GS REG 0
1861 007656 016706 177662          MOV      15,SP     ;GS REG 0-GS REG 5 ARE ON THE STACK
1862          ;RESTORE STACK PTR
1863 007662          65:
1864          75:
1865          ;*****
1866          ;#TEST 5 TEST UNIARY WORD INSTRUCTIONS USING ADDRESS MODE 1
1867          ;*****
1867 007662 112737 000005 001202      TESTS: MOV      #5,2#STSTNM ;LOAD TEST NUMBER
1868 007670 000004          SCOPE
1869 007672 012737 000005 001316      MOV      #5,2#STTIMES
1870 007700 000401          BR       .+4
1871 007702 000000          .WORD   0           ;RESERVE ADDRESS FOR TESTS
1872 007704 010702          MOV      PC,R2
1873 007706 162702 000004          SUB      #4,R2      ;R2 POINTS TO RESERVED WORD
1874 007712 005012          CLR      (R2)      ;PRESET (R2)
1875
1876 007714 000261          SEC
1877 007716 006012          ROR      (R2)      ;(R2)=100000,CC=1010
1878 007720 101402          BLOS    ROR1
1879 007722 100001          BPL     ROR1
1880 007724 002001          BGE     .+4
1881 007726 104000          ROR1:  HLT        ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
1882
1883 007730          CCC
1884 007732 000261          SEC
1885 007734 005312          DEC      (R2)      ;(R2)=077777,CC=0011
1886 007736 103001          DEC     DEC1
1887 007740 003401          BCC     .+4
1888 007742 104000          BLE     .+4
1889          DEC1:  HLT        ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
    
```

1891	007746	000261	SEC		
1892	007750	005512	ADC	(R2)	; (R2)=100000,CC=1010
1893	007752	103403	BCS	ADC1	
1894	007754	102002	BVC	ADC1	
1895	007756	100001	BPL	ADC1	
1896	007760	001001	BNE	.+4	
1897	007762	104000	ADC1:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
1898					
1899	007764	006112	ROL	(R2)	; (R2)=000000,CC=0111
1900	007766	103003	BCC	ROL1	
1901	007770	102002	BVC	ROL1	
1902	007772	001001	BNE	ROL1	
1903	007774	100001	BPL	.+4	
1904	007776	104000	ROL1:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
1905					
1906	010000	006112	ROL	(R2)	; (R2)=000001,CC=0000
1907	010002	101402	BLOS	ROL1A	; BRANCH IF C OR Z IS SET
1908	010004	102401	BVS	ROL1A	
1909	010006	100001	BPL	.+4	
1910	010010	104000	ROL1A:	HLT	
1911					
1912	010012	006212	ASR	(R2)	; (R2)=000000,CC=0111
1913	010014	103003	BCC	ASR1	
1914	010016	102002	BVC	ASR1	
1915	010020	001001	BNE	ASR1	
1916	010022	100001	BPL	.+4	
1917	010024	104000	ASR1:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
1918					
1919	010026	006012	ROR	(R2)	; (R2)=100000,CC=1010
1920	010030	103403	BCS	ROR1A	
1921	010032	102002	BVC	ROR1A	
1922	010034	001401	BEQ	ROR1A	
1923	010036	100401	BMI	.+4	
1924	010040	104000	ROR1A:	HLT	
1925					
1926	010042	000261	SEC		
1927	010044	005212	INC	(R2)	; (R2)=100001,CC=1001
1928	010046	103003	BCC	INC1	
1929	010050	102402	BVS	INC1	
1930	010052	001401	BEQ	INC1	
1931	010054	100401	BMI	.+4	
1932	010056	104000	INC1:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
1933					
1934	010060	005612	SBC	(R2)	; (R2)=100000,CC=1000
1935	010062	103403	BCS	SBC1	
1936	010064	102402	BVS	SBC1	
1937	010066	001401	BEQ	SBC1	
1938	010070	100401	BMI	.+4	
1939	010072	104000	SBC1:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
1940					
1941	010074	000261	SEC		
1942	010076	005612	SBC	(R2)	; (R2)=077777,CC=0010
1943	010100	103403	BCS	SBC1A	
1944	010102	102002	BVC	SBC1A	
1945	010104	001401	BEQ	SBC1A	

H05

```

1947 010110 104000 SBC1A: HLT ;ERROR! INCORRECT CC'S AS SHOEN ABOVE
1948
1949 010112 000261 SEC
1950 010114 005512 ADC (R2) ;(R2)=100000,CC=1010
1951 010116 100401 BMI .+4
1952 010120 104000 HLT
1953
1954 010122 000261 SEC
1955 010124 006312 ASL (R2) ;(R2)=000000,CC=0111
1956 010126 103003 BCC ASL1
1957 010130 102002 BVC ASL1
1958 010132 001001 BNE ASL1
1959 010134 100001 BPL .+4
1960 010136 104000 ASL1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
1961
1962 010140 005112 COM (R2) ;(R2)=177777,CC=1001
1963 010142 103002 BCC COM1
1964 010144 102401 BVS COM1
1965 010146 100401 BMI .+4
1966 010150 104000 COM1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
1967
1968 010152 000250 CLN
1969 010154 005712 TST (R2) ;(R2)=177777,CC=1000
1970 010156 103403 BCS TEST1
1971 010160 102402 BVS TEST1
1972 010162 100001 BPL TEST1
1973 010164 001001 BNE .+4
1974 010166 104000 TEST1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
1975
1976 010170 000262 SEV
1977 010172 005412 NEG (R2) ;(R2)=000001,CC=0000
1978 010174 103002 BCC NEG1
1979 010176 102401 BVS NEG1
1980 010200 001001 BNE .+4
1981 010202 104000 NEG1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
1982
1983 010204 005312 DEC (R2) ;(R2)=000000,CC=0101
1984 010206 103001 BCC DEC1A
1985 010210 001401 BEQ .+4
1986 010212 104000 DEC1A: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
1987
1988 ;*****
1989 ;*TEST 6 CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1
1990 ;*****
1990 010214 112737 000006 001202 †ST6: MOVB #6,30†ST6 ;LOAD TEST NUMBER
1991 010222 000004 SCOPE
1992 010224 000401 BR .+4 ;RESERVE A WORD
1993 010226 000000 .WORD 0 ;ADDRESS RESERVED FOR TESTS
1994 010230 010703 MOV PC,R3
1995 010232 162703 000004 SUB #4,R3 ;R3 POINTS TO EVEN BYTE OF WORD
1996 010236 010304 MOV R3,R4 ;R4 POINTS TO ODD BYTE OF WORD
1997 010240 005204 INC R4
1998 010242 005013 CLR (R3) ;PRESET DATA
1999
2000 010244 000261 15: SEC
2001 010246 105513 ADCB (R3) ;ADD CARRY TO EVEN BYTE

```

# I05

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052)  
T6

03-MAR-78 13:15 PAGE 41  
CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1

SEQ 0060

2003	010252	105214		INCB	(R4)		; INCREMENT ODD BYTE
2004	010254	000773		BR	15		
2005	010256	102401	25:	BVS	.+4		; (R3)=077600=(0774)(200),CC=1010
2006	010260	104000		HLT			
2007	010262	000242		CLV			
2008	010264	105214		INCB	(R4)		; (R3)=100200=(1000)(200),CC=1010
2009	010266	103402		BCS	INCB1		
2010	010270	102001		BVC	INCB1		
2011	010272	100401		BMI	.+4		
2012	010274	104000	INCB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE
2013							
2014	010276	106114		ROLB	(R4)		; (R3)=000200=(0000)(200),CC=0111
2015	010300	103002		BCC	ROLB1		
2016	010302	102001		BVC	ROLB1		
2017	010304	001401		BEQ	.+4		
2018	010306	104000	ROLB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE
2019							
2020	010310	105614		SBCB	(R4)		; (R3)=177600=(1774)(200), CC=1001
2021	010312	103002		BCC	SBCB1		
2022	010314	102401		BVS	SBCB1		
2023	010316	100401		BMI	.+4		
2024	010320	104000	SBCB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE
2025							
2026	010322	106313		ASLB	(R3)		; (R3)=177400,CC=0111
2027	010324	103002		BCC	ASLB1		
2028	010326	102001		BVC	ASLB1		
2029	010330	001401		BEQ	.+4		
2030	010332	104000	ASLB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE
2031							
2032	010334	105413		NEGB	(R3)		; (R3)=177400,CC=0100
2033	010336	103402		BCS	NEGB1		
2034	010340	102401		BVS	NEGB1		
2035	010342	001401		BEQ	.+4		
2036	010344	104000	NEGB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE
2037							
2038	010346	000277		SCC			
2039	010350	105313		DECB	(R3)		; (R3)=177777,CC=1001
2040	010352	103002		BCC	DECB1		
2041	010354	102401		BVS	DECB1		
2042	010356	001001		BNE	.+4		
2043	010360	104000	DECB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE
2044							
2045	010362	000241		CLC			
2046	010364	106013		RORB	(R3)		; (R3)=177577,CC=0011
2047	010366	103002		BCC	RORB1		
2048	010370	102001		BVC	RORB1		
2049	010372	100001		BPL	.+4		
2050	010374	104000	RORB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE
2051							
2052	010376	000241		CLC			
2053	010400	105114		COMB	(R4)		; (R3)=000177,CC=0101
2054	010402	103002		BCC	COMB1		
2055	010404	102401		BVS	COMB1		
2056	010406	001401		BEQ	.+4		
2057	010410	104000	COMB1:	HLT			; ERROR! INCORRECT CC'S AS SHOWN ABOVE

# J05

CEQKCC PDP 11-70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052)  
T6

03-MAR-78 13:15 PAGE 42  
CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1

SEQ 0061

2059	010412	106213	1S:	ASRB	(R3)		;SHIFT EVEN BYTE UNTIL V CLEARS
2060	010414	102002		BVC	2S		
2061	010416	105514		ADCB	(R4)		;AND ADD CARRY TO ODD BYTE
2062	010420	000774		BR	1S		
2063	010422	103401	2S:	BCS	ASRB1		
2064	010424	001401		BEQ	.+4		
2065	010426	104000	ASRB1:	HLT			;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2066							
2067	010430	106214		ASRB	(R4)		
2068	010432	106214		ASRB	(R4)		; (R3)=000400,CC=0011
2069	010434	103002		BCC	ASRB1A		
2070	010436	102001		BVC	ASRB1A		
2071	010440	001001		BNE	.+4		
2072	010442	104000	ASRB1A:	HLT			;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2073							
2074	010444	105314		DECB	(R4)		; (R3)=000000,CC=0100
2075	010446	001401		BEQ	.+4		
2076	010450	104000		HLT			;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2077							
2078	010452	000261		SEC			
2079	010454	106014		RORB	(R4)		; (R3)=100000,CC=1010
2080	010456	103402		BCS	RORB1A		
2081	010460	102001		BVC	RORB1A		
2082	010462	100401		BMI	.+4		
2083	010464	104000	RORB1A:	HLT			;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2084							
2085	010466	000242		CLV			
2086	010470	105314		DECB	(R4)		; (R3)=077400,CC=0100
2087	010472	102401		BVS	.+4		
2088	010474	104000		HLT			
2089							
2090	010476	000261		SEC			
2091	010500	105313		DECB	(R3)		; (R3)=077777,CC=1001
2092	010502	103002		BCC	DECB1A		
2093	010504	102401		BVS	DECB1A		
2094	010506	100401		BMI	.+4		
2095	010510	104000	DECB1A:	HLT			;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2096							
2097	010512	000277		SCC			
2098	010514	000313		SWAB	(R3)		; (R3)=177577=[1774][177],CC=0000
2099	010516	103402		BCS	SWAB1		
2100	010520	102401		BVS	SWAB1		
2101	010522	100001		BPL	.+4		
2102	010524	104000	SWAB1:	HLT			;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2103							
2104	010526	105714		TSTB	(R4)		; (R3)=177577=[1774][177],CC=1000
2105	010530	103402		BCS	TSTB1		
2106	010532	102401		BVS	TSTB1		
2107	010534	100401		BMI	.+4		
2108	010536	104000	TSTB1:	HLT			;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2109							
2110	010540	105014		CLRB	(R4)		; (R3)=000177=[0000][177],CC=0100
2111	010542	001401		BEQ	.+4		
2112	010544	104000		HLT			
2113	010546	106313		ASLB	(R3)		; (R3)=000376 ,CC=1010

K05

CEQKCC PDP 11/70 CPU EXERCISOR  
 CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 43  
 T6 CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1

SEQ 0062

2115	010552	102001			BVC	ASLB1A		
2116	010554	100401			BMI	.+4		
2117	010556	104000			ASLB1A:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2118								
2119	010560	105113			COMB	(R3)		;(R3)=000001,CC=0001
2120	010562	103002			BCC	COMB1A		
2121	010564	102401			BVS	COMB1A		
2122	010566	100001			BPL	.+4		
2123	010570	104000			COMB1A:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2124								
2125	010572	000313			SWAB	(R3)		;(R3)=000400, CC=0100
2126	010574	001401			BEQ	.+4		
2127	010576	104000			HLT			
2128								
2129	010600	105213			INCB	(R3)		
2130	010602	000261			SEC			
2131	010604	105613			SBCB	(R3)		;(R3)=000400,CC=0100
2132	010606	001401			BEQ	.+4		
2133	010610	104000			HLT			
2134	010612	022713	000400		CMP	#400,(R3)		;CHECK REMAINING RESULT
2135	010616	001401			BEQ	.+4		
2136	010620	104000			HLT			
2137								
2138					:***** :*TEST 7 CHECK UNIARY WORD OPS USING ADDRESS MODES 2 & 4 :*****			
2139					TST7:	MOVW	#7,#STSTNM	;LOAD TEST NUMBER
2140	010622	112737	000007	001202				
2141	010630	000004			SCOPE			
2142	010632	000401			BR	.+4		
2143	010634	000000			.WORD	0		;ADDRESS RESERVED FOR TESTS
2144	010636	010704			MOV	PC,R4		
2145	010640	162704	000004		SUB	#4,R4		;R4 AND R5 POINT TO
2146	010644	010405			MOV	R4,R5		;RESERVED WORD
2147	010646	005015			CLR	(R5)		;PRESET DATA=0
2148								
2149	010650	000277			SCC			
2150	010652	000244			CLZ			
2151	010654	005725			TST	(R5)+		;(R5)=000000,CC=0100
2152	010656	103402			BCS	TEST2		
2153	010660	102401			BVS	TEST2		
2154	010662	001401			BEQ	.+4		
2155	010664	104000			TEST2:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2156								
2157	010666	005145			COM	-(R5)		;(R5)=177777,CC=1001
2158	010670	103001			BCC	COM4		
2159	010672	100401			BMI	.+4		
2160	010674	104000			COM4:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2161								
2162	010676	000241			CLC			
2163	010700	006024			ROR	(R4)+		;(R4)=077777,CC=0011
2164	010702	103002			BCC	ROR2		
2165	010704	102001			BVC	ROR2		
2166	010706	100001			BPL	.+4		
2167	010710	104000			ROR2:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2168								
2169	010712	000257			CCC			



LOS

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052)  
T7

03-MAR-78 13:15 PAGE 44  
CHECK UNIARY WORD OPS USING ADDRESS MODES 2 & 4

SEQ 0063

2171	010716	102002	BVC	INC4	
2172	010720	001401	BEQ	INC4	
2173	010722	100401	BMI	.+4	
2174	010724	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2175					
2176	010726	000261	SEC		
2177	010730	000324	SWAB	(R4)+	; (R4)=000200,CC=1000
2178	010732	103401	BCS	SWAB2	
2179	010734	100401	BMI	.+4	
2180	010736	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2181					
2182	010740	005425	NEG	(R5)+	; (R5)=177600,CC=1001
2183	010742	103001	BCC	NEG2	
2184	010744	100401	BMI	.+4	
2185	010746	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2186					
2187	010750	005044	CLR	-(R4)	; (R4)=000000,CC=0100
2188	010752	001401	BEQ	.+4	
2189	010754	104000	HLT		
2190					
2191	010756	000261	SEC		
2192	010760	006045	ROR	-(R5)	; (R5)=100000,CC=1010
2193	010762	000261	SEC		
2194	010764	005525	ADC	(R5)+	; (R5)=100001,CC=1000
2195	010766	102401	BVS	ADC2	
2196	010770	100401	BMI	.+4	
2197	010772	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2198					
2199	010774	000262	SEV		
2200	010776	006224	ASR	(R4)+	; (R4)=140000,CC=1001
2201	011000	103002	BCC	ASR2	
2202	011002	102401	BVS	ASR2	
2203	011004	100401	BMI	.+4	
2204	011006	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2205					
2206	011010	000262	SEV		
2207	011012	006144	ROL	-(R4)	; (R4)=100001, CC=1001
2208	011014	103002	BCC	ROL4	
2209	011016	102401	BVS	ROL4	
2210	011020	100401	BMI	.+4	
2211	011022	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2212					
2213	011024	005645	SBC	-(R5)	; (R5)=100000,CC=1000
2214	011026	103001	BCC	.+4	
2215	011030	104000	HLT		;ERROR! 'C' BIT FAILED TO CLEAR
2216					
2217	011032	005325	DEC	(R5)+	; (R5)=077777,CC=0010
2218	011034	103402	BCS	DEC2	
2219	011036	102001	BVC	DEC2	
2220	011040	100001	BPL	.+4	
2221	011042	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2222					
2223	011044	006324	ASL	(R4)+	; (R4)=177776,CC=1010
2224	011046	102401	BVS	.+4	
2225	011050	104000	HLT		

M05

CEOKCC POP 11/70 CPU EXERCISOR  
CEOKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 45  
T7 CHECK UNIARY WORD OPS USING ADDRESS MODES 2 & 4

SEQ 0064

2227	011054	103003			BCC	ASL4	
2228	011056	102402			BVS	ASL4	
2229	011060	001401			BEQ	ASL4	
2230	011062	100401			BMI	.+4	
2231	011064	104000			HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2232							
2233	011066	022724	177774		CMP	#177774,(R4)+	
2234	011072	001401			BEQ	.+4	
2235	011074	104000			HLT		
2236	011076	020405			CMP	R4,R5	
2237	011100	001401			BEQ	.+4	
2238	011102	104000			HLT		
2239							
2240					;*****		
2241					;*TEST 10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4		
2242					;*****		
2243	011104	112737	000010	001202	↑ST10:	MOVB #10,#\$TSTNM	;LOAD TEST NUMBER
2244	011112	000004			SCOPE		
2245	011114	000401			BR	.+4	;RESERVE A WORD
2246	011116	000000			.WORD	0	;RESERVED WORD
2247	011120	010705			MOV	PC,R5	
2248	011122	162705	000004		SUB	#4,R5	;R5 POINTS TO EVEN BYTE OF RESERVED WORD
2249	011126	010500			MOV	R5,R0	
2250	011130	010002			MOV	R0,R2	
2251	011132	005202			INC	R2	;R2 POINTS TO ODD BYTE OF RESERVED WORD
2252	011134	005010			CLR	(R0)	;PRESET
2253							
2254	011136	000277			SCC		
2255	011140	000241			CLC		
2256	011142	105125			COMB	(R5)+	; (R0)=000377,CC=1001
2257	011144	103002			BCC	COMB2	
2258	011146	102401			BVS	COMB2	
2259	011150	100401			BMI	.+4	
2260	011152	104000			HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2261	011154	105542			ADCB	-(R2)	; (R0)=000000,CC=0101
2262	011156	001401			BEQ	.+4	
2263	011160	104000			HLT		;ERROR! INCORRECT RESULT AS SHOWN ABOVE
2264	011162	105525			ADCB	(R5)+	; (R0)=000400,CC=0000
2265	011164	103401			BCS	ADCB2	
2266	011166	001001			BNE	.+4	
2267	011170	104000			HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2268							
2269	011172	000263			+SEC!SEV		
2270	011174	106045			RORB	-(R5)	; (R0)=100000,CC=1001
2271	011176	103003			BCC	RORB4	
2272	011200	102402			BVS	RORB4	
2273	011202	001401			BEQ	RORB4	
2274	011204	100401			BMI	.+4	
2275	011206	104000			HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2276							
2277	011210	000277			SCC		
2278	011212	106122			ROLB	(R2)+	; (R0)=100001,CC=0000
2279	011214	103403			BCS	ROLB2	
2280	011216	102402			BVS	ROLB2	
2281	011220	001401			BEQ	ROLB2	

N05

CEQKCC PDP 11/70 CPU EXERCISOR MACY11 30A(1052) 03-MAR-78 13:15 PAGE 46  
 CEQKCC.P11 03-MAR-78 13:13 T10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4

SEQ 0065

2283	011224	104000	ROLB2:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2284						
2285	011226	000257		CCC		
2286	011230	106225		ASRB	(R5)+	;(R0)=140001, CC=1010
2287	011232	103402		BCS	ASRB2	
2288	011234	102001		BVC	ASRB2	
2289	011236	100401		BMI	.+4	
2290	011240	104000	ASRB2:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2291						
2292	011242	105242		INCB	-(R2)	;(R0)=140002, CC=0000
2293	011244	000277		SCC		
2294	011246	106222		ASRB	(R2)+	;(R0)=140001, CC=0000
2295	011250	103402		BCS	ASRB2A	
2296	011252	102401		BVS	ASRB2A	
2297	011254	100001		BPL	.+4	
2298	011256	104000	ASRB2A:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2299						
2300	011260	000266		+SEZ:SEV		;SET Z, V
2301	011262	106345		ASLB	-(R5)	;(R0)=100001, CC=1001
2302	011264	103003		BCC	ASLB4	
2303	011266	102402		BVS	ASLB4	
2304	011270	001401		BEQ	ASLB4	
2305	011272	100401		BMI	.+4	
2306	011274	104000	ASLB4:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2307						
2308	011276	105322		DECB	(R2)+	;(R0)=077401=[0774][001], CC=0010
2309	011300	103002		BCC	DECB2	
2310	011302	102001		BVC	DECB2	
2311	011304	100001		BPL	.+4	
2312	011306	104000	DECB2:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2313						
2314	011310	105645		SBCB	-(R5)	;(R0)=077400, CC=0100
2315	011312	103402		BCC	SBCB4	
2316	011314	102401		BVS	SBCB4	
2317	011316	001401		BEQ	.+4	
2318	011320	104000	SBCB4:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2319						
2320	011322	105442		NEGB	-(R2)	;(R0)=10400, CC=1001
2321	011324	103002		BCC	NEGB4	
2322	011326	102401		BVS	NEGB4	
2323	011330	100401		BMI	.+4	
2324	011332	104000	NEGB4:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2325						
2326	011334	105725		TSTB	(R5)+	;(R0)=100400, CC=0100
2327	011336	103401		BCS	TSTB2	
2328	011340	001401		BEQ	.+4	
2329	011342	104000	TSTB2:	HLT		
2330						
2331	011344	105722		TSTB	(R2)+	;(R0)=100400, CC=1000
2332	011346	001401		BEQ	TSTB2A	
2333	011350	100401		BMI	.+4	
2334	011352	104000	TSTB2A:	HLT		
2335						
2336	011354	000261		SEC		
2337	011356	000342		SWAB	-(R2)	;(R0)=000201, CC=1000

```

2339 011362 100401          BMI      .+4
2340 011364 104000          SWAB4:  HLT
2341 011366 000277          SCC
2342 011370 105225          INCB      (R5)+          ;(R0)=000601=(0004)(201),CC=0000
2343 011372 103003          BCC      INCB2
2344 011374 102402          BVS      INCB2
2345 011376 001401          BEQ      INCB2
2346 011400 100001          BPL      .+4
2347 011402 104000          INCB2:  HLT
2348 011404 022227 000601      CMP      (R2)+, #000601  ;CHECK END RESULT
2349 011410 001401          BEQ      .+4
2350 011412 104000          HLT
2351 011414 020205          CMP      R2, R5          ;CHECK REGISTERS
2352 011416 001401          BEQ      .+4
2353 011420 104000          HLT
2354 *****
2355 ;*TEST 11 CHECK UNIARY WORD OPS USING ADDRESS MODES 3 & 5
2356 ;*****
2357 TST11:  MOVW      #11, @STSTNM          ;LOAD TEST NUMBER
2358 011422 112737 000011 001202  SCOPE
2359 011430 000004          BR      .+6          ;RESERVE 2 WORDS
2360 011432 000402          .WORD   0          ;1 FOR THE ADDRESS
2361 011434 000000          .WORD   0          ;AND 1 FOR DATA
2362 011436 000000          MOV      PC, R3
2363 011440 010703          SUB      #4, R3
2364 011442 162703 000004      CLR      (R3)          ;PRESET DATA
2365 011444 005013          MOV      R3, R0          ;R0 POINTS TO DATA WORD
2366 011446 010300          TST      -(R3)
2367 011450 005743          MOV      R0, (R3)
2368 011452 010013          MOV      R3, R4
2369 011454 010304          CCC
2370 011456 000257          TST      @ (R3)+          ;(R0)=000000, CC=0100
2371 011460 005733          BEQ      .+4
2372 011462 001401          HLT
2373 011464 104000          SEC
2374 011466 000261          ROR      @-(R3)          ;(R0)=100000, CC=1010
2375 011470 006053          BCS      ROR5
2376 011472 103402          BVC      ROR5
2377 011474 102001          BMI      .+4
2378 011476 100401          RORS:  HLT
2379 011500 104000          CCC
2380 011502 000257          ASR      @ (R4)+          ;(R0)=140000, CC=1010
2381 011504 006234          BVC      ASR3
2382 011506 102001          BMI      .+4
2383 011508 100401          ASR3:  HLT
2384 011510 104000          CLN
2385 011512 000250          ASL      @ (R3)+          ;(R0)=100000, CC=1001
2386 011514 006333          BCC      ASL3
2387 011516 103002          BVS      ASL3
2388 011518 102401

```

CEKCC.P11 03-MAR-78 13:13 T11 CHECK UNIARY WORD OPS USING ADDRESS MODES 3 & 5

```

011530 104000 ASL3: HLT
011532 000277 SCC
011534 005354 DEC @-(R4) ;(R0)=077777, CC=0010
011536 103003 BCC DECS
011538 102002 BVC DECS
011540 001401 BEQ DECS
011542 100001 BPL .+4
011544 104000 DEC5: HLT
011550 005453 NEG @-(R3) ;(R0)=100001, CC=1001
011552 103002 BCC NEG5
011554 102401 BVS NEG5
011556 100401 BMI .+4
011560 104000 NEG5: HLT
011562 000262 SEV
011564 005134 COM @-(R4)+ ;(R0)=077776, CC=0001
011566 103001 BCC COM3
011570 102001 BVC .+4
011572 104000 COM3: HLT
011574 005233 INC @-(R3)+ ;(R0)=077777, CC=0001
011576 103001 BCC INC3
011600 100001 BPL .+4
011602 104000 INC3: HLT
011604 005554 ADC @-(R4) ;(R0)=100000, CC=1010
011606 103402 BCS ADC5
011610 102001 BVC ADC5
011612 100401 BMI .+4
011614 104000 ADC5: HLT
011616 000257 CCC
011620 006134 ROL @-(R4)+ ;(R0)=000000, CC=0111
011622 103002 BCC ROL3
011624 102001 BVC ROL3
011626 001401 BEQ .+4
011630 104000 ROL3: HLT
011632 005253 INC @-(R3) ;(R0)=000001, CC=0001
011634 005654 SBC @-(R4) ;(R0)=000000, CC=0100
011636 103401 BCS SBC5
011640 001401 BEQ .+4
011642 104000 SBC5: HLT
*****
: *TEST 12 CHECK UNIARY BYTE OPS USING ADDRESS MODES 3 & 5
*****
TST12: MOVB #12, @R1 ;LOAD TEST NUMBER
SCOPE
BR .+10 ;RESERVE 3 WORDS
.WORD 0 ;1 FOR EVEN BYTE ADDRESS
.WORD 0 ;1 FOR ODD BYTE ADDRESS
.WORD 0 ;AND 1 FOR DATA
MOV PC, R2

```

2501	011670	005742	TST	-(R2)	; DATA WORD
2502	011672	010200	MOV	R2, R0	; R0 POINTS TO THE DATA WORD
2503	011674	005010	CLR	(R0)	; PRESET DATA
2504	011676	005742	TST	-(R2)	; BACK R2 UP TO
2505	011700	005742	TST	-(R2)	; EVEN BYTE ADDRESS WORD
2506	011702	010022	MOV	R0, (R2)+	; LOAD ADDRESS
2507	011704	005200	INC	R0	; ODD BYTE ADDRESS
2508	011706	010022	MOV	R0, (R2)+	; LOAD ODD BYTE ADDRESS
2509	011710	010200	MOV	R2, R0	; RESET R0
2510	011712	010200	MOV	R2, R0	
2511	011714	105152	COMB	2-(R2)	; (R0)=177400, CC=1001
2512	011716	103001	BCC	COMB5	
2513	011720	100401	BMI	.+4	
2514	011722	104000	HLT		
2515	011724	105752	TSTB	2-(R2)	; (R0)=177400, CC=0100
2516	011726	001401	BEG	.+4	
2517	011730	104000	HLT		
2518	011732	000262	SEV		
2519	011734	106255	ASRB	2-(R5)	; (R0)=177400, CC=1001
2520	011736	103002	BCC	ASRB5	
2521	011740	102401	BVS	ASRB5	
2522	011742	100401	BMI	.+4	
2523	011744	104000	HLT		
2524	011746	105232	INCB	2(R2)+	; (R0)=177401, CC=000
2525	011750	103001	BCC	INCB3	
2526	011752	100001	BPL	.+4	
2527	011754	104000	HLT		
2528	011756	000241	CLC		
2529	011760	106055	RORB	2-(R5)	; (R0)=177400, CC=0111
2530	011762	103003	BCC	RORB5	
2531	011764	102002	BVC	RORB5	
2532	011766	001001	BNE	RORB5	
2533	011770	100001	BPL	.+4	
2534	011772	104000	HLT		
2535	011774	106332	ASLB	2(R2)+	; (R0)=177000, CC=1001
2536	011776	103002	BCC	ASLB3	
2537	012000	102401	BVS	ASLB3	
2538	012002	100401	BMI	.+4	
2539	012004	104000	HLT		
2540	012006	105552	ADCB	2-(R2)	; (R0)=177400, CC=1000
2541	012010	103401	BCS	ADCB5	
2542	012012	100401	BMI	.+4	
2543	012014	104000	HLT		
2544	012016	000277	SCC		
2545	012020	106135	ROLB	2(R5)+	; (R0)=177401, CC=0000
2546	012022	101402	BLOS	ROLB3	; BRANCH IF C OR Z IS SET
2547	012024	102401	BVS	ROLB3	
2548	012026	100001	BPL	.+4	
2549	012030	104000	HLT		

E06

```

2507 012034 100401      BMI      .+4
2508 012036 104000      HLT
2509
2510 012040 000261      SEC
2511 012042 105635      SBCB      2(R5)+      ;(R0)=000377, CC=0100
2512 012044 103401      BCS      SBCB3
2513 012046 001401      BEQ      .+4
2514 012050 104000      SBCB3:  HLT
2515
2516 012052 105432      NEGB     2(R2)+      ;(R0)=000001
2517 012054 105352      DECB     2-(R2)      ;(R0)=000000, CC=0101
2518 012056 103001      BCC
2519 012060 001401      BEQ      .+4
2520 012062 104000      DECB5:  HLT
2521
2522  ;*****
2523  ;*TEST 13 CHECK UNIARY WORD OPS USING ADDRESS MODE 6 (PC)
2524  ;*****
2525  ;ST13:  MOV      #13,2#STSTNM      ;LOAD TEST NUMBER
2526
2527 012064 112737 000013 001202  UWM6:  MOV      PC,RO      ;PRESET DATA = 0
2528 012072 000004      SCOPE      0      ;RESERVED FOR DATA
2529 012074 005027      CLR      (PC)+
2530 012076 000000      .WORD
2531 012100 010700      MOV      PC,RO      ;RO POINTS TO DATA WORD
2532 012102 024040      CMP      -(RO),-(RO)
2533 012104 000277      SCC
2534 012106 006167 177764      ROL      UWM6      ;(R0)=000001,CC=0000
2535 012112 103403      BCS      ROL6
2536 012114 102402      BVS      ROL6
2537 012116 001401      BEQ      ROL6
2538 012120 100001      BPL      .+4
2539 012122 104000      ROL6:  HLT
2540
2541 012124 005167 177746      COM      UWM6      ;(R0)=177776, CC=1001
2542 012130 103002      BCC      COM6
2543 012132 102401      BVS      COM6
2544 012134 100401      BMI      .+4
2545 012136 104000      HLT
2546 012140 006267 177732      COM6:  ASR      UWM6      ;(R0)=177777, CC=1010
2547 012144 103402      BCS      ASR6
2548 012146 102001      BVC      ASR6
2549 012150 100401      BMI      .+4
2550 012152 104000      ASR6:  HLT
2551
2552 012154 000277      SCC
2553 012156 005467 177714      NEG      UWM6      ;(R0)=000001, CC=0001
2554 012162 103003      BCC      NEG6
2555 012164 102402      BVS      NEG6
2556 012166 001401      BEQ      NEG6
2557 012170 100001      BPL      .+4
2558 012172 104000      NEG6:  HLT
2559
2560 012174 000277      SCC
2561 012176 006067 177674      ROR      UWM6      ;(R0)=100000, CC=1001
2562 012202 103003      BCC      ROR6
2563 012204 102402      BVS      ROR6
2564 012206 001401      BEQ      ROR6

```

F06

2563	012212	104000		ROR6:	HLT		
2564							
2565	012214	005667	177656		SBC	UWM6	;(RO)=077777, CC=0010
2566	012220	103402			BCS	SBC6	
2567	012222	102001			BVC	SBC6	
2568	012224	100001			BPL	.+4	
2569	012226	104000		SBC6:	HLT		
2570							
2571	012230	000242			CLV		
2572	012232	005267	177640		INC	UWM6	;(RO)=100000, CC=1011
2573	012236	103403			BCS	INC6	
2574	012240	102002			BVC	INC6	
2575	012242	001401			BEQ	INC6	
2576	012244	100401			BMI	.+4	
2577	012246	104000		INC6:	HLT		
2578							
2579	012250	006267	177622		ASR	UWM6	;(RO)=140000, CC=1010
2580	012254	000261			SEC		
2581	012256	006367	177614		ASL	UWM6	;(RO)=100000, CC=1001
2582	012262	103002			BCC	ASL6	
2583	012264	102401			BVS	ASL6	
2584	012266	100401			BMI	.+4	
2585	012270	104000		ASL6:	HLT		
2586							
2587	012272	005367	177600		DEC	UWM6	;(RO)=077777, CC=0011
2588	012276	103002			BCC	DEC6	
2589	012300	102001			BVC	DEC6	
2590	012302	100001			BPL	.+4	
2591	012304	104000		DEC6:	HLT		
2592							
2593	012306	005567	177564		ADC	UWM6	;(RO)=100000, CC=1010
2594	012312	103402			BCS	ADC6	
2595	012314	102001			BVC	ADC6	
2596	012316	100401			BMI	.+4	
2597	012320	104000		ADC6:	HLT		
2598	012322	000242			CLV		
2599	012324	000367	177546		SWAB	UWM6	
2600	012330	100401			BMI	.+4	
2601	012332	104000			HLT		
2602	012334	022710	000200		CMP	#200, (RO)	
2603	012340	001401			BEQ	.+4	
2604	012342	104000			HLT		
2605							
2606							
2607							
2608	012344	112737	000014	001202	TST14:	MOV B #14, @STSTNM	;LOAD TEST NUMBER
2609	012352	000004			SCOPE		
2610	012354	012700	012716		MOV	#UBM6, RO	
2611	012360	063700	001506		ADD	@FACTOR, RO	;RO POINTS TO ADDRESS OF DATA
2612	012364	005067	000326		CLR	UBM6	;CLEAR DATA
2613	012370	000277			SCC		
2614	012372	000244			CLZ		
2615	012374	105767	000316		TSTB	UBM6	
2616	012400	103403			BCS	TSTB6	
2617	012402	102402			BVS	TSTB6	



G06

CEQKCC PDP 11/70 CPU EXERCISOR  
 CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 52  
 T14 CHECK UNIARY BYTE OPS (EVEN/ODD) USING ADDRESS MODE 6 (PC)

SEQ 0071

2619	012406	100001		BPL	.+4	
2620	012410	104000		TSTB6:	HLT	
2621						
2622	012412	000257		CCC		
2623	012414	105767	000277	TSTB	UBM6+1	;TEST ODD BYTE
2624	012420	001401		BEQ	.+4	
2625	012422	104000		HLT		
2626						
2627	012424	105667	000266	SBCB	UBM6	;(R0)=000000, CC=0100
2628	012430	103402		BCS	SBCB6	
2629	012432	102401		BVS	SBCB6	
2630	012434	001401		BEQ	.+4	
2631	012436	104000		SBCB6:	HLT	
2632						
2633	012440	000261		1\$:	SEC	
2634	012442	105267	000250	INCB	UBM6	;LOOP UNTIL (R0)=077600, CC=1011
2635	012446	100403		BMI	2\$	
2636	012450	105567	000243	ADCB	UBM6+1	;INCB INST INCREMENTS EVEN BYTE
2637	012454	000771		BR	1\$	;ADCB INCREMENTS ODD BYTE
2638	012456	103001		2\$:	BCC	INCB6
2639	012460	102401		BVS	.+4	
2640	012462	104000		INCB6:	HLT	
2641						
2642	012464	106367	000226	ASLB	UBM6	;(R0)=077400, CC=0111
2643	012470	103003		BCC	ASLB6	
2644	012472	102002		BVC	ASLB6	
2645	012474	001001		BNE	ASLB6	
2646	012476	100001		BPL	.+4	
2647	012500	104000		ASLB6:	HLT	
2648						

2650	012504	105567	000207		ADCB	UBM6+1	;(RO)=100000, CC=1010
2651	012510	103402			BCS	ADCB6	
2652	012512	102001			BVC	ADCB6	
2653	012514	100401			BMI	.+4	
2654	012516	104000		ADCB6:	HLT		
2655							
2656	012520	000261			SEC		
2657	012522	106067	000171		RORB	UBM6+1	;(RO)=140000, CC=1010
2658	012526	103402			BCS	RORB6	
2659	012530	102001			BVC	RORB6	
2660	012532	100401			BMI	.+4	
2661	012534	104000		RORB6:	HLT		
2662							
2663	012536	105167	000154		COMB	UBM6	;(RO)=140377 CC=1001
2664	012542	103002			BCC	COMB6	
2665	012544	102401			BVS	COMB6	
2666	012546	100401			BMI	.+4	
2667	012550	104000		COMB6:	HLT		
2668							
2669	012552	000262			SEV		
2670	012554	105467	000137		NEGB	UBM6+1	;(RO)=040377, CC=0001
2671	012560	103002			BCC	NEGB6	
2672	012562	102401			BVS	NEGB6	
2673	012564	100001			BPL	.+4	
2674	012566	104000		NEGB6:	HLT		
2675							
2676	012570	106167	000123		ROLB	UBM6+1	;(RO)=100777, CC=1010
2677	012574	103402			BCS	ROLB6	
2678	012576	102001			BVC	ROLB6	
2679	012600	100401			BMI	.+4	
2680	012602	104000		ROLB6:	HLT		
2681							
2682	012604	106267	000106		ASRB	UBM6	;(RO)=100777, CC=1001
2683	012610	103002			BCC	ASRB6	
2684	012612	102401			BVS	ASRB6	
2685	012614	100401			BMI	.+4	
2686	012616	104000		ASRB6:	HLT		
2687							
2688	012620	105267	000072		INCB	UBM6	;(RO)=100400, CC=0101
2689	012624	103002			BCC	INCB6A	
2690	012626	102401			BVS	INCB6A	
2691	012630	001401			BEQ	.+4	
2692	012632	104000		INCB6A:	HLT		
2693							
2694	012634	105367	000057		DECB	UBM6+1	;(RO)=100000, CC=1001
2695	012640	103003			BCC	DECB6A	
2696	012642	102402			BVS	DECB6A	
2697	012644	001401			BEQ	DECB6A	
2698	012646	100401			BMI	.+4	
2699	012650	104000		DECB6A:	HLT		
2700							
2701	012652	000367	000040		SWAB	UBM6	;(RO)=000200, CC=1000
2702	012656	103401			BCS	SWAB6	
2703	012660	100401			BMI	.+4	
2704	012662	104000		SWAB6:	HLT		

```

2706 012664 106167 000026          ROLB  UBM6          ;(RO)=000000, CC=0111
2707 012670 103002          BCC   ROLB6A
2708 012672 102001          BVC   ROLB6A
2709 012674 001401          BEQ   .+4
2710 012676 104000          ROLB6A: HLT
2711
2712 012700 005767 000012          TST   UBM6          ;(RO)=000000, CC=0100
2713 012704 103402          BCS   TEST6
2714 012706 102401          BVS   TEST6
2715 012710 001401          BEQ   .+4
2716 012712 104000          TEST6: HLT
2717
2718 012714 000401          BR    .+4          ;RESERVE A WORD
2719 012716 000000          UBM6: .WORD 0      ;WORD RESERVED FOR DATA
2720 012720 000004          REL1: SCOPE
2721 012722 010702          MOV   PC,R2
2722 012724 062702 000012          ADD   #12,R2
2723 012730 012707 034242          MOV   #RELOC,PC    ;GO RELOCATE PROGRAM CODE
2724 012734 000000          REL11: .WORD 0
2725          ;11111111111111 LAST ADDRESS OF CODE TO BE RELOCATED 1111111111
2726
2727          ;*****
2728          ;*TEST 15 CHECK UNIARY WORD OPS USING ADDRESS MODE 7
2729          ;*****
2730 012736 112737 000015 001202  TST15: MOVB  #15,#STSTNM ;LOAD TEST NUMBER
2731 012744 012767 000001 166344  MOV   #1,$TIMES    ;;DO 1 ITERATION
2732 012752 000004          SCOPE
2733
2734          .SBTTL START OF SECTION 2
2735          :2222222222222222 FIRST ADDRESS TO BE RELOCATED 222222222
2736 012754 010700          REL2: MOV   PC,RO    ;GET PC
2737 012756 005740          TST   -(RO)       ;RO CONTAINS THE ADDRESS OF REL2
2738 012760 010037 001512          MOV   RO,#FRSTAD  ;SAVE
2739 012764 010700          MOV   PC,RO       ;GET CURRENT PC
2740 012766 162700 012766          SUB   #,RO        ;SUBTRACT RELOCATION FACTOR
2741 012772 010037 001506          MOV   RO,#FACTOR  ;SAVE RELOCATION FACTOR
2742 012776 010737 001212          MOV   PC,#$LPERR  ;SET LOOP ADDRESS
2743 013002 062737 000030 001212  ADD   #30,#$LPERR ;ADJUST
2744 013010 013737 001212 001210  MOV   #,$LPERR,#$LPADR
2745 013016 105737 001502          TSTB  #NEXEC      ;BR IF TEST CODE TO BE EXECUTED
2746 013022 001402          BEQ   .+6
2747 013024 000167 004060          TMP   REL2
2748 013030 000403          BR    UW7         ;RESERVE 3 WORDS FOR ADDRESSES & DATA
2749 013032 000000          UW7: .WORD 0      ;CONTAINS ADDRESS OF UW7
2750 013034 000000          .WORD 0          ;CONTAINS DATA
2751 013036 000000          .WORD 0          ;CONTAINS ADDRESS OF UW7
2752
2753 013040 010700          UW7: MOV   PC,RO
2754 013042 005740          TST   -(RO)
2755 013044 005740          TST   -(RO)
2756 013046 005040          CLR  -(RO)       ;CLEAR TEST DATA
2757 013050 010002          MOV   RO,R2
2758 013052 010240          MOV   R2,-(RO)   ;SET UP ADDRESS
2759 013054 005720          TST   (RO)+      ;MOVE RO TO NEXT ADDRESS
2760 013056 005720          TST   (RO)+

```

# JOB

CEQKCC PDP 11/70 CPU EXERCISOR MACY11 30A(1052) 03-MAR-78 13:15 PAGE 55  
 CEQKCC.P11 03-MAR-78 13:13 START OF SECTION 2

SEQ 0074

2762	013062	010200		MOV	R2,R0	;SET R0 POINTING TO DATA
2763	013064	000277		SCC		
2764	013066	000244		CLZ		
2765	013070	005772	000002	TST	02(2)	;(R0)=000000, CC=0100
2766	013074	001401		BEQ	.+4	
2767	013076	104000		HLT		
2768						
2769	013100	000277		SCC		
2770	013102	005672	177776	SBC	0-2(2)	;(R0)=177777, CC=1001
2771	013106	103002		BCC	SBC7	
2772	013110	102401		BVS	SBC7	
2773	013112	100401		BMI	.+4	
2774	013114	104000		HLT		
2775				SBC7:		
2776	013116	000277		SCC		
2777	013120	000241		CLC		
2778	013122	006372	000002	ASL	02(2)	;(R0)=177776, CC=1001
2779	013126	103002		BCC	ASL7	
2780	013130	102401		BVS	ASL7	
2781	013132	100401		BMI	.+4	
2782	013134	104000		HLT		
2783				ASL7:		
2784	013136	000257		CCC		
2785	013140	005372	000002	DEC	02(2)	;(R0)=177775, CC=1000
2786	013144	103402		BCC	DEC7	
2787	013146	102401		BVS	DEC7	
2788	013150	100401		BMI	.+4	
2789	013152	104000		HLT		
2790				DEC7:		
2791	013154	000262		SEV		
2792	013156	006272	177776	ASR	0-2(2)	;(R0)=177776, CC=1001
2793	013162	103002		BCC	ASR7	
2794	013164	102401		BVS	ASR7	
2795	013166	100401		BMI	.+4	
2796	013170	104000		HLT		
2797				ASR7:		
2798	013172	000241		CLC		
2799	013174	000262		SEV		
2800	013176	006072	177776	ROR	0-2(2)	;(R0)=077777, CC=0000
2801	013202	101402		BLOS	ROR7	;BRANCH IF C OR Z IS SET
2802	013204	102401		BVS	ROR7	
2803	013206	100001		BPL	.+4	
2804	013210	104000		HLT		
2805				ROR7:		
2806	013212	000262		SEV		
2807	013214	005472	000002	NEG	02(2)	;(R0)=100001, CC=1001
2808	013220	103002		BCC	NEG7	
2809	013222	102401		BVS	NEG7	
2810	013224	100401		BMI	.+4	
2811	013226	104000		HLT		
2812				NEG7:		
2813	013230	000250		CLN		
2814	013232	000372	177776	SWAB	0-2(2)	;(R0)=000600, CC=1000
2815	013236	103401		BCC	SWAB7	
2816	013240	100401		BMI	.+4	

```

2818
2819 013244 000262 SEV
2820 013246 005172 000002 COM 2(2) ;(RO)=177177, CC=1001
2821 013252 103002 BCC COM7
2822 013254 102401 BVS COM7
2823 013256 100401 BMI .+4
2824 013260 104000 COM7: HLT
2825
2826 013262 000372 000002 SWAB 2(2) ;(RO)=077776, CC=1000
2827 013266 100401 BMI .+4
2828 013270 104000 HLT
2829
2830 013272 000277 SCC
2831 013274 005572 177776 ADC 2-2(2) ;(RO)=077777, CC=0000
2832 013300 103402 BCS ADC7
2833 013302 102401 BVS ADC7
2834 013304 100001 BPL .+4
2835 013306 104000 ADC7: HLT
2836
2837 013310 005272 000002 INC 2(2) ;(RO)=100000, CC=1010
2838 013314 102001 BVC INC7
2839 013316 100401 BMI .+4
2840 013320 104000 INC7: HLT
2841
2842 013322 000257 CCC
2843 013324 006172 177776 ROL 2-2(2) ;(RO)=000000, CC=0111
2844 013330 103002 BCC ROL7
2845 013332 102001 BVC ROL7
2846 013334 001401 BEQ .+4
2847 013336 104000 ROL7: HLT
2848
2849 ;*****
2850 ;*TEST 16 CHECK UNIARY BYTE OPS USING ADDRESS MODE 7
2851 013340 112737 000016 001202 †ST16: MOVB #16,2*STSTNM ;LOAD TEST NUMBER
2852 013346 000004 SCOPE
2853 013350 012700 013034 MOV #UWM7,RO
2854 013354 063700 001506 ADD 2*FACTOR,RO
2855 013360 010002 MOV RO,R2
2856 013362 010067 177450 MOV RO,UWM7+2
2857 013366 005720 TST (RO)+
2858 013370 005210 INC (RO) ;WORD FOLLOWING UWM7 CONTAINS ADDRESS
2859 013372 005740 TST -(RO) ;OF ODD BYTE, RO POINTS TO DATA WORD
2860 013374 005010 CLR (RO) ;PRESET DATA
2861 013376 010067 177430 MOV RO,UWM7-2
2862 ;NOTE: 2(2) REFERENCES THE ODD BYTE, AND 2-2(2) REFERENCES THE EVEN BYTE.
2863
2864 013402 000263 +SEC!SEV ;SET C AND V
2865 013404 105672 000002 SBCB 2(2) ;(RO)=177400, CC=1001
2866 013410 103003 BCC SBCB7
2867 013412 102402 BVS SBCB7
2868 013414 001401 BEQ SBCB7
2869 013416 100401 BMI .+4
2870 013420 104000 SBCB7: HLT
2871
2872 013422 000277 SCC ;SET CONDITION CODES

```

L06

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 57  
T16 CHECK UNIARY BYTE OPS USING ADDRESS MODE 7

SEQ 0076

2874	013430	103403		BCS	ADCB7	
2875	013432	102402		BVS	ADCB7	
2876	013434	001401		BEQ	ADCB7	
2877	013436	100001		BPL	.+4	
2878	013440	104000		ADCB7:	HLT	
2879						
2880	013442	105172	177776	COMB	2-2(2)	;(RO)=177776, CC=1001
2881	013446	103002		BCC	COMB7	
2882	013450	102401		BVS	COMB7	
2883	013452	100401		BMI	.+4	
2884	013454	104000		COMB7:	HLT	
2885						
2886	013456	000241		CLC		;CLEAR CARRY
2887	013460	106072	000002	RORB	2(2)	;(RO)=077776, CC=0011
2888	013464	103002		BCC	RORB7	
2889	013466	102001		BVC	RORB7	
2890	013470	100001		BPL	.+4	
2891	013472	104000		RORB7:	HLT	
2892						
2893	013474	105272	000002	INCB	2(2)	;(RO)=100376, CC=1011
2894	013500	103002		BCC	INCB7	
2895	013502	102001		BVC	INCB7	
2896	013504	100401		BMI	.+4	
2897	013506	104000		INCB7:	HLT	
2898						
2899	013510	105372	177776	DECB	2-2(2)	;(RO)=100375, CC=1001
2900	013514	103002		BCC	DECB7	
2901	013516	102401		BVS	DECB7	
2902	013520	100401		BMI	.+4	
2903	013522	104000		DECB7:	HLT	
2904						
2905	013524	106372	000002	ASLB	2(2)	;(RO)=000375, CC=0111
2906	013530	103002		BCC	ASLB7	
2907	013532	102001		BVC	ASLB7	
2908	013534	001401		BEQ	.+4	
2909	013536	104000		ASLB7:	HLT	
2910						
2911	013540	000241		CLC		;CLEAR CARRY
2912	013542	106272	177776	ASRB	2-2(2)	;(RO)=000376, CC=1001
2913	013546	103002		BCC	ASRB7	
2914	013550	102401		BVS	ASRB7	
2915	013552	100401		BMI	.+4	
2916	013554	104000		ASRB7:	HLT	
2917						
2918	013556	105472	000002	NEGB	2(2)	;(RO)=000376, CC=0100
2919	013562	103402		BCS	NEGB7	
2920	013564	102401		BVS	NEGB7	
2921	013566	001401		BEQ	.+4	
2922	013570	104000		NEGB7:	HLT	
2923						
2924	013572	000262		SEV		
2925	013574	106172	177776	ROLB	2-2(2)	;(RO)=00374, CC=1001
2926	013600	103002		BCC	ROLB7	
2927	013602	102401		BVS	ROLB7	
2928	013604	100401		BMI	.+4	

M06

CEOKCC PDP 11/70 CPU EXERCISOR  
CEOKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 58  
T16 CHECK UNIARY BYTE OPS USING ADDRESS MODE 7

SEQ 0077

```

2930
2931 013610 105272 177776          INCB      R2-R(2)          ;(R0)=000375, CC=1001
2932 013614 105272 177776          INCB      R2-R(2)          ;(R0)=000376, CC=1001
2933 013620 105572 177776          ADCB      R2-R(2)          ;(R0)=000377, CC=1000
2934 013624 105172 177776          COMB      R2-R(2)          ;(R0)=000000, CC=0100
2935 013630 001401                    BEQ      .+4
2936 013632 104000                    HLT
2937
2938 ;*****
2939 ;TEST 17 CHECK BINARY OPS USING ADDRESS MODE 0
2940 ;*****
2941 ST17: 013634 112737 000017 001202  MOVB      #17,R#STSTNM      ;LOAD TEST NUMBER
2942 013642 000004                    SCOPE
2943 013644 000277                    SCC                      ;SET CONDITION CODES
2944 013646 010700                    MOV      PC,R0             ;RO=PC, CC=X001
2945 013650 103002                    BCC      MOV0
2946 013652 102401                    BVS      MOV0
2947 013654 001001                    BNE      .+4
2948 013656 104000                    MOVD:   HLT
2949
2950 013660 010002                    MOV      R0,R2            ;R2=R0
2951 013662 000262                    SEV                      ;SET V
2952 013664 160002                    SUB      R0,R2            ;R2=000000, CC=0100
2953 013666 103402                    BCS      SUB0
2954 013670 102401                    BVS      SUB0
2955 013672 001401                    BEQ      .+4
2956 013674 104000                    SUB0:   HLT
2957
2958 013676 000244                    CLZ
2959 013700 010203                    MOV      R2,R3            ;R2=R3=000000, CC=0100
2960 013702 103401                    BCS      MOV0A
2961 013704 001401                    BEQ      .+4
2962 013706 104000                    MOV0A:  HLT
2963
2964 013710 000257                    CCC
2965 013712 000272                    +SEV!SEN                 ;SET V & N
2966 013714 020203                    CMP      R2,R3            ;R2=R3=000000, CC=0100
2967 013716 103403                    BCS      CMPO
2968 013720 102402                    BVS      CMPO
2969 013722 001001                    BNE      CMPO
2970 013724 100001                    BPL      .+4
2971 013726 104000                    CMPO:   HLT
2972
2973 013730 010002                    MOV      R0,R2            ;R0=R2
2974 013732 010203                    MOV      R2,R3            ;R0=R2=R3
2975 013734 060203                    ADD      R2,R3            ;R3=2*R0
2976 013736 006302                    ASL      R2,R3            ;R2=2*R0
2977 013740 020203                    CMP      R2,R3            ;R2=R3=2*R0
2978 013742 001401                    BEQ      .+4
2979 013744 104000                    HLT
2980 ;ERROR! CHECK ADD INSTRUCTION
2981 ;THE FOLLOWING SUBTEST SHIFTS A BIT THROUGH R2 AND R5 AND DOES A
2982 ;BIT TEST (BIT) USING R2 AND R5.
2983 013746 005002                    CLR      R2
2984 013750 005202                    INC      R2
2985 013752 000402                    BR      R2

```

2986	013756	100407		BMI	48	
2987	013760	010205	28:	MOV	R2,R5	
2988	013762	000277		SCC		
2989	013764	030205		BIT	R2,R5	;R2=R5
2990	013766	103002		BCC	38	
2991	013770	102401		BVS	38	
2992	013772	001370		BNE	18	
2993	013774	104000	38:	HLT		
2994	013776	010205	48:	MOV	R2,R5	
2995	014000	000257		CCC		
2996	014002	030205		BIT	R2,R5	
2997	014004	100401		BMI	.+4	
2998	014006	104000		HLT		
2999						
3000	014010	005002		CLR	R2	
3001	014012	000277		SCC		
3002	014014	050002		BIS	R0,R2	
3003	014016	103002		BCC	BISO	
3004	014020	102401		BVS	BISO	
3005	014022	001001		BNE	.+4	
3006	014024	104000	BISO:	HLT		
3007						
3008	014026	010003		MOV	R0,R3	
3009	014030	000277		SCC		
3010	014032	000244		CLZ		
3011	014034	040003		BIC	R0,R3	
3012	014036	103003		BCC	BICO	
3013	014040	102402		BVS	BICO	
3014	014042	001001		BNE	BICO	
3015	014044	100001		BPL	.+4	
3016	014046	104000	BICO:	HLT		
3017						
3018	014050	010004		MOV	R0,R4	
3019	014052	005104		COM	R4	
3020	014054	040004		BIC	R0,R4	
3021	014056	005104		COM	R4	
3022	014060	020004		CMP	R0,R4	
3023	014062	001401		BEQ	.+4	
3024	014064	104000		HLT		
3025						
3026	014066	010004		MOV	R0,R4	
3027	014070	005104		COM	R4	
3028	014072	010403		MOV	R4,R3	
3029	014074	050003		BIS	R0,R3	
3030	014076	103001		BCC	BISOR	
3031	014100	100401		BMI	.+4	
3032	014102	104000	BISOR:	HLT		
3033	014104	005203		INC	R3	
3034	014106	001401		BEQ	.+4	
3035	014110	104000		HLT		
3036	014112	010304		MOV	R3,R4	
3037	014114	005103		COM	R3	:R3=R4=0
3038	014116	000261		SEC		:R3=177777
3039	014120	006004		ROR	R4	:SET C
3040	014122	060304		ADD	R3,R4	:R4=100000
						:R3=177777,R4=077777,CC=0011



```

3042 014126 102002          BVC      ADD0
3043 014130 001401          BEQ      ADD0
3044 014132 100001          BPL      .+4
3045 014134 104000          ADD0:   HLT
3046 014136 010700          MOV      PC,R0
3047 014140 022020          CMP      (R0)+,(R0)+
3048 014142 020007          CMP      R0,PC
3049 014144 001401          BEQ      .+4
3050 014146 104000          HLT
3051
3052 014150 010700          MOV      PC,R0
3053 014152 062700 000010        ADD      #10,R0
3054 014156 010002          MOV      R0,R2
3055 014160 020700          CMP      PC,R0
3056 014162 001002          BNE      CMPOA
3057 014164 020200          CMP      R2,R0
3058 014166 001401          BEQ      .+4
3059 014170 104000          HLT
3060
3061
3062
3063 014172 112737 000020 001202  *TEST 20 CHECK BINARY OPS USING ADDRESS MODE 1
3064 014200 000004          *ST20:  MOVWB #20,2*STSTNM ;LOAD TEST NUMBER
3065 014202 000402          SCOPE
3066 014204 000000          BR      .+6 ;RESERVE TWO WORDS
3067 014206 000000          .WORD 0 ;RESERVED FOR SOURCE DATA
3068 014210 010704          .WORD 0 ;RESERVED FOR DESTINATION DATA
3069 014212 005744          MOV      PC,R4
3070 014214 005044          TST     -(R4) ;R4 POINTS TO DESTINATION DATA
3071 014216 010403          CLR     -(R4)
3072 014220 005043          MOV      R4,R3 ;R3 POINTS TO SOURCE DATA
3073 014222 005113          CLR     -(R3)
3074 014224 005214          COM     (R3) ;(R3)=177777
3075 014226 000262          INC     (R4) ;(R4)=000001
3076 014230 061314          SEV
3077 014232 103002          ADD     (R3),(R4) ;(R3)=177777,(R4)=000000, CC=0101
3078 014234 102401          BCC     ADD1
3079 014236 001401          ADD1:  ADD1   ADD1
3080 014240 104000          BEQ     .+4
3081
3082
3083 014242 000277          SCC
3084 014244 000250          CLN
3085 014246 021314          CMP     (R3),(R4) ;(R3)=177777,(R4)=000000, CC=1000
3086 014250 103403          BCS     CMP1
3087 014252 102402          BVS     CMP1
3088 014254 001401          BEQ     CMP1
3089 014256 100401          BMI     .+4
3090 014260 104000          CMP1:  HLT
3091
3092 014262 000277          SCC
3093 014264 000244          CLZ
3094 014266 031314          BIT     (R3),(R4) ;(R3)=177777,(R4)=000000, CC=0101
3095 014270 103002          BITT1
3096 014272 102401          BVS     BITT1

```

```

3098 014276 104000      BITT1: HLT
3099
3100 014300 000277      SCC
3101 014302 000245      +CLC!CLZ
3102 014304 005114      COM      (R4)      ;(R4)=177777
3103 014306 161314      SUB      (R3),(R4) ;(R3)=177777,(R4)=000000,CC=0100
3104 014310 103402      BCS      SUB1
3105 014312 102401      BVS      SUB1
3106 014314 001401      BEQ      .+4
3107 014316 104000      SUB1: HLT
3108
3109 014320 105013      CLR      (R3)      ;(R3)=177400
3110 014322 000313      SWAB    (R3)      ;(R3)=000377
3111 014324 000270      SEN
3112 014326 011314      MOV      (R3),(R4) ;(R3)=(R4)=000377
3113 014328 100001      BPL      .+4
3114 014330 104000      HLT
3115 014332 000314      SWAB    (R4)      ;(R3)=000377,(R4)=177400
3116 014334 000263      +SEC!SEV ;SET C & V
3117 014336 051314      BIS      (R3),(R4) ;(R3)=000377,(R4)=177777,CC=1001
3118 014338 103002      BCC      BIS1
3119 014340 102401      BVS      BIS1
3120 014342 100401      BMI      .+4
3121 014344 104000      BIS1: HLT
3122
3123 014350 041314      BIC      (R3),(R4) ;(R3)=000377,(R4)=177400,CC=1001
3124 014352 103002      BCC      BIC1
3125 014354 102401      BVS      BIC1
3126 014356 100401      BMI      .+4
3127 014358 104000      BIC1: HLT
3128
3129 014364 000262      SEV
3130 014366 021314      CMP      (R3),(R4) ;SET V
3131 014370 103003      BCC      (R3)=000377,(R4)=177400,CC=0001
3132 014372 102402      BVS      CMP1A
3133 014374 001401      BEQ      CMP1A
3134 014376 100001      BPL      CMP1A
3135 014400 104000      CMP1A: HLT
3136
3137 014402 005013      CLR      (R3)      ;(R3)=000000
3138 014404 000261      SEC
3139 014406 006013      ROR      (R3)      ;(R3)=100000
3140 014410 011314      MOV      (R3),(R4) ;(R3)=(R4)=100000
3141 014412 005114      COM      (R4)      ;(R4)=077777
3142 014414 161314      SUB      (R3),(R4) ;(R3)=100000,(R4)=177777,CC=1011
3143 014416 103002      BCC      SUB1A
3144 014420 102001      BVC      SUB1A
3145 014422 100401      BMI      .+4
3146 014424 104000      SUB1A: HLT
3147
3148 014426 000277      SCC
3149 014430 161314      SUB      (R3),(R4) ;(R3)=100000,(R4)=077777,CC=0000
3150 014432 101402      BLOS    SUB1B    ;BRANCH IF C OR Z IS SET
3151 014434 102401      BVS      SUB1B
3152 014436 100001      BPL      .+4

```

```

3154      014442 011314      MOV      (R3), (R4)      ;(R3)=100000, (R4)=100000, CC=1000
3155      014444 001401      BEQ      MOV1
3156      014446 100401      BMI      .+4
3157      014448 104000      MOV1:   HLT
3158      014452 061314      ADD      (R3), (R4)      ;(R3)=100000, (R4)=000000, CC=0111
3159      014454 103003      BCC      ADD1A
3160      014456 102002      BVC      ADD1A
3161      014460 001001      BNE      ADD1A
3162      014462 100001      BPL      .+4
3163      014464 104000      ADD1A:  HLT
3164      014466 005113      COM      (R3)            ;(R3)=077777
3165      014470 011314      MOV      (R3), (R4)      ;(R4)=077777
3166      014472 061314      ADD      (R3), (R4)      ;(R3)=077777, (R4)=177776, CC=1010
3167      014474 103402      BCS      ADD1B
3168      014476 102001      BVC      ADD1B
3169      014500 100401      BMI      .+4
3170      014502 104000      ADD1B:  HLT
3171      014504 062714 000002      ADD      #2, (R4)
3172      014510 005714      TST      (R4)            ;CHECK FINAL RESULT
3173      014512 001401      BEQ      .+4
3174      014514 104000      HLT
3175      *****
3176      *TEST 21 CHECK BINARY BYTE OPS USING ADDRESS MODE 1
3177      *****
3178      *ST21: MOV8  #21, #STSTMM ;LOAD TEST NUMBER
3179      *****
3180      014516 112737 000021 001202      ST21:  MOV8  SCOPE
3181      014524 000004      BR      .+6
3182      014526 000402      .WORD  0
3183      014530 000000      .WORD  0
3184      014532 010705      MOV      PC, R5
3185      014534 005745      TST      -(R5)
3186      014536 005045      CLR      -(R5)            ;(R5)=000000
3187      014540 010502      MOV      R5, R2
3188      014542 005042      CLR      -(R2)            ;(R2)=000000
3189      014544 005202      INC      R2                ;R2 POINTS TO ODD BYTE
3190      014546 105112      COMB     (R2)              ;(R2)=177400
3191      014552 000277      SCC      (R2), (R5)        ;(R2)=177400, (R5)=000377, CC=1001
3192      014554 111215      MOV8     (R2), (R5)
3193      014556 103005      BCC      MOV81
3194      014560 102404      BVS      MOV81
3195      014562 001403      BEQ      MOV81
3196      014564 100002      BPL      MOV81
3197      014566 105215      INCB     (R5)            ;CHECK RESULT
3198      014570 001401      BEQ      .+4
3199      014572 104000      MOV81:  HLT
3200      014574 106312      ASLB     (R2)            ;SHIFT (R2) UNTIL
3201      014576 102376      BVC      -2              ;(R2)=000000
3202      014600 106012      RORB     (R2)            ;(R2)=10000C
3203      014602 105315      DECB     (R5)            ;(R5)=00377

```

```

3210 014606 000257      CCC
3211 014610 121512      CMPB      (R5) (R2)      ;(R5)=000177,(R2)=100000, CC=1010
3212 014612 102001      BVC      CMPB1
3213 014614 100401      BMI      .+4
3214 014616 104000      CMPB1:  HLT
3215
3216 014620 005003      CLR      R3
3217 014622 000261      SEC
3218 014624 006003      ROR      R3      ;R3=100000
3219 014626 050315      BIS      R3,(R5)  ;(R5)=100177
3220 014630 000273      +SEC!SEV!SEN  ;SET C V A & N
3221 014632 131215      BITB      (R2) (R5) ; (R2)=100000,(R5)=100177, CC=0101
3222 014634 103002      BCC      BITB1
3223 014636 102401      BVS      BITB1
3224 014640 001401      BEQ      .+4
3225 014642 104000      BITB1:  HLT
3226
3227 014644 151215      BISB      (R2) (R5) ; (R2)=100000,(R5)=100377, CC=1001
3228 014646 103001      BCC      BISB1
3229 014650 100401      BMI      .+4
3230 014652 104000      BISB1:  HLT
3231
3232 014654 141215      BICB      (R2) (R5) ; (R2)=100000,(R5)=100177, CC=0001
3233 014656 103002      BCC      BICB1
3234 014660 001401      BEQ      BICB1
3235 014662 100001      BPL      .+4
3236 014664 104000      BICB1:  HLT
3237
3238 014666 105112      COMB      (R2)      ; (R2)=077400,(R5)=100177
3239 014670 121215      CMPB      (R2) (R5)
3240 014672 001401      BEQ      .+4
3241 014674 104000      HLT
3242
3243 014676 141512      BICB      (R5) (R2) ; (R5)=100177,(R2)=000000, CC=0100
3244 014700 001002      BNE      BICB1A
3245 014702 105712      TSTB      (R2)
3246 014704 001401      BEQ      .+4
3247 014706 104000      BICB1A: HLT
3248
3249 014710 000402      BR      .+6      ;RESERVE TWO WORDS FOR DATA
3250 014712 000000      .WORD 0      ;SOURCE DATA
3251 014714 000000      .WORD 0      ;DEST DATA
3252 014716 010705      MOV      PC,R5
3253 014720 005745      TST      -(R5)
3254 014722 105045      CLRB      -(R5)      ;R5 POINTS TO DEST ODD BYTE
3255 014724 010504      MOV      R5,R4
3256 014726 105044      CLRB      -(R4)      ;R4 POINTS TO DEST EVEN BYTE
3257 014730 010403      MOV      R4,R3
3258 014732 105043      CLRB      -(R3)      ;R3 POINTS TO SOURCE ODD BYTE
3259 014734 010302      MOV      R3,R2
3260 014736 105042      CLRB      -(R2)      ;R2 POINTS TO SOURCE EVEN BYTE
3261
3262      ;COMMENTS ARE LEAST SIGNIFICANT 4 BITS OF BYTES POINTED TO BY R2,R3
3263      ;R4, AND R5 RESPECTIVELY AND THE REMAINING BITS ARE 0'S.
3264 014740 000261      SEC      ;SET CARRY

```

```

3266 014742 106112 ROLB (R2) ;0001,0000,0000,0000
3267 014744 111214 MOVB (R2),(R4) ;0001,0000,0001,0000
3268 014746 106112 ROLB (R2) ;0010,0000,0001,0000
3269 014750 111213 MOVB (R2),(R3) ;0010,0010,0001,0000
3270 014752 106112 ROLB (R2) ;0100,0010,0001,0000
3271 014754 111315 MOVB (R3),(R5) ;0100,0010,0001,0010
3272 014756 106112 ROLB (R2) ;1000,0010,0001,0010
3273 014760 106113 ROLB (R3) ;1000,0100,0001,0010
3274 014762 151215 BISB (R2),(R5) ;1000,0100,0001,1010
3275 014764 131512 BITB (R5),(R2) ;1000,0100,0001,1010
3276 014766 001426 BEQ BIN1
3277 014770 151314 BISB (R3),(R4) ;1000,0100,0101,1010
3278 014772 131413 BITB (R4),(R3) ;1000,0100,0101,1010
3279 014774 001423 BEQ BIN1
3280 014776 105213 INCB (R3) ;1000,0101,0101,1010
3281 015000 121314 CMPB (R3),(R4) ;1000,0101,0101,1010
3282 015002 001020 BNE BIN1
3283 015004 106113 ROLB (R3) ;1000,1010,0101,1010
3284 015006 121315 CMPB (R3),(R5) ;1000,1010,0101,1010
3285 015010 001015 BNE BIN1
3286 015012 106212 ASRB (R2) ;0100,1010,0101,1010
3287 015014 131214 BITB (R2),(R4) ;0100,1010,0101,1010
3288 015016 001412 BEQ BIN1
3289 015020 106015 RORB (R5) ;0100,1010,0101,0101
3290 015022 121415 CMPB (R4),(R5) ;0100,1010,0101,0101
3291 015024 001007 BNE BIN1
3292 015026 105314 DECB (R4) ;0100,1010,0100,0101
3293 015030 141214 BICB (R2),(R4) ;0100,1010,0000,0101
3294 015032 001004 BNE BIN1
3295 015034 111314 MOVB (R3),(R4) ;0100,1010,1010,0101
3296 015036 106213 ASRB (R3) ;0100,0101,1010,0101
3297 015040 141315 BICB (R3),(R5) ;0100,0101,1010,0101
3298 015042 001401 BEQ .+4
3299 015044 104000 HLT
3300 BIN1:
3301 ;*****
3302 ;*TEST 22 CHECK BINARY WORD OPS USING ADDRESS MODE 2 & 4
3303 ;*****
3303 015046 112737 000022 001202 †ST22: MOVB #22,2#STSTNM ;LOAD TEST NUMBER
3304 015054 000004 SCOPE
3305 015056 012704 014714 MOV #BICB1A+6,R4
3306 015062 012702 014712 MOV #BICB1A+4,R2
3307 015066 063702 001506 ADD 2#FACTOR,R2
3308 015072 063704 001506 ADD 2#FACTOR,R4
3309 015076 010405 MOV R4,R5 ;SET DESTINATION REGISTER
3310 015100 012715 000001 MOV #1,(R5)
3311 015104 012712 177777 MOV #-1,(R2)
3312 015110 000257 CCC
3313 015112 000262 SEV
3314 015114 062225 ADD (R2)+,(R5)+ ;(R2)=177777,(R5)=000000,CC=0101
3315 015116 103002 BCC ADD2
3316 015120 102401 BVS ADD2
3317 015122 001401 BEQ .+4
3318 015124 104000 ADD2: HLT
3319
3320 015126 000262 SEV ;SET V

```

G07

3322	015134	103002		BCC	CMP2		
3323	015136	102401		BVS	CMP2		
3324	015140	100401		BMI	.+4		
3325	015142	104000		HLT			
			CMP2:				
3327	015144	054225		BIS	-(R2), (R5)+	;(R2)=177777, (R5)=177777, CC=1001	
3328	015146	103001		BCC	BIS2		
3329	015150	100401		BMI	.+4		
3330	015152	104000		HLT			
			BIS2:				
3331	015154	000277		SCC			
3332	015156	000244		CLZ			
3333	015160	162245		SUB	(R2)+, -(R5)	;(R2)=177777, (R5)=000000, CC=0100	
3334	015162	103402		BCS	SUB2		
3335	015164	102401		BVS	SUB2		
3336	015166	001401		BEQ	.+4		
3337	015170	104000		HLT			
			SUB2:				
3339	015172	005442		NEG	-(R2)	;(R2)=000001	
3340	015174	005115		COM	(R5)	;(R5)=177777	
3341	015176	000277		SCC			
3342	015200	000250		CLN			
3343	015202	042225		BIC	(R2)+, (R5)+	;(R2)=000001, (R5)=177776, CC=1001	
3344	015204	103003		BCC	BIC2		
3345	015206	102402		BVS	BIC2		
3346	015210	001401		BEQ	BIC2		
3347	015212	100401		BMI	.+4		
3348	015214	104000		HLT			
			BIC2:				
3350	015216	012742	125252	MOV	#125252, -(R2)		
3351	015222	012245		MOV	(R2)+, -(R5)		
3352	015224	005125		COM	(R5)+	;(R5)=052525	
3353	015226	000262		SEV			
3354	015230	034245		BIT	-(R2), -(R5)	;(R2)=125252, (R5)=052525, CC=0101	
3355	015232	103002		BCC	BITT2		
3356	015234	102401		BVS	BITT2		
3357	015236	001401		BEQ	.+4		
3358	015240	104000		HLT			
			BITT2:				
3360	015242	000262		SEV			
3361	015244	052225		BIS	(R2)+, (R5)+	;(R2)=125252, (R5)=177777, CC=1001	
3362	015246	103002		BCC	BIS2A		
3363	015250	102401		BVS	BIS2A		
3364	015252	100401		BMI	.+4		
3365	015254	104000		HLT			
			BIS2A:				
3367	015256	042745	125252	BIC	#125252, -(R5)	;(R5)=052525	
3368	015262	005125		COM	(R5)+	;(R5)=125252	
3369	015264	024245		CMP	-(R2), -(R5)		
3370	015266	001401		BEQ	.+4		
3371	015270	104000		HLT			
3373	015272	005012		CLR	(R2)		
3374	015274	005122		COM	(R2)+	;(R2)=177777	
3375	015276	162742	000001	SUB	#1-(R2)	;(R2)=177776, CC=1000	
3376	015302	103402		BCS	SUB2A		

H07

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 66  
T22

CHECK BINARY WORD OPS USING ADDRESS MODE 2 & 4

SEQ 0085

```

3378 015306 100401
3379 015310 104000
3380 015312 010702
3381 015314 010205
3382 015316 124245
3383 015320 001401
3384 015322 104000
3385 015324 020237 001512
3386 015330 001372
3387
3388
3389
3390 015332 112737 000023 001202
3391 015340 000004
3392 015342 000402
3393 015344 000000
3394 015346 000000
3395 015350 010703
3396 015352 005743
3397
3398
3399 015354 010300
3400 015356 010002
3401 015360 005302
3402 015362 010604
3403 015364 010605
3404 015366 005745
3405
3406 015370 114046
3407 015372 020506
3408 015374 001021
3409 015376 020200
3410 015400 001017
3411 015402 122026
3412 015404 020406
3413 015406 001014
3414 015410 020003
3415 015412 001012
3416 015414 154640
3417 015416 020506
3418 015420 001007
3419 015422 020200
3420 015424 001005
3421 015426 142620
3422 015430 020406
3423 015432 001002
3424 015434 020003
3425 015436 001401
3426 015440 104000
3427 015442 010003
3428 015444 112743 000200
3429 015450 112743 000377
3430 015454 010304
3431 015456 112744 000177
3432 015462 112744 000000

```

```

SUB2A: BMI .+4
HLT
MOV PC,R2 ;GET CURRENT PC
MOV R2,R5 ;MOVE TO R5
15: CMPB -(R2),-(R5) ;COMPARE ALL PREVIOUS MEMORY ADDRESSES
BEQ .+4
HLT ;ERROR!
CMP R2,#FRSTAD ;CHECK FOR LOW LIMIT
BNE 15
;*****
; *TEST 23 CHECK BINARY BYTE OPS USING ADDRESS MODE 2 & 4
;*****
3423: MOVB #23,#STSTNM ;LOAD TEST NUMBER
SCOPE
BR .+6 ;RESERVE TWO WORDS
.WORD 0 ;SOURCE DATA
.WORD 0 ;DESTINATION DATA
MOV PC,R3
TST -(R3)

;FIRST CHECK AUTO INCREMENT/DECREMENT
MOV R3,R0
MOV R0,R2
DEC R2
MOV SP,R4
MOV SP,R5
TST -(R5)

MOVB -(R0),-(SP)
CMP R5,SP
BNE BINB
CMP R2,R0
BNE BINB
CMPB (R0)+,(SP)+
CMP R4,SP
BNE BINB
CMP R0,R3
BNE BINB
BISB -(SP),-(R0)
CMP R5,SP
BNE BINB
CMP R2,R0
BNE BINB
BICB (SP)+,(R0)+
CMP R4,SP
BNE BINB
CMP R0,R3
BEQ .+4
BINB: HLT
MOV R0,R3
MOVB #200,-(R3)
MOVB #377,-(R3) ;(R3)=100377
MOV R3,R4
MOVB #177,-(R4)
MOVB #0,-(R4) ;(R4)=077400

```

```

3434 015470 104000 HLT
3435 015472 152324 BISB (R3)+,(R4)+ ;(R3)=100377,(R4)=077777
3436 015474 100401 BMI .+4
3437 015476 104000 HLT
3438 015500 122324 CMPB (R3)+,(R4)+
3439 015502 103402 BCS CMPB2
3440 015504 102001 BVC CMPB2
3441 015506 100001 BPL .+4
3442 015510 104000 CMPB2: HLT
3443 015512 000261 SEC
3444 015514 134344 BITB -(R3),-(R4)
3445 015516 103002 BCC BITB2
3446 015520 102401 BVS BITB2
3447 015522 001401 BEQ .+4
3448 015524 104000 BITB2: HLT
3449 015526 000244 CLZ
3450 015530 144344 BICB -(R3),-(R4) ;(R3)=100377,(R4)=077400
3451 015532 001401 BEQ .+4
3452 015534 104000 HLT
*****
; *TEST 24 CHECK BINARY WORD OPS USING ADDRESS MODES 3 & 5
*****
3453 015536 112737 000024 001202 †ST24: MOV B #24,2*STSTNM ;LOAD TEST NUMBER
3454 015544 000004 SCOPE
3455 015546 000404 BR 2$ ;RESERVE SPACE FOR DATA AND ADDRESSES
3456 015550 000000 .WORD 0 ;CONTAINS ADDRESS OF SOURCE DATA
3457 015552 000000 .WORD 0 ;CONTAINS ADDRESS OF DEST DATA
3458 015554 000000 .WORD 0 ;CONTAINS SOURCE DATA
3459 015556 000000 .WORD 0 ;CONTAINS DEST DATA
3460 015560 010701 2$: MOV PC,R1
3461 015562 010100 MOV R1,R0 ;SET SCOPE PTR
3462 015564 024040 CMP -(R0),-(R0) ;ADJUST R0
3463 015566 010005 MOV R0,R5 ;R5 POINTS TO DEST DATA
3464 015570 024545 CMP -(R5),-(R5) ;SUB 4 FROM R5
3465 015572 010015 MOV R0,(R5) ;R5 POINTS TO ADDRESS OF DEST DATA
3466 015574 010502 MOV R5,R2
3467 015576 010004 MOV R0,R4 ;R4 POINTS TO DEST DATA
3468 015600 005740 TST -(R0)
3469 015602 010003 MOV R0,R3 ;R3 POINTS TO SOURCE DATA
3470 015604 010042 MOV R0,-(R2) ;R2 POINTS TO ADDRESS OF SOURCE DATA
3471 015606 005013 CLR (R3) ;PRESET SOURCE DATA
3472 015610 005014 CLR (R4) ;PRESET DEST DATA
3473 015612 000277 SCC
3474 015614 000244 CLZ
3475 015616 163235 SUB 2(R2)+,2(R5)+ ;(R3)=000000,(R4)=000000,CC=0100
3476 015620 103402 SUB3 SUB3
3477 015622 102401 BCS SUB3
3478 015624 001401 BVS SUB3
3479 015626 104000 BEQ .+4
3480 HLT
SUB3:

```



```

3490 015634 062755 000001      ADD      #1,2-(R5)      ;(R4)=000001
3491 015640 163235      SUB      2(R2)+,2(R5)+ ;(R3)=100000,(R4)=100001,CC=1011
3492 015642 103002      BCC     SUB3A
3493 015644 102001      BVC     SUB3A
3494 015646 100401      BMI     .+4
3495 015650 104000      SUB3A:  HLT
3496
3497 015652 005414      NEG     (R4)          ;(R4)=077777
3498 015654 035255      BIT     2-(R2),2-(R5) ;(R3)=100000,(R4)=077777
3499 015656 001401      BEQ     .+4
3500 015660 104000      HLT
3501 015662 023235      CMP     2(R2)+,2(R5)+
3502 015664 102401      BVS     .+4
3503 015666 104000      HLT
3504 015670 005152      COM     2-(R2)
3505 015672 000257      CCC
3506 015674 063255      ADD     2(R2)+,2-(R5)
3507 015676 102001      BVC     ADD3
3508 015700 100401      BMI     .+4
3509 015702 104000      ADD3:  HLT
3510 015704 000261      SEC
3511 015706 045235      BIC     2-(R2),2(R5)+ ;(R3)=077777,(R4)=100000
3512 015710 103001      BCC     BIC3
3513 015712 100401      BMI     .+4
3514 015714 104000      BIC3:  HLT
3515
3516 015716 005155      COM     2-(R5)        ;(R4)=077777
3517 015720 023235      CMP     2(R2)+,2(R5)+ ;(R3)=077777,(R4)=077777
3518 015722 001401      BEQ     .+4
3519 015724 104000      HLT
3520
3521      ;*****
3522      ;*TEST 25 CHECK BINARY BYTE OPS USING ADDRESS MODES 3 & 5
3523      ;*****
3524      ;ST25:  MOV#B #25,2#STSTNM ;LOAD TEST NUMBER
3525      SCOPE
3526      BR      15 ;RESERVE SPACE FOR ADDRESS AND DATA
3527      .WORD 0 ;CONTAINS ADDRESS OF SOURCE DATA (EVEN BYTE)
3528      .WORD 0 ;CONTAINS ADDRESS OF SOURCE DATA (ODD BYTE)
3529      .WORD 0 ;CONTAINS ADDRESS OF DEST DATA (EVEN BYTE)
3530      .WORD 0 ;CONTAINS ADDRESS OF DEST DATA (ODD BYTE)
3531      .WORD 0 ;CONTAINS SOURCE DATA
3532      .WORD 0 ;CONTAINS DEST DATA
3533
3534      15:  MOV     PC,R0
3535      CMP     -(R0),-(R0) ;R0=ADDRESS OF DEST DATA
3536      MOV     R0,R3 ;R3
3537      MOV     R3,R5 ;R5
3538      TST     -(R3) ;SUB 2 FROM R3
3539      MOV     R0,-(R3) ;R3 POINTS TO ADDRESS OF DEST DATA
3540      INC     (R3) ;ODD BYTE
3541      MOV     R0,-(R3) ;EVEN BYTE
3542      MOV     R3,R4
3543      TST     -(R0) ;R0=ADDRESS OF SOURCE DATA
3544      MOV     R0,-(R4) ;R4 POINTS TO ADDRESS OF SOURCE DATA
3545      INC     (R4) ;ODD BYTE

```

K07

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 69  
T25 CHECK BINARY BYTE OPS USING ADDRESS MODES 3 & 5

SEQ 0088

```

3546
3547 016006 000261 SEC ;SET CARRY
3548 016010 012734 177001 MOV #177001,@(R4)+
3549 016014 112734 000200 MOVB #200,@(R4)+ ;SOURCE DATA=100001
3550 016020 115433 MOVB @-(R4),@(R3)+
3551 016022 115433 MOVB @-(R4),@(R3)+ ;DEST DATA=000600
3552 016024 103401 BCS .+4
3553 016026 104000 HLT ;ERROR! MOV DOES AFFECT C BIT IN PSW
3554 016030 022715 000600 CMP #600,(R5) ;CHECK DEST DATA
3555 016034 001401 BEQ .+4
3556 016036 104000 HLT ;ERROR! INCORRECT RESULT
3557 016040 024343 CMP -(R3),-(R3) ;POINT R4 BACK TO EVEN BYTE
3558 016042 153433 BISB @(R4)+,@(R3)+
3559 016044 153433 BISB @(R4)+,@(R3)+ ;DEST DATA=100601
3560 016046 022715 100601 CMP #100601,(R5) ;CHECK RESULT
3561 016052 001401 BEQ .+4
3562 016054 104000 HLT ;ERROR! INCORRECT DEST DATA AFTER BISB
3563 016056 145453 BICB @-(R4),@-(R3)
3564 016060 145453 BICB @-(R4),@-(R3)
3565 016062 133433 BITB @(R4)+,@(R3)+
3566 016064 001002 BNE BITB3
3567 016066 135433 BITB @-(R4),@(R3)+
3568 016070 001001 BNE .+4
3569 016072 104000 BITB3: HLT
3570
3571 016074 123453 CMPB @(R4)+,@-(R3)
3572 016076 001002 BNE CMPB3
3573 016100 123453 CMPB @(R4)+,@-(R3)
3574 016102 001401 BEQ .+4
3575 016104 104000 CMPB3: HLT
3576
3577 ;*****
3578 ;*TEST 26 CHECK BINARY OPS USING ADDRESS MODE 6
3579 ;*****
3579 016106 112737 000026 001202 †ST26: MOVB #26,@#STSTNM ;LOAD TEST NUMBER
3580 016114 000004 SCOPE
3581 016116 000402 BR .+6 ;RESERVE TWO LOCATIONS
3582 016120 000000 SDATA: .WORD 0 ;RESERVED FOR SOURCE DATA
3583 016122 000000 DDATA: .WORD 0 ;RESERVED FOR DESTINATION DATA
3584
3585 016124 013702 001506 MOV @#FACTOR,R2 ;GET RELOCATION FACTOR AND USE AS AN
3586 016130 010205 MOV R2,R5 ;INDEX VALUE TO POINT TO DATA
3587 016132 005065 016122 CLR DDATA(5) ;PRESET DESTINATION DATA
3588 016136 012762 000001 016120 MOV #1,SDATA(2) ;THIS ROUTINE PUT A 1 BIT INTO EVERY
3589 016144 056265 016120 016122 1$: BIS SDATA(2),DDATA(5) ;OTHER BIT POSITION IN THE DEST-
3590 016152 006362 016120 ASL SDATA(2) ;INATION ADDRESS (52525)
3591 016156 006362 016120 ASL SDATA(2)
3592 016162 103370 BCC 1$
3593 016164 022765 052525 016122 CMP #52525,DDATA(5) ;CHECK RESULT
3594 016172 001401 BEQ .+4
3595 016174 104000 HLT ;ERROR! INCORRECT RESULT
3596 016176 012762 177777 016120 MOV #-1,SDATA(2)
3597 016204 046562 016122 016120 BIC DDATA(5),SDATA(2) ;SOURCE DATA=125252
3598 016212 036265 016120 016122 BIT SDATA(2),DDATA(5)
3599 016220 001401 BEQ .+4
3600 016222 104000 HLT ;ERROR! BIT INST FAILED

```

```

3602 016230 026265 016120 016122      CMP      SDATA(2),DDATA(5)
3603 016236 001401                      BEQ      .+4
3604                                     ;
3605 016240 104000                      HLT      ;ERROR! CMP INST FAILED
3606 016242 000257                      CCC
3607 016244 066265 016120 016122      ADD      SDATA(2),DDATA(5)
3608 016252 103002                      BCC      ADD6
3609 016254 102001                      BVC      ADD6
3610 016256 100001                      BPL      .+4
3611 016260 104000                      ADD6:   HLT
3612                                     ;
3613 016262 006362 016120 016122      ASL      SDATA(2) ;SDATA=52524
3614 016266 166265 016120 016122      SUB      SDATA(2),DDATA(5)
3615 016274 103401                      BCS      SUB6
3616 016276 001401                      BEQ      .+4
3617 016300 104000                      SUB6:   HLT
3618                                     ;
3619 016302 112700 000377 016120      MOVB     #377,R0 ;R0=177777 (MOVB %R EXTENDS SIGN)
3620 016306 010062 016120 016122      MOV      R0,SDATA(2)
3621 016312 012765 177777 016122      MOV      #-1,DDATA(5)
3622 016320 166500 016122 016122      SUB      DDATA(5),R0
3623 016324 001401                      BEQ      .+4
3624 016326 104000                      HLT
3625 016330 066265 016120 016122 1$:  ADD      SDATA(2),DDATA(5)
3626 016336 006362 016120 016122      ASL      SDATA(2)
3627 016342 005162 016120 016122      COM      SDATA(2)
3628 016346 036265 016120 016122      BIT      SDATA(2),DDATA(5)
3629 016354 001401                      BEQ      .+4
3630 016356 104000                      HLT
3631 016360 005162 016120 016122      COM      SDATA(2)
3632 016364 026265 016120 016122      CMP      SDATA(2),DDATA(5)
3633 016372 001401                      BEQ      .+4
3634 016374 104000                      HLT
3635 016376 026200 016120 016122      CMP      SDATA(2),R0
3636 016402 001352                      BNE     1$
3637                                     ;
3638                                     ;*****
3639                                     ;*TEST 27 CHECK BINARY BYTE OPS USING ADDRESS MODE 6
3640                                     ;*****
3640 016404 112737 000027 001202 1$T27: MOVB     #27,%STSTNM ;LOAD TEST NUMBER
3641 016412 000004                      SCOPE
3642                                     ;NOTE: SDATA(2) AND DDATA(4) REFERENCE EVEN BYTE OF SOURCE & DEST DATA
3643                                     ;AND SDATA(3), AND DDATA(5) REFERENCE ODD BYTE OF SOURCE & DEST DATA
3644                                     ;
3645 016414 013702 001506                      MOV      %FACTOR,R2 ;GET INDEX VALUE
3646 016420 010204                      MOV      R2,R4 ;R2 FOR SOURCE EVEN BYTE INDEX, R4 FOR
3647 016422 010403                      MOV      R4,R3 ;DEST ODD BYTE, R3 FOR SOURCE EVEN
3648 016424 005203                      INC      R3 ;AND R5 FOR DEST ODD BYTE
3649 016426 010305                      MOV      R3,R5
3650 016430 000261                      SEC      ;SET CARRY
3651 016432 012762 125252 016554      MOV      #125252,SDATAB(2)
3652 016440 112763 177125 016554      MOVB     #177125,SDATAB(3) ;SOURCE DATA = 052652
3653 016446 016264 016554 016556      MOV      SDATA(2),DDATAB(4)
3654 016454 052764 125125 016556      BIS      #125125,DDATAB(4) ;DEST DATA = 177777
3655 016462 136263 016554 016554      BITB     SDATA(2),SDATAB(3)
3656 016470 001401                      BEQ      .+4

```

```

3658 016474 146264 016554 016556 BICB SDATEB(2),DDATEB(4)
3659 016502 103401 BCS .+4
3660 016504 104000 HLT ;ERROR MOV,BIS,BIT;BIC DO NOT AFFECT 'C'
3661 016506 126364 016554 016556 CMPB SDATEB(3),DDATEB(4)
3662 016514 001401 BEQ .+4
3663 016516 104000 HLT
3664 016520 146365 016554 016556 BICB SDATEB(3),DDATEB(5)
3665 016526 126265 016554 016556 CMPB SDATEB(2),DDATEB(5)
3666 016534 001401 BEQ .+4
3667 016536 104000 HLT
3668 016540 136564 016556 016556 BITB DDATEB(5),DDATEB(4)
3669 016546 001401 BEQ .+4
3670 016550 104000 HLT
3671 016552 000412 BR UB7 ;RESERVE TWO WORDS
3672 016554 000000 SDATEB: .WORD 0 ;RESERVED FOR SOURCE DATA
3673 016556 000000 DDATEB: .WORD 0 ;RESERVED FOR DEST DATA
3674
3675 ;*****
3676 ;*TEST 30 CHECK BINARY WORD OPS USING ADDRESS MODE 7
3677 ;* R2=ADDRESS OF SOURCE DATA, AND R3= ADDRESS OF DEST DATA
3678 ;*****
3679
3680
3681
3682 016560 112737 000030 001202 ST30: MOVB #30,20STSTM ;LOAD TEST NUMBER
3683 016566 000004 SCOPE
3684 016570 000000 SBIN7: .WORD 0 ;CONTAINS ADDRESS OF SOURCE DATA
3685 016572 000000 DBIN7: .WORD 0 ;CONTAINS ADDRESS OF DEST DATA
3686 016574 000000 .WORD 0 ;CONTAINS SOURCE DATA
3687 016576 000000 .WORD 0 ;CONTAINS DEST DATA
3688
3689 UB7: MOV PC,R0
3690 016602 024040 CMP -(R0),-(R0)
3691 016604 010002 MOV R0,R2
3692 016606 024242 CMP -(R2),-(R2)
3693 016610 010012 MOV R0,(R2)
3694 016612 010203 MOV R2,R3
3695 016614 024043 CMP -(R0),-(R3)
3696 016616 010013 MOV R0,(R3)
3697
3698 016620 000261 SEC
3699 016622 012777 100000 177740 MOV #100000,2SBIN7 ;SOURCE DATA = 100000
3700 016630 017777 177734 177734 MOV 2SBIN7,2DBIN7 ;DEST DATA = 100000
3701 016636 103001 MOV7
3702 016640 100401 BMI .+4
3703 016642 104000 HLT
3704 016644 006377 177722 MOV7: ASL 2DBIN7 ;DEST DATA = 000000
3705 016650 102001 BVC .+4
3706 016652 001401 BEQ .+4
3707 016654 104000 HLT
3708
3709 016656 027777 177706 177706 CMP 2SBIN7,2DBIN7 ;(R2)=100000,(R3)=000000
3710 016664 103402 BCS CMP7
3711 016666 102401 BVS CMP7
3712 016670 100401 BMI +4

```

NO7

CEOKCC PDP 11 70 CPU EXERCISOR MACY11 30A(1052) 03-MAR-78 13:15 PAGE 72  
CEOKCC.P11 03-MAR-78 13:13 T30 CHECK BINARY WORD OPS USING ADDRESS MODE 7

SEQ 0091

3714

```

3716 016702 103003      BCC      SUB7
3717 016704 102002      BVC      SUB7
3718 016706 001401      BEQ      SUB7
3719 016710 100401      BMI      .+4
3720 016712 104000      SUB7:    HLT
3721
3722 016714 006277 177650      ASR      @SBIN7      ;(R2)=140000
3723 016720 067777 177644 177644      ADD      @SBIN7,@DBIN7 ;(R2)=140000,(R3)=040000
3724 016726 103003      BCC      ADD7
3725 016730 102002      BVC      ADD7
3726 016732 001401      BEQ      ADD7
3727 016734 100001      SPL      .+4
3728 016736 104000      ADD7:    HLT
3729
3730 016740 047777 177624 177624      BIC      @SBIN7,@DBIN7 ;(R2)=140000,(R3)=000000
3731 016746 001401      BEQ      .+4
3732 016750 104000      HLT
3733
3734 016752 057777 177612 177612      BIS      @SBIN7,@DBIN7 ;(R2)=140000,(R3)=140000
3735 016760 100401      BMI      .+4
3736 016762 104000      HLT
3737
3738 016764 027777 177600 177600      CMP      @SBIN7,@DBIN7
3739 016772 001401      BEQ      .+4
3740 016774 104000      HLT
3741
3742
3743
3744
3745 016776 112737 000031 001202  TST31:  MOV      #31,@$TSTNM      ;LOAD TEST NUMBER
3746 017004 000004      SCOPE
3747 017006 005000      CLR      R0
3748 017010 005067 000072      CLR      1$
3749 017014 010707      MOV      PC,PC
3750 017016 120707      CMPB     PC,PC
3751 017020 030707      BIT      PC,PC
3752 017022 060007      ADD      R0,PC
3753 017024 105707      TSTB     PC
3754 017026 005507      ROR      PC
3755 017030 021007      CMP      (R0),PC
3756 017032 131007      BITB     (R0),PC
3757 017034 062707 000000      ADD      #0,PC
3758 017040 023707 001506      CMP      @#FACTOR,PC
3759 017044 133707 001506      BITB     @#FACTOR,PC
3760 017050 000240      NOP
3761
3762
3763 017052 163707 001506      SUB      @#FACTOR,PC      ;JUMPS TO UNRELOCATED CODE
3764 017056 063707 001506      ADD      @#FACTOR,PC      ;RETURNS
3765 017062 000240      NOP
3766 017064 024607      CMP      -(SP),PC
3767 017066 132607      BITB     (SP)+,PC
3768 017070 026707 000012      CMP      1$,PC
3769 017074 166707 000006      SUB      1$,PC
3770 017100 046707 000002      BIC      1$,PC

```

```

*****
: *TEST 31      SOME MISCELLANEOUS OPERATIONS INVOLVING THE PC
: *      NOTE: NONE OF THESE OPERATIONS SHOULD AFFECT THE PC
*****

```

```

; THE NEXT TWO INSTRUCTION CAUSE THE PROGRAM TO JUMP TO THE UNRELOCATED
; CODE AND TO RETURN ON THE FOLLOWING INST (IF THE CODE IS RELOCATED)

```

```

3772 017106 000000
3773 017110 000004
3774 017112 010702
3775 017114 062702 000012
3776 017120 012707 034242
3777 017124 000000
3778
3779
3780
3781
3782
3783 017126 112737 000032 001202
3784 017134 012767 000001 162154
3785 017142 000004
3786
3787
3788
3789 017144 010700
3790 017146 005740
3791 017150 010037 001512
3792 017154 010700
3793 017156 162700 017156
3794 017162 010037 001506
3795 017166 010737 001212
3796 017172 062737 000030 001212
3797 017200 013737 001212 001210
3798 017206 105737 001502
3799 017212 001402
3800 017214 000167 002250
3801 017220 012703 125252
3802 017224 010304
3803 017226 140304
3804 017230 022704 125000
3805 017234 001401
3806 017236 104000
3807
3808 017240 005004
3809 017242 150304
3810 017244 022704 000252
3811 017250 001401
3812 017252 104000
3813
3814 017254 110404
3815 017256 022704 177652
3816 017262 001401
3817 017264 104000
3818
3819 017266 132704 177525
3820 017272 001401
3821 017274 104000
3822
3823 017276 105104
3824 017300 110404
3825 017302 022704 000125
3826 017306 001401

```

```

IS: 0
RELE2: SCOPE
      MOV PC,R2
      ADD #12,R2
      MOV #RELOC,PC ;GO RELOCATE PROGRAM CODE
REL22: .WORD 0
;222222222222 LAST ADDRESS OF CODE TO BE RELOCATED 2222222222
;*****
; *TEST 32 CHECK BINARY BYTE OPS USING ADDRESS MODE 0
;*****
TST32: MOVB #32,#STSTMM ;LOAD TEST NUMBER
      MOV #1,#TIMES ;;DO 1 ITERATION
      SCOPE
;SBTTL START OF SECTION 3
;333333333333 FIRST ADDRESS TO BE RELOCATED 3333333333
REL3: MOV PC,R0 ;GET PC
      TST -(R0) ;R0 CONTAINS THE ADDRESS OF REL3
      MOV R0,#FRSTAD ;SAVE
      MOV PC,R0 ;GET CURRENT PC
      SUB #,R0 ;SUBTRACT RELOCATION FACTOR
      MOV R0,#FACTOR ;SAVE RELOCATION FACTOR
      MOV PC,#SLPERR ;SET LOOP ADDRESS
      ADD #30,#SLPERR ;ADJUST
      MOV #SLPERR,#SLPADR
      TSTB #NEXEC ;BR IF TEST CODE TO BE EXECUTED
      BEQ .+6
      JMP RELE3
      MOV #125252,R3
      MOV R3,R4 ;R3=R4=125252
      BICB R3,R4 ;R3=125252,R4=125000
      CMP #125000,R4 ;CHECK RESULT
      .+4
      CLR R4 ;R3=125252,R4=0
      BITB R3,R4 ;R3=125252,R4=000252
      CMP #252,R4 ;CHECK RESULT
      BEQ .+4
      MOVB R4,R4 ;R4=177652
      CMP #177652,R4 ;CHECK RESULT
      BEQ .+4
      BITB #177525,R4
      BEQ .+4
      COMB R4 ;R4=177525
      MOVB R4,R4 ;R4=000125
      CMP #125,R4 ;CHECK RESULT
      BEQ .+4

```

```

3828
3829 017312 150304          BISB  R3,R4          ;R3=125252,R4=000377
3830 017314 105204          INCB  R4
3831 017316 001401          BEQ   .+4
3832 017320 104000          HLT
3833
3834 ;*****
3835 ;*TEST 33 CHECK BINARY BYTE OPS USING ADDRESS MODE 7
3836 ;*****
3836 017322 112737 000033 001202 †ST33: MOVB  #33,2#STSTNM ;LOAD TEST NUMBER
3837 017330 000004          SCOPE
3838 017332 000406          BR    BINB7          ;RESERVE SPACE FOR ADDRESSES & DATA
3839 017334 000000          SBINB7: .WORD 0      ;CONTAINS ADDRESS OF SOURCE EVEN BYTE
3840 017336 000000          .WORD 0              ;CONTAINS ADDRESS OF SOURCE ODD BYTE
3841 017340 000000          .WORD 0              ;CONTAINS ADDRESS OF DEST EVEN BYTE
3842 017342 000000          .WORD 0              ;CONTAINS ADDRESS OF DEST ODD BYTE
3843 017344 000000          DBINB7: .WORD 0      ;CONTAINS SOURCE DATA
3844 017346 000000          .WORD 0              ;CONTAINS DEST DATA
3845
3846 017350 010700          BINB7: MOV   PC,R0
3847 017352 024040          CMP   -(R0),-(R0)    ;R0 = ADDRESS OF DEST DATA
3848 017354 010060 177772          MOV   RO,-6(R0)      ;LOAD ADDRESS OF DEST EVEN BYTE DATA
3849 017360 010060 177774          MOV   RO,-4(R0)
3850 017364 005260 177774          INC   -4(R0)          ;LOAD ADDRESS OF DEST ODD BYTE DATA
3851 017370 005740          †ST  -(R0)            ;R0=ADDRESS OF SOURCE DATA
3852 017372 010060 177770          MOV   RO,-10(R0)     ;LOAD ADDRESS OF SOURCE EVEN BYTE DATA
3853 017376 010060 177772          MOV   RO,-6(R0)
3854 017402 005260 177772          INC   -6(R0)          ;LOAD ADDRESS OF SOURCE ODD BYTE DATA
3855
3856 017406 005002          CLR   R2              ;SET INDEX REGISTERS
3857 017410 012703 000002          MOV   #2,R3           ;2#SBINB7(2);2#SBINB7(3) REFERENCE EVEN &
3858 017414 012704 177774          MOV   #-4,R4          ;ODD BYTE SOURCE DATA; 2#DBINB7(4);2#DBINB7(5)
3859 017420 012705 177776          MOV   #-2,R5          ;REFERENCE DEST EVEN& ODD BYTE DATA
3860
3861
3862 017424 005020          CLR   (R0)+           ;PRESET SOURCE DATA
3863 017426 005010          CLR   (R0)            ;PRESET DEST DATA
3864 017430 013746 001506          MOV   2#FACTOR,-(SP) ;GET RELOCATION FACTOR
3865 017434 061602          ADD   (SP),R2         ;AND ADD TO INDEX VALUES
3866 017436 061603          ADD   (SP),R3
3867 017440 061604          ADD   (SP),R4
3868 017442 062605          ADD   (SP)+,R5
3869
3870 017444 112773 177777 017334          MOVB  #-1,2#SBINB7(3) ;SRC DATA = 177400
3871 017452 132772 000377 017334          BITB  #377,2#SBINB7(2) ;CHECK THAT EVEN BYTE WAS NOT AFFECTED
3872 017460 001401          BEQ   .+4             ;BY MOVB INSTRUCTION
3873 017462 104000          HLT
3874
3875 017464 157374 017334 017344          BISB  2#SBINB7(3),2#DBINB7(4)
3876 017472 105274 017344          INCB  2#DBINB7(4)    ;CHECK THAT BIS SET ALL BITS
3877 017476 001401          BEQ   .+4
3878 017500 104000          HLT
3879
3880 017502 105375 017344          DECB  2#DBINB7(5)    ;DEST DATA = 177400
3881 017506 005274 017344          INC   2#DBINB7(4)    ;DEST DATA = 177401
3882 017512 127375 017334 017344          CMPB  2#SBINB7(3),2#DBINB7(5)

```



```

3884 017522 104000 HLT
3885
3886 017524 147375 017334 017344 BICB @SBIN87(3),@DBIN87(5)
3887 017532 001401 BEQ .+4
3888 017534 104000 HLT
3889
3890 017536 105073 017334 CLRB @SBIN87(3) ;SRC DATA = 000000
3891 ;THIS ROUTINE SETS ALL BITS IN THE SOURCE ODD BYTE BY BISING A BIT FROM
3892 ;THE DEST EVEN BYTE INTO THE SOURCE ODD BYTE
3893 017542 157473 017344 017334 BIS7: BISB @DBIN87(4),@SBIN87(3)
3894 017550 106174 017344 ROLB @DBIN87(4)
3895 017554 103372 BCC BIS7
3896 017556 022772 177400 017334 CMP #177400,@SBIN87(2) ;CHECK RESULT
3897 017564 001401 BEQ .+4
3898 017566 104000 HLT
3899
3900 017570 000372 017334 SWAB @SBIN87(2) ;SRC DATA = 000377
3901 017574 112775 000200 017344 MOVB #200,@DBIN87(5) ;DEST DATA = 100000
3902
3903 017602 147572 017344 017334 BIC7: BICB @DBIN87(5),@SBIN87(2)
3904 017610 106075 017344 RORB @DBIN87(5)
3905 017614 103372 BCC BIC7
3906 017616 005772 017334 TST @SBIN87(2)
3907 017622 001401 BEQ .+4
3908 017624 104000 HLT
3909
3910 017626 012702 000001 OAERR: MOV #1,R2 ;LOAD R2 WITH ODD #
3911 017632 010703 MOV PC,R3
3912 017634 000401 BR .+4 ;RESERVE SPACE FOR A WORD
3913 017636 000000 .WORD 0 ;WILL CONTAIN AN ODD ADDRESS
3914 017640 005723 TST (R3)+ ;STEP R3 TO POINT TO WORD ABOVE
3915 017642 010313 MOV R3,(R3)
3916 017644 005213 INC (R3) ;AND MAKE ODD
3917 017646 012737 017774 000004 MOV #1,@ERRVEC ;SET ODD ADDRESS & RESERVED INSTRUCTION
3918 017654 063737 001506 000004 ADD @FACTOR,@ERRVEC
3919 017662 013737 000004 000010 MOV @ERRVEC,@RESVEC ;TO TRAP TO 1$ BELOW
3920
3921 017670 000277 SCC ;SET ALL CC'S
3922 017672 160212 SUB R2,(R2)
3923 017674 104000 HLT
3924 017676 060222 ADD R2,(R2)+
3925 017700 104000 HLT
3926 017702 006342 ASL -(R2)
3927 017704 104000 HLT
3928 017706 106512 MFPD (R2) ;NOTE: MAY BE RESERVED
3929 017710 104000 HLT
3930 017712 170412 CLRF (R2)
3931 017714 104000 HLT
3932 017716 042202 BIC (R2)+,R2
3933 017720 104000 HLT
3934 017722 164202 SUB -(R2),R2
3935 017724 104000 HLT
3936 017726 155202 BISB @-(R2),R2
3937 017730 104000 HLT
3938 017732 105532 ADCB @R2)+

```

# F08

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13.15 PAGE 77  
T33 CHECK BINARY BYTE OPS USING ADDRESS MODE 7

SEQ 0096

```

3940 017736 163302          SUB    @ (R3)+,R2
3941 017740 104000          HLT
3942 017742 005733          TST    @ (R3)+
3943 017744 104000          HLT
3944 017746 106533          MFPD  @ (R3)+
3945 017750 104000          HLT
3946 017752 170453          CLRD  @-(R3)
3947 017754 104000          HLT
3948 017756 137702 177775      BITB  @.+1,R2
3949 017762 104000          HLT
3950 017764 105477 177773      NEGB  @.-1
3951 017770 104000          HLT
3952 017772 000406          BR    2$
3953
3954 017774 062716 000002      1$:  ADD    #2,(SP)          ;ADJUST RETURN PC
3955 020000 052766 000017 000002  BIS    #17,2(SP)       ;SET CONDITION CODES ON RETURN
3956 020006 000002          RTI
3957
3958 020010 012706 000700      2$:  MOV    #SUPSTK,SP      ;RESET STACK PTR
3959 020014 012737 053442 000004  MOV    #ERPRT,@ERRVEC ;RESET TIME OUT VECTOR
3960 020022 012737 053370 000010  MOV    #RESERR,@RESVEC
3961
3962
3963
3964 020030 112737 000034 001202  ;*****
3965 020036 000004          ;*TEST 34 CHECK JUMP INSTRUCTIONS
3966 020040 010700          ;*****
3967 020042 062700 000012  †ST34: MOVB  #34,@$STSTN' ;LOAD TEST NUMBER
3968 020046 000277          SCOPE
3969 020050 000110          MOV    PC,R0
3970 020052 000402          ADD    #12,R0          ;SET ADDRESS FOR JMP INST
3971 020054 000250          SCC          ;SET CC'S
3972 020056 000775          JMP    (R0)
3973
3974 020060 103003          BR    .+6
3975 020062 102002          ;JMP INST JUMPS HERE
3976 020064 001001          BCC    JMP1
3977 020066 100001          BVC    JMP1
3978 020070 104000          BNE    JMP1
3979
3980 020072 005002          BPL    .+4
3981 020074 010703          JMP1  ;ERROR! INCORRECT CC'S AFTER JMP
3982 020076 000401          CLR    R2          ;SET INDICATOR
3983 020100 000000          MOV    PC,R3
3984 020102 005723          BR    .+4          ;RESERVE WORD FOR JMP ADDRESS
3985 020104 010313          .WORD 0          ;CONTAINS ADDRESS FOR JMP INST
3986 020106 010300          TST    (R3)+
3987 020110 062713 000022      MOV    R3,(R3)
3988 020114 010300          MOV    R3,R0
3989 020116 000133          ADD    #2,(R3)      ;(R3) IS JMP ADDRESS
3990 020120 000402          MOV    R3,R0
3991 020122 005102          JMP    @ (R3)+      ;JUMP TO ADDRESS CONTAINED IN R3
3992 020124 000775          BR    .+6
3993 020126 005202          COM    R2          ;COMPLEMENT INDICATOR
3994 020130 001003          BR    .-4
          INC    R2
          BNE    JMP3

```

3996	020134	020003				CMP	R0,R3		;CHECK AUTO-INC R3
3997	020136	001401				BEQ	.+4		
3998	020140	104000			JMP3:	HLT			
3999									
4000	020142	005002				CLR	R2		;SET INDICATOR
4001	020144	010704				MOV	PC,R4		;SET UP JMP REGISTER
4002	020146	010400				MOV	R4,R0		;SET UP CHECK REGISTER
4003	020150	000402				BR	1\$		
4004	020152	005102				COM	R2		;COMPLEMENT INDICATOR
4005	020154	000403				BR	2\$		
4006	020156	022424			1\$:	CMP	(R4)+,(R4)+		
4007	020160	005724				TST	(R4)+		;R4=JMP ADDRESS
4008	020162	000144				JMP	-(R4)		;USE R4 AS ADDRESS
4009	020164	005202			2\$:	INC	R2		;CHECK INDICATOR
4010	020166	001003				BNE	JMP4		
4011	020170	022020				CMP	(R0)+,(R0)+		
4012	020172	020004				CMP	R0,R4		;CHECK AUTO-DEC R4
4013	020174	001401				BEQ	.+4		
4014	020176	104000			JMP4:	HLT			
4015									
4016	020200	010703				MOV	PC,R3		
4017	020202	000401				BR	.+4		;RESERVE WORD FOR JMP ADDRESS
4018	020204	000000			1\$:	.WORD	0		;CONTAINS JUMP ADDRESS
4019	020206	005723				TST	(R3)+		
4020	020210	010313				MOV	R3,(R3)		
4021	020212	062723	000016			ADD	#16,(R3)+		
4022	020216	010300				MOV	R3,R0		;LOAD CHECK REGISTER
4023	020220	000402				BR	3\$		
4024	020222	005102			2\$:	COM	R2		
4025	020224	000401				BR	4\$		
4026	020226	000153			3\$:	JMP	2-(R3)		;JUMP TO 2\$ VIA 1\$ ABOVE
4027	020230	005202			4\$:	INC	R2		;CHECK INDICATOR
4028	020232	001003				BNE	JMP5		
4029	020234	005740				TST	-(R0)		
4030	020236	020003				CMP	R0,R3		;CHECK AUTO-DEC R3
4031	020240	001401				BEQ	.+4		
4032	020242	104000			JMP5:	HLT			
4033									
4034	020244	000402				BR	2\$		
4035	020246	005102			1\$:	COM	R2		;COMPLEMENT INDICATOR
4036	020250	000402				BR	3\$		
4037	020252	000167	177770		2\$:	JMP	1\$		
4038	020256	005202			3\$:	INC	R2		
4039	020260	001401				BEQ	.+4		
4040	020262	104000			JMP6:	HLT			
4041									
4042	020264	012767	020302	000020		MOV	#1\$ 7\$		;SET UP JMP ADDRESS
4043	020272	063767	001506	000012		ADD	#FACTOR,7\$		;ADD RELOCATION FACTOR
4044	020300	000402				BR	2\$		;GO TO JMP 27\$ INST
4045	020302	005102			1\$:	COM	R2		;COMPLEMENT INDICATOR
4046	020304	000403				BR	3\$		;GO TO CHECK ROUTINE
4047	020306	000177	000000		2\$:	JMP	27\$		;JMP TO 1\$ ABOVE VIA 7\$
4048	020312	000000			7\$:	.WORD	0		;CONTAINS JMP ADDRESS
4049	020314	005202			3\$:	INC	R2		;CHECK INDICATOR
4050	020316	001401				BEQ	.+4		

# H08

```

4052      ;*****
4053      ;*TEST 35      CHECK JSR INSTRUCTIONS
4054      ;*****
4055      020322 112737 000035 001202 ↑T35:  MOV#B  #35,2#STSTNM      ;LOAD TEST NUMBER
4056      020330 000004          SCOPE
4057      020332 013705 001506          JSR1:  MOV      2#FACTOR,R5      ;GET RELOCATION FACTOR
4058      020336 012702 020370          MOV      #3$ R2      ;FORM DEST ADRS
4059      020342 060502          ADD      R5,R2      ;ADD RELOCATION FACTOR
4060      020344 000277          SCC          ;PRESET CC'S
4061      020346 000242          CLV
4062      020350 004512          JSR      R5,(R2)      ;GO TO 3$ VIA R2
4063      020352 005702          1$:   TST      R2      ;CHECK INDICATOR
4064      020354 001017          BNE     4$          ;R2 SHOULD=0
4065      020356 023705 001506          CMP     2#FACTOR,R5      ;CHECK THAT RTS  R5 RESTORED R5
4066      020362 001014          BNE     4$
4067      020364 000414          BR      JSR3          ;GO TO NEXT TEST
4068      020366 000205          2$:   RTS      R5      ;RETURN FROM SUBROUTINE
4069      020370 103011          3$:   BCC     4$          ;CHECK THAT JSR DID NOT
4070      020372 102410          BVS     4$
4071      020374 001007          BNE     4$          ;AFFECT CC'S
4072      020376 100006          BPL     4$
4073      020400 005002          CLR     R2          ;CLEAR INDICATOR
4074      020402 012704 020352          MOV     #1$ R4      ;GET UNRELOCATED RETURN ADDRESS
4075      020406 061604          ADD     (SP),R4      ;ADD RELOCATION FACTOR (OLD R5)
4076      020410 020405          CMP     R4,R5      ;CHECK THAT OLD R5 WAS PLACED ON THE
4077      020412 001765          BEQ     2$          ;STACK & THAT NEW R5 CONTAINS RETURN PC
4078      020414 104000          4$:   HLT
4079
4080      ;CHECK JSR INSTRUCTION ADDRESS MODE 3
4081      020416 013704 001506          JSR3:  MOV     2#FACTOR,R4      ;GET RELOCATION FACTOR
4082      020422 005000          CLR     R0          ;SET INDICATOR
4083      020424 012705 020444          MOV     #1$ R5
4084      020430 060405          ADD     R4,R5      ;SET UP JSR DEFERRED ADRS
4085      020432 010502          MOV     R5,R2
4086      020434 012715 020462          MOV     #5$ (R5)
4087      020440 060415          ADD     R4,(R5)      ;(R5)=DEST ADRS
4088      020442 000401          BR      2$          ;RESERVE WORD FOR ADDRESS
4089      020444 000000          1$:   .WORD 0          ;CONTAINS DEST ADRS FOR JSR
4090      020446 004435          2$:   JSR     R4,2(R5)+      ;JSR TO 5$ VIA 1$ ABOVE
4091      020450 005200          3$:   INC     R0          ;CHECK INDICATOR
4092      020452 001013          BNE     6$
4093      020454 000413          BR      JSR4
4094      020456 005100          4$:   COM     R0          ;COMPLEMENT INDICATOR
4095      020460 000204          RTS     4          ;RETURN FROM SUBROUTINE
4096      020462 012703 020450          5$:   MOV     #3$ R3      ;GET UNRELOCATED RETURN ADDRESS
4097      020466 061603          ADD     (SP),R3      ;ADD RELOCATION FACTOR (OLD R4)
4098      020470 020403          CMP     R4,R3
4099      020472 001003          BNE     6$
4100      020474 005722          TST     (R2)+
4101      020476 020205          CMP     R2,R5      ;CHECK AUTO-INC R5
4102      020500 001766          BEQ     4$          ;GO TO RTS
4103      020502 104000          6$:   HLT          ;ERROR ABOVE
4104
4105      ;CHECK JSR INST ADDRESS MODE 4
4106      020504 013704 001506          JSR4:  MOV     2#FACTOR,R4
    
```

```

4108 020512 010703      MOV      PC,R3
4109 020514 000401      BR       2$
4110 020516 000405      1$:     BR       4$
4111 020520 022323      2$:     CMP      (R3)+,(R3)+
4112 020522 000277      SCC
4113 020524 004443      JSR      R4,-(R3)      ;GO TO 2$
4114 020526 104000      3$:     HLT
4115 020530 000414      BR       JSR6          ;GO TO NEXT TEST
4116 020532 103012      4$:     BCC      5$
4117 020534 102011      BVC      5$
4118 020536 001010      BNE      5$
4119 020540 100007      BPL      5$
4120 020542 012702 020526    MOV      #3$,R2      ;GET UNRELOCATED RETURN ADDRESS
4121 020546 061602      ADD      (SP),R2    ;ADD RELOCATION FACTOR (OLD R4)
4122 020550 020204      CMP      R2,R4      ;CHECK THAT CALCULATED RETURN
4123 020552 001002      BNE      5$        ;PC = NEW R4
4124 020554 005724      TST      (R4)+
4125 020556 000204      RTS      R4
4126 020560 104000      5$:     HLT

4128      ;TEST JSR INST ADDRESS MODE 6
4129 020562 000401      JSR6:    BR       2$
4130 020564 000405      1$:     BR       3$
4131 020566 010700      2$:     MOV      PC,R0
4132 020570 004767 177770    JSR      PC,1$
4133 020574 100407      BMI      JSR7          ;GO TO NEXT TEST
4134 020576 104000      HLT      ;ERROR ON CC'S
4135 020600 022020      3$:     CMP      (R0)+,(R0)+
4136 020602 020016      CMP      R0,(SP)    ;CHECK THAT RETURN ADDRESS IS ON THE
4137 020604 001401      BEQ      .+4        ;STACK
4138 020606 104000      HLT
4139 020610 000270      SEN
4140 020612 000207      RTS      PC          ;SET N

4142      ;TEST JSR INST ADDRESS MODE 7
4143 020614 013746 001506    JSR7:    MOV      2*FACTOR,-(SP) ;GET RELOCATION FACTOR
4144 020620 062716 020640    ADD      #1$,(SP)    ;FORM ADDRESS OF 1$ BELOW
4145 020624 000277      SCC      ;SET ALL CC'S
4146 020626 004076 000000    JSR      R0,2*(SP)  ;JSR TO 1$
4147 020632 003003      BGT      3$
4148 020634 102002      BVC      3$
4149 020636 000402      BR       4$
4150
4151 020640 000200      1$:     RTS      R0      ;RETURN
4152 020642 104000      3$:     HLT      ;ERROR!! INCORRECT CC'S
4153 020644
4154
4155      ;*****
4156      ;*TEST 36 CHECK IOT TRAP (AND ROLB/ASLB)
4157      ;* THIS TEST CHECKS THAT THE PSW IS CORRECT AFTER THE IOT AND THAT THE
4158      ;* 'NEW'PSW (FROM IOTVEC+2) IS CORRECT.
4159      ;* *****
4159 020644 112737 000036 001202    IOT36:   MOVB    #36,2*STSTM      ;LOAD TEST NUMBER
4160 020652 000004      SCOPE
4161 020654 012705 000022    IOTTST:  MOV     #IOTVEC+2,R5
4162 020660 005000      CLR     R0

```

# JOB

CEQKCC PDP 11/70 CPU EXERCISOR MACY11 30A(1052) 03-MAR-78 13:15 PAGE 81  
 CEQKCC.P11 03-MAR-78 13:13 T36 CHECK IOT TRAP (AND ROLB/ASLB)

SEQ 0100

4164	020666	011015					MOV	(R0),(R5)	;SET IOTVEC+2 = PSW
4165	020670	011504					MOV	(R5),R4	;SAVE IN R4
4166	020672	010746					MOV	PC-(SP)	
4167	020674	062716	000036				ADD	#1\$-(SP)	
4168	020700	012645					MOV	(SP)+,(R5)	;LOAD IOT TRAP VECTOR
4169	020702	042710	000357				BIC	#PR7+17,(R0)	
4170	020706	052710	000244				BIS	#PRS+4,(R0)	;PSW=X XXX X00 101 1X1 000
4171	020712	012003					MOV	(R0)+,R3	;R3 = PSW ABOVE
4172	020714	010340					MOV	R3,-(R0)	;RESTORE PSW (MOV CHANGED IT)
4173	020716	000004					IOT		
4174	020720	012737	044534	000020	10\$:		MOV	##SCOPE,2#IOTVEC	;RESTORE IOT VECTOR
4175	020726	104000					HLT		;ERROR! IOT FAILED TO TRAP
4176	020730	000457					BR	TST37	;GO TO NEXT TEST
4177									
4178	020732	012002			1\$:		MOV	(R0)+,R2	;GET PSW AFTER IOT TRAP
4179									;NOTE: R0=0
4180	020734	012725	044534				MOV	##SCOPE,(R5)+	;RESTORE IOTVEC
4181	020740	012715	000200				MOV	#PR4,(R5)	;AND IOTVEC+2
4182	020744	010746					MOV	PC-(SP)	;FORM PC OF 10\$ ABOVE
4183	020746	062716	177752				ADD	#10\$-(SP)	
4184	020752	022626					CMP	(SP)+,(SP)+	;CHECK RETURN PC ON STACK
4185	020754	001036					BNE	99\$	
4186	020756	022603					CMP	(SP)+,R3	;CHECK SAVED PSW
4187	020760	001034					BNE	99\$	
4188	020762	032703	140000				BIT	#UM,R3	;BRANCH TO 3\$ IF IN USER MODE
4189	020766	100413					BMI	3\$	
4190	020770	001003					BNE	2\$	;BRANCH TO 2\$ IF IN SUPER MODE
4191	020772	020204					CMP	R2,R4	;CHECK PSW AFTER IOT
4192	020774	001026					BNE	99\$	
4193	020776	000413					BR	4\$	
4194									
4195	021000	042704	030000		2\$:		BIC	#PUM,R4	;CLEAR PREV MODE BITS
4196	021004	052704	010000				BIS	#PSM,R4	;SET PREV SUPER MODE
4197	021010	020204					CMP	R2,R4	;CHECK PSW AFTER IOT
4198	021012	001017					BNE	99\$	
4199	021014	000404					BR	4\$	
4200									
4201	021016	052704	030000		3\$:		BIS	#PUM,R4	;SET PREV USER MODE
4202	021022	020204					CMP	R2,R4	;CHECK PSW AFTER IOT
4203	021024	001012					BNE	99\$	
4204									
4205	021026	005002			4\$:		CLR	R2	
4206	021030	000261					SEC		
4207	021032	106100					ROLB	R0	;ROTATE R0
4208	021034	102375					BVC	.-2	;UNTIL V SETS (R0=200)
4209									
4210	021036	106300					ASLB	R0	;SHIFT SHOULD SET CARRY
4211	021040	103004					BCC	99\$	
4212	021042	102003					BVC	99\$	
4213	021044	001002					BNE	99\$	
4214	021046	005700					TST	R0	
4215	021050	001401					BEQ	+.4	
4216	021052	104000			99\$:		HLT		;ERROR! ROL/ASL FAILED TO SET
4217									;CC'S PROPERLY (IF R2=0) OR IN-
4218									;CORRECT PSW AFTER IOT (IF R2 NOT 0)

K08

CEQKCC POP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 82  
T36 CHECK IOT TRAP (AND ROLB/ASLB)

SEQ 0101

```

4220 021060 010437 177776          MOV      R4,2#PSW          ;RESTORE PSW
4221 021064 012706 000700          MOV      #SUPSTK,SP       ;RESTORE STACK PTR
4222                                     ;*****
4223                                     ;*TEST 37 CHECK EMT TRAP SEQUENCE
4224                                     ;*****
4225 021070 112737 000037 001202  †ST37:  MOVB     #37,2#STSTNM      ;LOAD TEST NUMBER
4226 021076 000004                                     ;
4227                                     .EQUIV   IOT,HLT          ;REDEFINE HLT CALL
4228 021100 012737 044774 000020     MOV      #ERROR,2#IOTVEC  ;SETUP VECTOR
4229 021106 012737 000340 000022     MOV      #PR7,2#IOTVEC+2
4230 021114 005000     CLR      RO
4231 021116 010746     MOV      PC,-(SP)
4232 021120 062716 000030     ADD      #EMT1-(SP)
4233 021124 012637 000030     MOV      (SP)+,2#EMTVEC
4234 021130 000262     SEV
4235 021132 013737 177776 000032     MOV      2#PSW,2#EMTVEC+2 ;SET V
4236 021140 000265     +SEZ!SEC ;RETAIN CURRENT PSW ON TRAP
4237 021142 104000     EMT
4238 021144 001433     BEQ      EMT1C           ;TRAP TO EMT1
4239 021146 000004     HLT
4240 021150 102027     EMT1:  BVC      EMT1B           ;GO TO EMT1C
4241 021152 105100     COMB    RO              ;ERROR! INCORRECT CC'S WERE SET ON RETURN
4242 021154 105500     ADCB   RO              ;'V' SHOULD'VE SET ON EMT TRAP
4243 021156 106000     RORB   RO              ;RO=000377,CC'S=1001
4244 021160 102023     BVC    EMT1B           ;RO=000000,CC'S=0101
4245 021162 100022     BPL    EMT1B           ;RO=000200,CC'S=1010
4246 021164 000257     CCC
4247 021166 105400     NEGB   RO              ;RO=000200,CC'S=1010
4248 021170 102017     BVC    EMT1B
4249 021172 100016     BPL    EMT1B
4250 021174 000242     CLV
4251 021176 000261     SEC
4252 021200 105300     DECB   RO              ;CLEAR 'V'
4253 021202 102012     BVC    EMT1B           ;AND SET 'C'
4254 021204 100411     BMI    EMT1B           ;RO=000177,CC'S=0011
4255 021206 000242     CLV
4256 021210 105200     INCB   RO              ;CLEAR 'V'
4257 021212 103006     BCC    EMT1A           ;RO=000200,CC'S=1011
4258 021214 102005     BVC    EMT1B
4259 021216 100004     BPL    EMT1B
4260 021220 000242     CLV
4261 021222 106200     ASRB   RO              ;CLEAR 'V'
4262 021224 102776     BVS    .-2             ;SHIFT RO UNTIL 'V' CLEARS
4263 021226 000401     BR     .+4
4264 021230 000004     EMT1B: HLT
4265 021232 000002     EMT1C: RTI              ;ERROR!
4266 021234 105500     ADCB   RO              ;EXIT WITH RO=000377
4267 021236 103003     BCC    EMT1D           ;RO=000000
4268 021240 001002     BNE    EMT1D
4269 021242 005700     TST    RO
4270 021244 001401     BEQ    .+4
4271 021246 000004     EMT1D: HLT
4272 021250 012737 044774 000030     MOV      #ERROR,2#EMTVEC  ;RESTORE EMT TO ERROR
4273 021256 012737 000340 000032     MOV      #PR7,2#EMTVEC+2 ;SET PRIORITY 7 ON ERROR
4274 021264 012737 044534 000020     MOV      #SCOPE,2#IOTVEC ;RESTORE IOT VECTOR

```

```

4276 .EQUIV ERROR,HLT ;REDEFINE HLT CALL
4277 ;*****
4278 ;*TEST 40 CHECK TRAP INSTRUCTION TRAP SEQUENCE
4279 ;*****
4280 021276 112737 000040 001202 †ST40: MOVB #40,‡STSTNM ;LOAD TEST NUMBER
4281 021304 000004 SCOPE
4282 021306 052737 000340 177776 BIS #PR7,‡‡PSW ;LOCK OUT LINE CLOCK
4283 021314 052737 000340 000016 BIS #PR7,‡‡TBITVEC+2
4284 021322 010746 MOV PC, -(SP)
4285 021324 062716 000056 ADD #TRAP1, -(SP)
4286 021330 012637 000034 MOV (SP)+,‡‡TRAPVEC
4287 021334 000270 SEN ;SET N
4288 021336 013737 177776 000036 MOV ‡‡PSW,‡‡TRAPVEC+2 ;RETAIN CURRENT PSW ON TRAP
4289 021344 000261 SEC ;SET CARRY
4290 021346 010700 MOV PC,R0
4291 021350 000264 SEZ ;SET Z BIT
4292 021352 104400 TRAP ;TRAP TO TRAP1
4293 021354 103404 BCS .+12
4294 021356 012737 051374 000034 MOV #STRAP,‡‡TRAPVEC ;RESTORE TRAP VECTOR
4295 021364 104000 HLT
4296 021366 001404 BEQ .+12
4297 021370 012737 051374 000034 MOV #STRAP,‡‡TRAPVEC ;RESTORE TRAP VECTOR
4298 021376 104000 HLT
4299 021400 000420 BR TRAP1C
4300 021402 100404 TRAP1: BMI .+12 ;N BIT GOT SET ON TRAP
4301 021404 012737 051374 000034 MOV #STRAP,‡‡TRAPVEC ;RESTORE TRAP VECTOR
4302 021412 104000 HLT
4303 021414 062700 000004 ADD #4,70
4304 021420 020016 CMP R0,(SP) ;CHECK LOW BYTE OF RETURN PC ON
4305 021422 001404 BEQ .+12 ;STACK
4306 021424 012737 051374 000034 MOV #STRAP,‡‡TRAPVEC ;RESTORE TRAP VECTOR
4307 021432 104000 HLT
4308 021434 124646 CMPB -(SP),-(SP)
4309 021436 032626 BIT (SP)+,(SP)+
4310 021440 000002 RTI ;RETURN TO INST FOLLOWING TRAP (1$)
4311
4312 021442 012702 000036 TRAP1C: MOV #TRAPVEC+2,R2 ;RESTORE VECTORS
4313 021446 012712 000340 MOV #PR7,(R2)
4314 021452 012742 051374 MOV #STRAP,-(R2)
4315 021456 042737 000340 000016 BIC #PR7,‡‡TBITVEC+2
4316 021464 105037 177776 CLRB ‡‡PSW ;GO BACK TO PRIORITY 0
4317
4318 021470 000004 RELE3: SCOPE
4319 021472 010702 MOV PC,R2
4320 021474 062702 000012 ADD #12,R2
4321 021500 012707 034242 MOV #RELOC,1'C ;GO RELOCATE PROGRAM CODE
4322 021504 000000 REL33: .WORD 0
4323 ;3333333333333333 LAST ADDRESS OF CODE TO BE RELOCATED 333333333333
4324
4325 ;*****
4326 ;*TEST 41 CHECK STACK OVERFLOW
4327 ;*****
4328 021506 112737 000041 001202 †ST41: MOVB #41,‡‡STSTNM ;LOAD TEST NUMBER
4329 021514 012767 000001 157574 MOV #1,$TIMES ;DO 1 ITERATION
4330 021522 000004 SCOPE

```



```

4332          SBTTL START OF SECTION 4
4333          ;4444444444444444 FIRST ADDRESS TO BE RELOCATED 4444444444
4334 REL4:    MOV    PC,RO          ;GET PC
4335          TST    -(AD)         ;AD CONTAINS THE ADDRESS OF REL4
4336          MOV    RO,#FRSTAD    ;SAVE
4337          MOV    PC,RO          ;GET CURRENT PC
4338          SUB    #,RO          ;SUBTRACT RELOCATION FACTOR
4339          MOV    RO,#FACTOR    ;SAVE RELOCATION FACTOR
4340          MOV    PC,#SLPERR    ;SET LOOP ADDRESS
4341          ADD    #30,#SLPERR    ;ADJUST
4342          MOV    #SLPERR,#SLPADR
4343          TSTB   #NEXEC        ;BR IF TEST CODE TO BE EXECUTED
4344          BEQ   .+6
4345          JMP   RELE4
4346
4347          021600 013767 177776 000334 OVFLW: MOV    #PSW,7$      ;SAVE STATUS IN 7$ BELOW
4348          021606 005037 177776          CLR    #PSW          ;SET KERNEL MODE
4349          021612 004737 052340          JSR   PC,#CLRTBIT    ;GO CLEAR 'T' BIT IF SET
4350          021616 052737 000340 177776  BIS    #P7,#PSW      ;SET PRIORITY LEVEL 7 TO BLOCK CLOCK
4351          021624 010746          MOV    PC,-(SP)      ;PUSH CURRENT PC ONTO STACK
4352          021626 062716 000152          ADD    #2$,-(SP)     ;FORM ADDRESS OF 2$ BELOW
4353          021632 011637 000004          MOV    (SP),#ERRVEC ;SET ERROR VECTOR
4354          021636 012737 000340 000006  MOV    #340,#ERRVEC+2 ;SET PRIORITY LEVEL 7 ON TRAP
4355          021644 013727 000014          MOV    #BPTVEC,(PC)+ ;SAVE BPT VECTOR ADDR
4356          021650 000000          ;
4357          021652 062716 000100 43$:   .WORD 0
4358          021656 012637 000014          ADD    #41$-2$(SP)  ;FORM ADDRESS OF 41$ BELOW
4359          021662 012737 000340 000016  MOV    (SP)+,#BPTVEC ;SET BPT TRAP VECTOR TO 41$
4360
4361          021670 012703 000376          MOV    #376,R3
4362          021674 010313          MOV    R3,(R3)      ;LOAD 376 INTO ADDRESS 376
4363          021676 010306          MOV    R3,SP        ;SET STACK PTR AT BOUNDARY
4364          021700 032767 140000 000234  BIT    #UH,7$
4365          021706 001015          BNE   1$           ;CHECK IF ENTERED TEST IN KERNEL
4366                                     ;MODE. BRANCH IF NOT IN KERNEL
4367          ;THE BELOW INSTRUCTIONS SHOULD NOT CAUSE AN OVERFLOW TRAP
4368          021710 005716          TST   (SP)          ;BECAUSE TST IS A NON MODIFYING INST
4369          021712 021666 177776          CMP   (SP),-2(SP)  ;SO IS COMPARE
4370          021716 012656          MOV   (SP)+,#-(SP) ;BECAUSE OF ADDRESS MODE 5
4371          021720 057636 000000          BIS   #-(SP),#-(SP) ;BECAUSE OF ADDRESS MODE 3
4372          021724 054676 000000          BIS   -(SP),#-(SP) ;BECAUSE OF ADDRESS MODE 7
4373          021730 005006          CLR   SP
4374          021732 013766 020000 020000  MOV   #20000,20000(SP)
4375          021740 000425          BR    3$           ;BRANCH OVER NON KERNEL MODE TESTS
4376
4377          ;NOTE: NO OVEFLOW TRAP WILL OCCUR IF NOT IN KERNEL MODE!!!
4378          021742 156737 000175 177777 1$:   BISB  7$+1,#PSW+1  ;RESTORE MODE BITS IN PSW
4379          021750 012706 000376          MOV   #376,SP      ;SET STACK PTR
4380          021754 016646 177776          MOV   -2(SP),-(SP) ;SHOULD NOT TRAP
4381          021760 051616          BIS   (SP),(SP)
4382          021762 061666 177776          ADD   (SP),-2(SP)
4383          021766 105037 177777          CLRB  #PSW+1       ;SET KERNEL MODE
4384          021772 012706 000700          MOV   #SUPSTK,SP   ;RESTORE THE STACK
4385          021776 000451          BR    6$           ;EXIT TEST
4386

```

```

4388 022000 012600      25:  MOV      (SP)+,R0      ;SAVE PC OF INSTRUCTION THAT TRAPPED
4389 022002 012602      MOV      (SP)+,R2      ;SAVE PSW
4390 022004 012706 000700  MOV      #SUPSTK,SP    ;SET STACK PTR
4391 022010 104000      HLT                      ;ERROR! AN INSTRUCTION THAT WAS NOT
4392                                     ;SUPPOSED TO TRAP TRAPPED
4393                                     ;R0 CONTAINS PC, R2 CONTAINS PSW
4394 022012 000443      BR        65            ;EXIT TEST
4395                                     ;THE BELOW INSTRUCTIONS WILL CAUSE A STACK OVERFLOW
4396                                     ;STACK PTR IS AT 376
4397 022014 062737 000066 000004 35:  ADD      #45-25,2#ERRVEC ;SET ERROR VECTOR TO 45
4398 022022 010306      MOV      R3,SP         ;SET STACK PTR AT 376
4399 022024 112702 000001      MOV      #1,R2
4400 022030 005000      CLRB     R0
4401 022032 005016      CLR      (SP)          ;SETS BIT 0 IN R0
4402 022034 006302      ASL      R2            ;SHIFT INDICATOR BIT
4403 022036 105226      INCB     (SP)+         ;SETS BIT 1 IN R0
4404 022040 006302      ASL      R2
4405 022042 060746      ADD      PC,-(SP)      ;SETS BIT 2 IN R0
4406 022044 006302      ASL      R2
4407 022046 000003      BPT                      ;SETS BIT 3 IN R0
4408 022050 006302      ASL      R2
4409 022052 004767 000014      JSR      PC,40$        ;SETS BIT 4 IN R0
4410 022056 006302      ASL      R2
4411 022060 050666 177776      BIS      SP,-2(SP)     ;SETS BIT 5 IN R0
4412 022064 000410      BR        55
4413                                     ;PROGRAM WILL TRAP HERE ON OVERFLOW TRAP
4414 022066 050200      45:  BIS      R2,R0         ;SET APPROPRIATE BIT IN R0
4415 022070 000002      RTI                      ;RETURN FROM TRAP
4416
4417 022072 052700 001000      40$:  BIS      #1000,R0     ;SET IND THAT JSR WAS EXECUTED
4418 022076 000207      RTS      PC
4419
4420 022100 052700 000400      41$:  BIS      #400,R0      ;SET IND THAT BPT WAS EXECUTED
4421 022104 000002      RTI
4422
4423                                     ;CHECK THAT ABOVE INSTRUCTIONS DID TRAP
4424 022106 012706 000700      55:  MOV      #SUPSTK,SP    ;SET STACK PTR
4425 022112 022700 001477      CMP      #1477,R0      ;EACH INSTRUCTION SET A BIT IN R0
4426 022116 001401      BEQ      .+4           ;R0= 1477
4427 022120 104000      HLT
4428
4429                                     ;EXIT ROUTINE
4430 022122 012706 001200      65:  MOV      #KERSTK,SP   ;SET KERNEL STACK PTR
4431 022126 01E737 177516 000014  MOV      43$,2#BPTVEC  ;RESTORE BPT VECTOR
4432 022134 005037 000016      CLR      2#BPTVEC+2
4433 022140 012746      MOV      (PC)+,-(SP)   ;PUSH OLD PSW ONTO STACK
4434 022142 000000      .WORD    0             ;CONTAINS SAVED PSW
4435 022144 010746      MOV      PC,-(SP)     ;PUSH CURRENT PC ONTO STACK
4436 022146 062716 000006      ADD      #6,(SP)      ;ADD OFFSET
4437 022152 000002      RTI
4438 022154 012706 000700      MOV      #SUPSTK,SP   ;SET STACK PTR
4439 022160 012737 053442 000004  MOV      #ERPRT,2#ERRVEC ;RESET TIME OUT VECTOR
4440 022166 013737 177776 000006  MOV      2#PSW,2#ERRVEC+2
4441 022174 052737 000340 000006  BIS      #PR7,2#ERRVEC+2

```

```

444 022210 005037 177766          CLR      @#CPUERR
4445          :*****
4446          :*TEST 42      CHECK THAT ALL RESERVED INSTRUCTIONS TRAP
4447          :*****
4448          :*ST42:     MOV#   #42,@#STSTNM          ;LOAD TEST NUMBER
4449          SCOPE
4450          RESTRP: MOV    #55,R2          ;GET ADDRESS OF RESERVED INSTRUCTION TABLE
4451          ADD    @#FACTOR,R2
4452          BITB   #40,@#OPT.CP+1        ;CHECK IF 11/45 FLOATING POINT IS AVAIL.
4453          BEQ    .+6                    ;BRANCH IF NOT AVAILABLE
4454          CLR    #50$                   ;SET TABLE TERMINATOR AT GROUP 7
4455          MOV    #45,@#RESVEC          ;SET RESERVED INSTRUCTION TRAP
4456          ADD    @#FACTOR,@#RESVEC
4457          1$:   MOV    (R2)+,R3          ;GET FIRST RESERVED INSTRUCTION
4458          BEQ    7$                    ;0 TERMINATES THE TABLE
4459          MOV    (R2)+,R4              ;GET LAST RESERVED INSTRUCTION IN GROUP
4460          2$:   MOV    R3,(PC)          ;EXECUTE RESERVED INSTRUCTION
4461          3$:   .WORD 0                  ;CONTAINS RESERVED INSTRUCTION
4462          HLT
4463          HLT
4464          HLT
4465          BR     41$                   ;ERROR! INSTRUCTION IN R3
4466          MOV    #415,(SP)             ;(2$) ABOVE FAILED TO CAUSE A
4467          ADD    @#FACTOR,(SP)         ;RESERVED INSTRUCTION TRAP
4468          RTI
4469          41$:  CMP    R3,R4            ;ADJUST RETURN PC
4470          BEQ    1$                    ;TO RETURN TO 41$
4471          INC    R3                    ;RETURN TO 41$
4472          BR     2$                    ;HAS GROUP OF RESERVED INSTRUCTIONS
4473          5$:   .TABLE OF 11/40,11/45 RESERVED INSTRUCTIONS (0 TERMINATES THE TABLE)
4474          7     7                      ;GROUP 1
4475          210   210                     ;GROUP 2
4476          227   227                     ;GROUP 3
4477          7000  7000                    ;GROUP 4
4478          7777  7777                    ;GROUP 5
4479          75040 75040                   ;GROUP 6
4480          76777 76777                   ;GROUP 7
4481          106400 106400                  ;GROUP 7
4482          106477 106477                  ;GROUP 7
4483          106700 106700                  ;GROUP 7
4484          107777 107777                  ;GROUP 7
4485          170000 170000                  ;GROUP 7
4486          177777 177777                  ;GROUP 7
4487          0
4488          0
4489          0
4490          022366 012737 053370 000010 7$:   MOV    @#RESERR,@#RESVEC          ;RESTORE RESERVED TRAP
4491          :*****
4492          :*TEST 43      CHECK THAT ALL BITS IN THE PSW CAN BE SET AND CLEARED
4493          :*****
4494          :*ST43:     MOV#   #43,@#STSTNM          ;LOAD TEST NUMBER
4495          SCOPE
4496          PSWCHK: TSTB   @#MMON           ;IF MEM MGMT IS ON SKIP THIS TEST
4497          BNE    4$
4498          MOV    @#PSW,3$              ;SAVE STATUS

```

```

4500 022424 004737 052340 JSR PC, @CLRTBIT ;GO CLEAR 'T' BIT IF SET
4501 022424 013746 000016 MOV @BITVEC+2, -(SP)
4502 022424 012704 177776 MOV #PSW, R4 ;LOAD ADDRESS OF PSW INTO R4
4503 022424 000250 CLN ;
4504 022424 005714 TST (R4) ;CHECK THAT PSW WAS CLEARED
4505 022424 001401 BEQ .+4 ;
4506 022424 104000 HLT ;ERROR! PSW FAILED TO CLEAR
4507 022424 012700 170357 MOV #170357, R0
4508 022424 052700 170000 BIS #170000, R0 ;SET BITS 15-12 IF MEM MGMT
4509 022424 012702 000001 MOV #1, R2 ;R2 = TEST BIT
4510 022424 030200 10$: BIT R2, R0 ;CHECK IF BIT CAN BE SET/CLEARED
4511 022424 001423 BEQ 20$ ;
4512 022424 005037 000016 CLR @BITVEC+2
4513 022424 030227 000020 BIT R2, #20 ;CHECK IF TEST WILL SET 'T' BIT
4514 022424 001403 BEQ 20$ ;
4515 022424 012737 000002 000016 MOV #RTI, @BITVEC+2 ;SET RTI INTO RETURN
4516 022424 005014 20$: CLR (R4) ;CLEAR PSW
4517 022424 050214 BIS R2, (R4) ;SET R2 INTO PSW
4518 022424 011403 MOV (R4), R3 ;GET BIT
4519 022424 020203 CMP R2, R3 ;CHECK THAT BIT WAS SET IN PSW
4520 022424 001401 BEQ .+4 ;
4521 022424 104000 HLT ;ERROR! BIT IN R2 FAILED TO SET IN PSW
4522 022424 000244 CLZ ;CLEAR Z BIT
4523 022424 040214 BIC R2, (R4) ;CLEAR BIT IN PSW
4524 022424 011403 MOV (R4), R3 ;GET PSW RESULT
4525 022424 001401 BEW 25$ ;BRANCH IF BIC ABOVE CLEARED BIT IN PSW
4526 022424 104000 HLT ;ERROR! BIT IN R2 FAILED TO CLEAR IN PSW
4527 022424 006302 25$: ASL R2 ;SHIFT TEST BIT
4528 022424 103351 BCC 1$ ;BRANCH IF ALL BITS NOT TESTED
4529 022424 005014 CLR (R4) ;CLEAR STATUS
4530 022424 012637 000016 MOV (SP)+, @BITVEC+2 ;RESTORE T BIT RETURN
4531 022424 012746 MOV (PC)+, -(SP) ;PUSH ORIGINAL STATUS ON STACK
4532 022424 000000 35$: .WORD 0 ;CONTAINS ORIGINAL PSW
4533 022424 010746 MOV PC, -(SP) ;SET RETURN PC
4534 022424 062716 000006 ADD #6, (SP) ;
4535 022424 000002 RTI ;RETURN
4536 022424 013704 177776 45$: MOV @PSW, R4 ;SAVE PSW IN R4
4537 022424 112737 000340 177776 MOVB #340, @PSW ;SET PRIORITY LEVEL 7
4538 022424 004737 052340 JSR PC, @CLRTBIT ;GO CLEAR 'T' BIT IF SET
4539 *****
4540 ;*TEST 44 CHECK THAT ALL BITS IN THE CURRENT STACK PTR CAN BE SET CLEARED
4541 *****
4542 022602 112737 000044 001202 †ST44: MOVB #44, @STSTNM ;LOAD TEST NUMBER
4543 022610 000004 SCOPE ;
4544 022612 010603 CHKSP: MOV SP, R3 ;SAVE STACK PTR
4545 022614 000257 CCC ;
4546 022616 112706 000377 MOVB #377, SP ;SET STACK PTR = -1
4547 022622 006006 1$: ROR SP ;ROTATE 0 BIT THROUGH ALL BIT
4548 022624 103776 BCS 1$ ;BIT POSITIONS
4549 022626 005206 INC SP ;SHOULD INCREMENT SP TO 0
4550 022630 001403 BEQ 25$ ;
4551 022632 010602 MOV SP, R2 ;SAVE ERROR STACK PTR
4552 022634 010306 MOV R3, SP ;SET STACK PTR FOR TRAP
4553 022636 104000 HLT ;ERROR!
4554

```

```

4556
4557
4558 022642 010600
4559 022644 010003
4560
4561 022646 005043
4562 022650 112746 177777
4563 022654 022713 000377
4564 022660 001002
4565 022662 020306
4566 022664 001401
4567 022666 104000 1$: HLT
4568
4569 022670 105226
4570 022672 005723
4571 022674 001002
4572 022676 020006
4573 022700 001401
4574 022702 104000 2$: HLT
4575
4576 022704 005143
4577 022706 144613
4578 022710 022713 177400
4579 022714 001002
4580 022716 020603
4581 022720 001401
4582 022722 104000 3$: HLT
4583
4584 022724 132627 000377
4585 022730 001002
4586 022732 020600
4587 022734 001401
4588 022736 104000 4$: HLT
4589
4590 022740 012746 000001
4591 022744 062706 000002
4592 022750 012702 177401
4593 022754 120246
4594 022756 001004
4595 022760 122602
4596 022762 001002
4597 022764 020006
4598 022766 001401
4599 022770 104000 5$: HLT
4600 022772 105037 177776
4601 022776 010446
4602 023000 010746
4603 023002 062716 000006
4604 023006 000002
4605
4606
4607
4608 023010 112737 000045 001202
4609 023016 000004
4610 023020 012727 177776

;CHECK BYTE OPERATIONS USING THE STACK
SPCHK: MOV SP,R0 ;SAVE STACK PTR
        MOV R0,R3
        CLR -(R3)
        MOVB #-1,-(SP) ;(SP) = 377
        CMP #377,(R3) ;CHECK THAT ONLY EVEN BYTE WAS AFFECTED
        BNE 1$
        CMP R3,SP ;CHECK AUTO-DEC
        BEQ .+4
1$: HLT
        INCB (SP)+
        TST (R3)+ ;CHECK RESULT
        BNE 2$
        CMP R0,SP ;CHECK AUTO-INC
        BEQ .+4
2$: HLT
        COM -(R3) ;(R3)=177777
        BICB -(SP),(R3)
        CMP #177400,(R3) ;CHECK RESULT
        BNE 3$
        CMP SP,R3
        BEQ .+4
3$: HLT
        BITB (SP)+,#377
        BNE 4$
        CMP SP,R0
        BEQ .+4
4$: HLT
        MOV #1,-(SP)
        ADD #2,SP
        MOV #177401,R2
        CMPB R2,-(SP)
        BNE 5$
        CMPB (SP)+,R2
        BNE 5$
        CMP R0,SP
        BEQ .+4
5$: HLT
        CLRB 2#PSW
        MOV R4,-(SP) ;RESTORE ORIGINAL PSW TO STACK
        MOV PC,-(SP)
        ADD #6,(SP)
        RTI
;*****
;TEST 45 CHECK THAT 'C' BIT SETS/CLEARs PROPERLY
;*****
TST45: MOVB #45,2#STSTNM ;LOAD TEST NUMBER
        SCOPE
CBIT: MOV #177776,(PC)+ ;LOAD CONSTANT

```

```

4612 023026 010700          MOV      PC,RO      ;GET CURRENT PC
4613 023030 162700 000004   SUB      #4,RO      ;POINT RO TO 1$ ABOVE
4614 023034 005520          ADC      (RO)+      ;ADD 'C' BIT TO 1$ ABOVE
4615 023036 006340          ASL      -(RO)      ;SHIFT 1$
4616 023040 102375          BVC     2$         ;UNTIL 'V' BIT SETS
4617 023042 022767 077776 177754  CMP      #077776,1$ ;CHECK RESULT
4618 023050 001401          BEQ     .+4
4619 023052 104000          HLT
                                ;ERROR! INCORRECT RESULT IN 1$ ABOVE
                                ;RO=ADDRESS OF DATA
4620
4621
4622          ;CHECK THAT CONDITION CODES ARE SET PROPERLY WHEN A NUMBER (CURRENT PC)
4623          ;AND THAT NUMBER +1 ARE COMPARED, AND VICE VERSA.
4624 023054 010700  CMPN:  MOV      PC,RO      ;GET CURRENT PC
4625 023056 010002          MOV      RO,R2      ;SAVE IN R2
4626 023060 005202          INC      R2         ;MAKE R2 = RO+1
4627 023062 000277          SCC
4628 023064 000251          +CLC!CLN          ;CLEAR C & N BITS
4629 023066 020002          CMP      RO,R2      ;COMPARE # WITH #+1
4630 023070 103003          BCC     1$         ;CARRY BIT SHOULD SET
4631 023072 102402          BVS     1$         ;V BIT SHOULD CLEAR
4632 023074 001401          BEQ     1$         ;Z BIT SHOULD CLEAR
4633 023076 100401          BMI     .+4
4634 023100 104000          HLT          ;N BIT SHOULD SET
4635          ;ERROR! COMPARE # WITH #+1 FAILED TO
4636          ;SET CONDITION CODES IN PSW CORRECTLY
4637 023102 000277          SCC          ;SET CONDITION CODES IN PSW
4638 023104 120200          CMPB    R2,RO      ;COMPARE #+1 WITH #
4639 023106 103403          BCS     2$         ;C BIT SHOULD CLEAR
4640 023110 102402          BVS     2$         ;V BIT SHOULD CLEAR
4641 023112 001401          BEQ     2$         ;Z BIT SHOULD CLEAR
4642 023114 100001          BPL     .+4
4643 023116 104000          HLT          ;N BIT SHOULD CLEAR
4644          ;ERROR! COMPARE #+1 WITH # FAILED TO SET
4645          ;CONDITION CODES IN PSW CORRECTLY
4646          ;ENSURE PRIORITY 0
4647 023120 105037 177776  CLRB    2#PSW
4648 023124 000004  RELE4:  SCOPE
4649 023130 062702 000012  MOV      PC,R2
4650 023134 012707 034242  ADD      #12,R2
4651 023140 000000  MOV      #RELOC,PC ;GO RELOCATE PROGRAM CODE
4652          REL44: .WORD 0
4653          ;4444444444444444 LAST ADDRESS OF CODE TO BE RELOCATED 444444444444
4654          ;*****
4655          ;*TEST 46 CHECK EXTENDED INSTRUCTION SET
4656          ;*****
4656 023142 112737 000046 001202  †ST46:  MOVB    #46,2#STSTNM ;LOAD TEST NUMBER
4657 023150 012767 000001 156140  MOV      #1,$TIMES ;;DO 1 ITERATION
4658 023156 000004          SCOPE
4659
4660          .SBTTL START OF SECTION 5
4661          ;5555555555555555 FIRST ADDRESS TO BE RELOCATED 5555555555
4662 023160 010700  RELS:  MOV      PC,RO      ;GET PC
4663 023162 005740          TST     -(RO)      ;RO CONTAINS THE ADDRESS OF RELS
4664 023164 010037 001512  MOV      RO,2#FRSTAD ;SAVE
4665 023170 010700          MOV      PC,RO      ;GET CURRENT PC
4666 023172 162700 023172  SUB      #.,RO      ;SUBTRACT RELOCATION FACTOR

```

4668	023202	010737	001212		MOV	PC, 0#SLPERR	; SET LOOP ADDRESS
4669	023206	062737	000030	001212	ADD	#30, 0#SLPERR	; ADJUST
4670	023214	013737	001212	001210	MOV	0#SLPERP, 0#SLPADR	
4671	023222	105737	001502		TSTB	0#NEXEC	; BR IF TEST CODE TO BE EXECUTED
4672	023226	001402			BEQ	+6	
4673	023230	000167	001454		JMP	RELES	
4674	023234	005000			EXTINST: CLR	RO	
4675	023236	000277			SCC		; PRESET CC'S
4676	023240	005700			SXT	RO	; EXTEND SIGN (1) INTO RO
4677	023242	103005			BCC	SXTO	; CHECK RESULT CC'S
4678	023244	102404			BVS	SXTO	
4679	023246	001403			BEQ	SXTO	
4680	023250	100002			BPL	SXTO	
4681	023252	005200			INC	RO	; CHECK RESULT
4682	023254	001401			BEQ	+4	
4683	023256	104000			SXTO:	HLT	
4684							
4685	023260	010700			MOV	PC, RO	
4686	023262	010002			MOV	RO, R2	
4687	023264	012703	177777		MOV	#-1, R3	
4688	023270	005102			COM	R2	
4689	023272	000243			+CLV:CLC		; CLEAR C AND V BITS
4690	023274	074003			XOR	RO, R3	; R3 SHOULD CONTAIN COMPLEMENT OF RO
4691	023276	103404			BCS	XORO	; CHECK THAT C WAS NOT AFFECTED
4692	023300	102403			BVS	XORO	; AND THAT V WAS CLEARED
4693	023302	001402			BEQ	XORO	
4694	023304	020203			CMP	R2, R3	; CHECK RESULT
4695	023306	001401			BEQ	+4	
4696	023310	104000			XORO:	HLT	; ERROR! XOR FAILED
4697							
4698	023312	010700			MOV	PC, RO	
4699	023314	022020			CMP	(RO)+, (RO)+	; SET ADDRESS REGISTER
4700	023316	000401			BR	1\$	; RESERVE WORD FOR TEST DATA
4701	023320	000000			.WORD	0	; CONTAINS TEST DATA
4702	023322	005700			1\$: TST	RO	; EXTEND SIGN OF ADDRESS INTO
4703	023324	006710			SXT	(RO)	; ADDRESS (RO)=-1 IF MSB RO=1
4704	023326	005002			CLR	R2	; OTHERWISE (RO)=0
4705	023330	005700			TST	RO	; CHECK SIGN OF ADDRESS
4706	023332	100001			BPL	+4	
4707	023334	005102			COM	R2	; COMPLEMENT CHECK REG IF NEG
4708	023336	021002			CMP	(RO), R2	; CHECK RESULT OF SXT
4709	023340	001401			BEQ	+4	
4710	023342	104000			SXT1:	HLT	; ERROR! SXT FAILED TO EXTEND SIGN PROPERLY
4711							
4712	023344	012710	100000		MOV	#100000, (RO)	; PRESET DATA
4713	023350	011002			MOV	(RO), R2	
4714	023352	000277			SCC		; PRESET CC'S
4715	023354	074210			XOR	R2, (RO)	; XOR 100000 WITH 100000 RESULT = 0
4716	023356	103007			BCC	XOR1	; CHECK CC'S AFTER XOR
4717	023360	102406			BVS	XOR1	
4718	023362	001005			BNE	XOR1	
4719	023364	100404			BMI	XOR1	
4720	023366	005710			TST	(RO)	; CHECK RESULT (0)
4721	023370	001002			BNE	XOR1	
4722	023372	005402			NEG	R2	; CHECK THAT REG WAS NOT AFFECTED

4724	023376	104000		XOR1:	HLT		
4725							
4726	023400	010702			MOV	PC,R2	
4727	023402	022222			CMP	(R2)+,(R2)+	
4728	023404	000401			BR	SXT4	;PRESERVE WORD FOR DATA
4729	023406	000000			.WORD	0	;RESERVED FOR DATA
4730	023410	012722	125252	SXT4:	MOV	#125252,(R2)+	;PRESET DATA
4731	023414	005742			SXT	-(R2)	;EXTEND SIGN
4732	023416	074722			XOR	PC,(R2)+	
4733	023420	010700			MOV	PC,R0	;GET PC
4734	023422	005740			TST	-(R0)	;SUBTRACT 2 FROM PC
4735	023424	005100			COM	R0	;R0=RESULT OF XOR PC-1 ABOVE
4736	023426	074042			XOR	R0,-(R2)	;CHECK RESULT OF SXT AND XOR ABOVE
4737	023430	001401			BEQ	+.4	
4738	023432	104000		XOR24:	HLT		;ERROR! SXT & XOR ABOVE INCORRECT
4739							
4740	023434	012704	000001		MOV	#1,R4	;SET R4
4741	023440	005767	000063		SXT	XOR6A	;PRESET DATA=0
4742	023444	074467	000054	2\$:	XOR	R4,XOR6A	
4743	023450	100423			BMI	XOR6	
4744	023452	005304			ASL	R4	;SHIFT R4
4745	023454	102373			BVC	2\$	;UNTIL V SETS (R4=100000)
4746	023456	100020			BPL	XOR6	;BRANCH IF 'N' IS CLEAR
4747	023460	074467	000040		XOR	R4,XOR6A	;XOR6A=177777
4748	023464	100015			BPL	XOR6	
4749	023466	074767	000032		XOR	PC,XOR6A	;XOR PC WITH XOR6A (177777)
4750	023472	010767	000030		MOV	PC,XOR6B	;FORM PC AS USED IN XOR ABOVE
4751	023476	162767	000004	000022	SUB	#4,XOR6B	
4752	023504	005167	000016		COM	XOR6B	
4753	023510	026767	000012	000006	CMP	XOR6B,XOR6A	;XOR6A SHOULD = COMPLEMENT OF PC
4754	023516	001401			BEQ	+.4	
4755	023520	104000		XOR6:	HLT		;ERROR! XOR TESTS ABOVE FAILED
4756							
4757	023522	000402			BR	+.6	
4758							
4759	023524	000000		XOR6A:	.WORD	0	;CONTAINS DATA USED BY TEST ABOVE
4760	023526	000000		XOR6B:	.WORD	0	
4761							
4762							
4763	023530	012700	077777		MOV	#077777,R0	;SET SOURCE OPERAND FOR ADD
4764	023534	006767	177764		SXT	XOR6A	;CLEAR XOR6A
4765	023540	001004			BNE	SXT6	;CHECK CC'S AFTER EXTENDING ZERO'S
4766	023542	100403			BMI	SXT6	
4767	023544	103402			BCS	SXT6	
4768	023546	102401			BVS	SXT6	
4769	023550	000401			BR	+.4	
4770	023552	104000		SXT6:	HLT		;ERROR! SXT FAILED
4771							
4772	023554	012702	000001		MOV	#1,R2	;SET DEST OPERAND FOR ADD
4773	023560	013703	001506		MOV	J#FACTOR,R3	;LOAD INDEX REGISTER
4774	023564	060002			ADD	R0,R2	;RESULT OF ADD=100000
4775	023566	006763	023524		SXT	XOR6A(3)	;EXTEND SIGN OF ADD ABOVE
4776	023572	001403			BEQ	SXT6A	
4777	023574	005267	177724		INC	XOR6A	;CHECK RESULT OF SXT
4778	023600	001401			BEQ	+.4	



```

4780                                     ;SIGN
4781 023604 010703 MOV PC,R3
4782 023606 000402 BR 0+6 ;PRESERVE 2 WORDS FOR DATA
4783 023610 000000 SXRA: .WORD 0 ;RESERVED WORD FOR DATA
4784 023612 000000 SXRB: .WORD 0 ;RESERVED WORD FOR DATA
4785 023614 005723 TST (R3)+
4786 023616 010304 MOV R3,R4 ;R3 = ADDRESS OF SXRA
4787 023620 000250 CLN ;CLEAR N BIT
4788 023622 006724 SXT (R4)+ ;EXTEND ZEROS INTO SXRA
4789 023624 001401 BEQ .+4
4790 023626 104000 SXT2: HLT ;ERROR! SXT FAILED
4791
4792 023630 010467 177754 MOV R4,SXRA ;SXRA = ADDRESS OF SXRB
4793 023634 000257 CCC ;CLEAR CONDITION CODES
4794 023636 006733 SXT 2(R3)+ ;EXTEND ZEROS INTO SXRB
4795 023640 001401 BEQ .+4
4796 023642 104000 SXT3: HLT ;ERROR!
4797
4798 023644 000270 SEN ;SET N BIT
4799 023646 006753 SXT 2-(R3) ;EXTEND ONES INTO SXRB
4800 023650 100401 BMI .+4
4801 023652 104000 SXT5: HLT ;ERROR!
4802
4803 023654 012704 025252 MOV #025252,R4 ;R4 = 025252
4804 023660 074433 XOR R4,2(R3)+ ;SXRB = 152525 (COMPLEMENT OF R4)
4805 023662 005002 CLR R2
4806 023664 074253 XOR R2,2-(R3) ;SXRB REMAINS UNCHANGED
4807 023666 001405 BEQ XOR35 ;CHECK CONDITION CODES
4808 023670 100004 BPL XOR35
4809 023672 005104 COM R4 ;R4 = 152525
4810 023674 020467 177712 CMP R4,SXRB ;CHECK XOR
4811 023700 001401 BEQ .+4
4812 023702 104000 XOR35: HLT ;ERROR! XOR FAILED
4813
4814 023704 005743 TST -(R3) ;R3 = ADDRESS OF SXRA-2
4815 023706 000250 CLN ;CLEAR N BIT
4816 023710 006773 000002 SXT 2(R3) ;SXRB = 0
4817 023714 001401 BEQ .+4
4818 023716 104000 SXT7: HLT ;ERROR! SXT FAILED
4819
4820 023720 074473 000002 XOR R4,2(R3) ;SXRB = R4
4821 023724 020473 000002 CMP R4,2(R3) ;CHECK XOR
4822 023730 001401 BEQ .+4
4823 023732 104000 XOR7: HLT ;ERROR! XOR FAILED
4824
4825 ;*****
4826 ;*TEST 47 SOB TEST
4827 ;* NOTE: DO NOT INSERT ANY CODE IN FOLLOWING SOB TESTS
4828 ;* SINCE IT TESTS THE MAXIMUM BRANCH WIDTH OF THE INSTRUCTION.
4829 ;*****
4829 023734 112737 000047 001202 ST47: MOVB #47,2#STSTNM ;LOAD TEST NUMBER
4830 023742 000004
4831

```

4833	023746	000407		BR	SOB0	;BRANCH TO SOB TEST
4834						
4835	023750	005004		SOB10:	CLR R4	;R4 = 0
4836	023752	005705			TST R5	;CHECK ERROR INDICATOR
4837	023754	001401			BEQ .+4	;SOB BRANCHED CORRECTLY
4838	023756	104000			HLT	;ERROR!
4839						
4840	023760	005005		SOB9:	CLR R5	;CLEAR INDICATOR (R5)
4841	023762	006004			ROR R4	;ROTATE RIGHT R4
4842	023764	000467			BR SOB8	
4843						
4844	023766	012700	000010	SOB0:	MOV #10,R0	;R0=10
4845	023772	000277			SCC	;SET CONDITION CODES
4846	023774	001012		SOB1:	BNE SOB2	;CHECK CONDITION CODES AFTER SOB
4847	023776	100011			BPL SOB2	;SOB SHOULD NOT EFFECT THE
4848	024000	102010			BVC SOB2	;CONDITION CODES.
4849	024002	103007			BCC SOB2	
4850	024004	077005			SOB RO,SOB1	
4851	024006	001005			BNE SOB2	;CHECK CONDITION CODES AFTER
4852	024010	100004			BPL SOB2	;SOB FALLS THROUGH
4853	024012	102003			BVC SOB2	;SOB SHOULD NOT EFFECT
4854	024014	103002			BCC SOB2	;CONDITION CODES.
4855	024016	005700			TST R0	;CHECK IF R0=0
4856	024020	001401			BEQ .+4	
4857	024022	104000		SOB2:	HLT	;ERROR!
4858						
4859	024024	012702	000100		MOV #100,R2	;R2=100
4860	024030	012700	000101		MOV #101,R0	;SET CHECK REGISTER, R0=101
4861	024034	001414		SOB3:	BEQ SOB4	;CHECK CONDITION CODES AFTER
4862	024036	100413			BMI SOB4	;SOB BRANCH,
4863	024040	102412			BVS SOB4	;SOB SHOULD NOT EFFECT
4864	024042	103411			BCS SOB4	;CONDITION CODES.
4865	024044	005300			DEC R0	;DECREMENT CHECK REGISTER
4866	024046	020002			CMP R0,R2	;CHECK THAT SOB DECREMENTS
4867	024050	001006			BNE SOB4	
4868	024052	000257			CCC	;SET CONDITION CODES BEFORE SOB
4869	024054	077211			SOB R2,SOB3	;BRANCH TO SOB3 UNTIL R2=0
4870	024056	001403			BEQ SOB4	;CHECK CONDITION CODES AFTER
4871	024060	100402			BMI SOB4	;SOB FALLS THROUGH
4872	024062	005702			TST R2	;CHECK IF R2=0
4873	024064	001401			BEQ .+4	
4874	024066	104000		SOB4:	HLT	;ERROR!
4875						
4876	024070	012700	000001	SOB5:	MOV #1,R0	;R0=1
4877	024074	000401			BR .+4	
4878	024076	104000			HLT	;ERROR!
4879	024100	077002			SOB R0,-2	;SOB SHOULD NOT BRANCH
4880						
4881	024102	005700			TST R0	;CHECK IF R0=0 AFTER SOB
4882	024104	001401			BEQ .+4	
4883	024106	104000			HLT	;ERROR!
4884						
4885	024110	012704	100000	SOB5A:	MOV #100000,R4	;R4=100000
4886	024114	000403			BR 1\$	
4887	024116	005204		3\$:	INC R4	;R4=100000

```

4889 024122 104000 HLT ;ERROR! SOB DID NOT
4890 ;INCREMENT PROPERLY
4891
4892 024124 077404 15: SOB R4,35 ;SOB SHOULD BRANCH
4893 024126 104000 HLT ;ERROR! SOB DID NOT BRANCH
4894
4895 024130 012703 000100 25: MOV #100,R3 ;R3=100
4896 024134 077301 SOB6: SOB R3,S0B6 ;USE SOB TO BRANCH TO ITSELF
4897 024136 005703 TST R3 ;CHECK IF R3=0
4898 024140 001703 BEQ SOB10
4899 024142 104000 SOB7: HLT ;ERROR!
4900
4901 024144 005705 SOB8: TST R5 ;CHECK INDICATOR (R5)
4902 ;IF SOB BRANCHES INCORRECTLY
4903 ;WHEN CHECKING MAX. BRANCH,
4904 ;R5 WILL NOT BE CLEARED AT
4905 ;THIS POINT INDICATING AN ERROR.
4906
4907 024146 001401 BEQ .+4 ;BRANCH IF SOB BRANCHES CORRECTLY
4908 024150 104000 HLT ;ERROR!
4909
4910 024152 005205 INC R5 ;SET INDICATOR (R5)
4911 024154 077477 SOB R4,S0B9 ;TEST MAX. BRANCH OF SOB
4912 024156 005704 TST R4 ;CHECK IF R4=0
4913 024160 001401 BEQ .+4
4914 024162 104000 HLT ;ERROR!
4915
4916 ;*****
4917 ;*TEST SO CHECK THE MARK INSTRUCTION
4918 ;*****
4918 024164 112737 000050 001202 †TSTSO: MOV #50,2#STSTNM ;LOAD TEST NUMBER
4919 024172 000004 SCOPE
4920 024174 010602 MRKTST: MOV SP,R2
4921 024176 010705 MOV PC,R5
4922 024200 010500 MOV R5,R0 ;THE STACK LOOKS LIKE THIS AFTER
4923 024202 010546 MOV R5,-(SP) ;THE JSR INSTRUCTION
4924 024204 010746 MOV PC,-(SP) ; -2(SP)=R0 THIS IS A
4925 024206 010746 MOV PC,-(SP) ; -4(SP)=PC STRING
4926 024210 010746 MOV PC,-(SP) ; -6(SP)=PC+2 OF
4927 024212 010746 MOV PC,-(SP) ; -10(SP)=PC+4 FIVE
4928 024214 010746 MOV PC,-(SP) ; -12(SP)=PC+6 DUMMY
4929 024216 012746 006405 MOV #MARK+5,-(SP) ; -14(SP)=PC+10 ARGUMENTS
4930 024222 010605 SP,R5 ; -16(SP)=MARK 5
4931 024224 004767 000002 JSR PC,MARK1 ; -20(SP)=PC PUSHED BY JSR
4932 024230 000403 BR +10
4933 024232 000205 MARK1: RTS R5
4934 024234 104000 HLT ;ERROR! SHOULD BE DOING MARK 5 INST.
4935 024236 000407 BR MARKEX
4936 024240 020602 CMP SP,R2
4937 024242 001402 BEQ .+6
4938 024244 104000 HLT ;ERROR! SP NOT RETURNED TO PROPER
4939 024246 000403 BR MARKEX ;VALUE BY MARK INSTRUCTION
4940 024250 020005 CMP R0,R5
4941 024252 001401 BEQ .+4
4942 024254 104000 HLT ;ERROR! DID NOT RESTORE R5 FROM STACK
4943 024256 010206 MARKEX: MOV R2,SP ;RESTORE SP

```

```

4945 : *TEST 51 RTT/RTI TEST
4946 : * RTT/RTI TEST INSURES THAT CP DOES THE INSTRUCTION FOLLOWING
4947 : * AN RTT IF THE "T"BIT IS SET IN THE PSW,BUT DOES HONOR
4948 : * THE TRAP IMMEDIATELY IF IT EXECUTES AN RTI
4949 : * INSTRUCTION SEQUENCE-RTT
4950 : * 2$: RTT ; NO 'T' TRAP AFTER RTT
4951 : * INC RO ; RO=000001
4952 : * ; 'T' TRAP TO 5$ AFTER INC
4953 : * 5$: COM RO ; RO=177776
4954 : * MOV SAVPSW,2(SP) ; CLEAR 'T' BIT IN RETURN PSW
4955 : * RTI ; RETURN TO INSTRUCTION FOLLOWING INC
4956 : * CMP #RTT,2$ ; CHECK
4957 : * ETC
4958 : *
4959 : * INSTRUCTION SQUENCE-RTI
4960 : * 2$: RTI ; 'T' TRAP AFTER RTI
4961 : * 5$: COM RO ; RO=177777
4962 : * MOV SAVPSW,2(SP) ; CLEAR 'T' BIT IN RETURN PSW
4963 : * RTI ; RETURN TO INC INSTRUCTION
4964 : * INC RO ; RO=000000
4965 : * CMP #RTT,2$ ; CHECK
4966 : * ETC
4967 : *
4968 024260 112737 000051 001202 T51: MOVB #51,#STSTNM ;LOAD TEST NUMBER
4969 024266 000004 SCOPE
4970 024270 013767 177776 000202 RTT1: MOV #PSW,SAVPSW ;SAVE PSW
4971 024276 032767 000020 000174 BIT #20,SAVPSW ;CHECK IF "T"BIT SET
4972 024304 001176 BNE RTT2EX ;BRANCH TO EXIT
4973 024306 010746 1$: MOV PC, -(SP) ;GET CURRENT PC
4974 024310 062716 000116 ADD #5$-.,(SP) ;FORM RELOCATED PC
4975 024314 012637 000014 MOV (SP)+,#TBITVEC ;LOAD INTO TRAP VECTOR
4976 024320 016746 000154 MOV SAVPSW, -(SP) ;GET CURRENT PSW
4977 024324 011637 000016 MOV (SP),#TBITVEC+2
4978 024330 052737 000340 177776 BIS #PR7,#PSW ;SET PRIORITY LEVEL 7
4979 024336 005000 CLR RO
4980 024340 052716 000360 BIS #PR7+20,(SP) ;SET "T"BIT IN PSW ON STACK
4981 024344 010746 MOV PC, -(SP) ;PUT THE PC ON THE STACK
4982 024346 062716 000006 ADD #6,(SP) ;ADJUST PC FOR NEXT INSTRUCTION
4983 024352 000006 2$: RTT
4984 024354 005200 INC RO ;DONE TO SEE IF INSTR. FOLLOWING
4985 : RTT IS EXECUTED IF T-BIT SET
4986 024356 042737 000340 177776 BIC #PR7,#PSW ;SET PRIORITY LEVEL 0
4987 024364 022767 000006 177760 CMP #RTT,2$
4988 024372 001005 BNE 3$ ;CHECK IF INC WAS EXECUTED
4989 024374 022700 177776 CMP #177776,RO ;CHECK IF COM-RO EXECUTED
4990 024400 001406 BEQ 4$
4991 024402 104000 HLT ;ERROR!RO NOT COMPLIMENTED
4992 024404 000415 BR 6$ ;EXIT TEST
4993 024406 005700 3$: TST RO ;TEST IF TRAPED BEFORE INC INST.
4994 : WAS EXECUTED
4995 024410 001413 BEQ 6$
4996 024412 104000 HLT ;ERROR!
4997 024414 000411 BR 6$ ;EXIT TEST
4998 024416 012767 000002 177726 4$: MOV #RTI,2$
4999 024424 000730 BR 1$

```

```

5001 024430 016766 000044 000002      MOV      SAVPSW,2(SP)
5002 024436 000002                    RTI
5003 024440 012767 000006 177704 6S:    MOV      #RTT,2S
5004 024446 012737 001472 000014      MOV      #SATAN,2#TBITVEC          ;RESTORE 'T' TRAP VECTOR
5005 024454 005037 000016      CLR      2#TBITVEC+2
5006 024460 042737 000360 000016      BIC      #PR7+BIT4,2#TBITVEC+2
5007 024466
5008
5009
5010
5011 024466 112737 000052 001202  TST52:  MOVB   #52,2#STSTNM          ;LOAD TEST NUMBER
5012 024474 000004                    SCOPE
5013 024476 000401                    BR       RTT2A
5014 024500 000000      SAVPSW: .WORD 0
5015 024502 016700 177772      RTT2A:  MOV   SAVPSW,R0          ;GET SAVED PSW
5016 024506 105000                    CLR     R0          ;CLEAR PRIORITY LEVEL,T, AND COND CODES
5017 024510 012702 144000      MOV   #UM+REG,R2
5018 024514 074002                    XOR   R0,R2
5019 024516 001435                    BEQ   2S          ;USER MODE REG. SET #1 ON
5020 024520 012702 044000      MOV   #SM+REG,R2
5021 024524 074002                    XOR   R0,R2
5022 024526 001447                    BEQ   3S          ;SUPER MODE REG. SET #1 ON
5023 024530 032700 140000      BIT   #UM,R0
5024 024534 001062                    BNE   RTT2EX
5025
5026
5027 024536 012702 177777      ;TEST THAT RTT CLEARS BITS 11,12,13 & PRIORITY LEVEL BITS IN KERNEL MODE
5028 024542 012737 034240 177776      MOV   #-1,R2          ;KERNEL MOCE REG. SET 0 ON
5029 024550 005002                    MOV   #PUM+REG+PRS,2#PSW          ;SELECT REG. SET #1
5030 024552 012746 000100      CLR   R12          ;SHOULD CLEAR REG #12
5031 024556 010746                    MOV   #PR2,-(SP)
5032 024560 062716 000006      MOV   PC,-(SP)
5033 024564 000006      ADD   #1$-.,(SP)          ;FORM NEW PC
5034 024566 013700 177776      RTT
5035 024572 005702 1S:    MOV   2#PSW,R0          ;NOW USING REG SET 0
5036 024574 001001                    TST   R2          ;SHOULD TEST R2 NOT R12
5037 024576 104000                    BNE   4S
5038 024600 022700 000100 4S:    HLT
5039 024604 001436                    CMP   #PR2,R0          ;TESTS THE PSW AFTER THE RTT
5040 024606 104000                    BEQ   RTT2EX
5041 024610 000434                    HLT          ;ERROR! INCORRECT PSW AFTER THE RTT
5042
5043
5044 024612 052737 030340 177776      ;TEST TO INSURE THAT RTI DOES NOT CLEAR BITS 11-15 IN USER MODE
5045 024620 005046 2S:    BIS   #PUM+PR7,2#PSW          ;PSW<15-5>=144X
5046 024622 010746                    CLR   -(SP)
5047 024624 062716 000006      MOV   PC,-(SP)
5048 024630 000002      ADD   #5$-.,(SP)
5049 024632 022737 174340 177776 5S:    RTI          ;ATTEMPS TO INSERT A PSW OF 0
5050 024640 001420                    CMP   #UM+PUM+REG+PR7,2#PSW          ;SHOULD CHECK AGAINST REG #0
5051 024642 104000                    BEQ   RTT2EX
5052 024644 000416                    HLT          ;ERROR! RTI CLEARED BITS IN PSW
5053
5054
5055 024646 052737 030200 177776 3S:    ;TEST THAT BITS 11-15 AND PRIORITY BITS ARE NOT ALTERED IN SUPER MODE
5056

```

```

5057 024660 010746          MOV    PC, -(SP)
5058 024662 062716 000006    ADD    #6$, -(SP)
5059 024666 000006          RTT
; ATTEMPTS TO CLEAR 11-15 AND ALTER PR
5060
5061 024670 022737 074200 177776 6$:  CMP    #SM+PUM+REG+PR4, 2#PSW
5062 024676 001401          BEQ    RTT2EX
5063 024700 104000          HLT
; ERROR! RTT ALTERED PR IN
; SUPER MODE OR BITS 11-15.
5064
5065 024702 016737 177572 177776 RTT2EX: MOV    SAVPSW, 2#PSW
5066 024710 000004          RELES: SCOPE
5067 024712 010702          MOV    PC, R2
5068 024714 062702 000012    ADD    #12, R2
5069 024720 012707 034242    MOV    #RELOC, PC ; GO RELOCATE PROGRAM CODE
5070 024724 000000          RELSS: .WORD 0
; 5555555555555555 LAST ADDRESS OF CODE TO BE RELOCATED 5555555555
5071
5072
5073
5074
5075
5076 024726 112737 000053 001202 †ST53: MOVB  #53, 2#STSTNM ; LOAD TEST NUMBER
5077 024734 012767 000001 154354    MOV    #1, $TIMES ; DO 1 ITERATION
5078 024742 000004          SCOPE
5079
5080
5081
5082 024744 010700          .SBTTL START OF SECTION 6
5083 024746 005740          :6666666666666666 FIRST ADDRESS TO BE RELOCATED 6666666666
5084 024750 010037 001512    REL6: MOV    PC, R0 ; GET PC
5085 024754 010700          TST    -(R0) ; R0 CONTAINS THE ADDRESS OF REL6
5086 024756 162700 024756    MOV    R0, 2#FRSTAD ; SAVE
5087 024762 010037 001506    MOV    PC, R0 ; GET CURRENT PC
5088 024766 010737 001212    SUB    #, R0 ; SUBTRACT RELOCATION FACTOR
5089 024772 062737 000030 001212    MOV    R0, 2#FACTOR ; SAVE RELOCATION FACTOR
5090 025000 013737 001212 001210    MOV    PC, 2#SLPERR ; SET LOOP ADDRESS
5091 025006 105737 001502    ADD    #30, 2#SLPERR ; ADJUST
5092 025012 001402          TSTB  2#NEXEC ; BR IF TEST CODE TO BE EXECUTED
5093 025014 000167 002016    BEQ    +6
5094 025020 012700 000001    JMP    REL6
5095 025024 012703 000021    ASHL0: MOV   #1, R0 ; R0 WILL BE THE SHIFT COUNT
5096 025030 005067 000014    1$:  MOV   #17, R3 ; MAX SHIFT COUNT
5097 025034 010002          CLR    2$ ; PRESET SAVED CC'S LOCATION=0
5098 025036 010705          MOV   R0, R2 ; GET SHIFT COUNT FOR PASS
5099 025040 010504          MOV   PC, R5 ; R5 & R4 WILL BE DATA SHIFTED BY
5100 025042 072502          MOV   R5, R4 ; ASH & ASL INSTRUCTIONS
5101 025044 113727 177776    ASH   R2, R5 ; SHIFT R5
5102 025050 000000 2$:  MOVB  2#PSW, (PC)+ ; SAVE CC'S
; CONTAINS ASH CC'S IN EVEN BYTE
; ASL CC'S IN ODD BYTE
5103
5104 025052 006304 3$:  ASL   R4 ; SHIFT R4
5105 025054 113746 177776    MOVB  2#PSW, -(SP) ; SAVE PSW ON STACK
5106 025060 132716 000002    BITB  #2, (SP) ; CHECK IF ASL SET V BIT
5107 025064 001403          BEQ    30$
5108 025066 152767 000002 177755    BISB  #2, 2$+1 ; IF ASL SET V THEN SET V IN 2$+1
5109 025074 112637 177776 30$: MOVB  (SP)+, 2#PSW ; RESTORE ORIGINAL PSW
5110 025100 077214          SOB   R2, 3$ ; SHIFT R4 R2 TIMES
5111 025102 153767 177776 177741    BISB  2#PSW, 2$+1 ; SAVE CC'S AFTER ASL

```

S113	025112	001004			BNE	4\$		
S114	025114	126767	177730	177727	CMPB	2\$ 2\$+1		; CHECK ASH & ASL CC'S
S115	025122	001401			BEQ	.+4		
S116	025124	104000			4\$: HLT			; ERROR! INCORRECT RESULT OR CC'S
S117	025126	005200			INC	R0		; INCREMENT PASS SHIFT COUNT
S118	025130	020003			CMP	R0, R3		
S119	025132	001336			BNE	1\$		
S120								
S121	025134	012700	177777		ASHRO: MOV	#-1, R0		; RO = RIGHT SHIFT COUNT FOR PASS
S122	025140	012703	177757		MOV	#-1, R3		; MAX SHIFT COUNT
S123	025144	010002			1\$: MOV	R0, R2		; GET SHIFT COUNT FOR PASS
S124	025146	010705			MOV	PC, R5		; R5 & R4 = DATA TO BE SHIFTED
S125	025150	010504			MOV	R5, R4		; BY ASH & ASR INSTRUCTIONS
S126	025152	072502			ASH	R2, R5		; SHIFT R5 R2 TIMES
S127	025154	113727	177776		MOVW	2\$PSW, (PC)+		; SAVE CC'S IN EVEN BYTE
S128	025160	000000			2\$: .WORD	0		; CONTAINS ASH CC'S IN EVEN BYTE
S129								; ASR CC'S IN ODD BYTE
S130	025162	005402			NEG	R2		
S131	025164	006204			3\$: ASR	R4		; SHIFT R4
S132	025166	077202			SOB	R2, 3\$		; SHIFT R4 R2 TIMES
S133	025170	113767	177776	177763	MOVW	2\$PSW, 2\$+1		; SAVE CC'S AFTER ASR
S134	025176	142767	000002	177755	BICB	2\$ 2\$+1		; ASH RIGHT WILL NOT SET V ASR MAY SET V
S135	025204	020504			CMP	R5, R4		; CHECK ASH & ASR RESULTS
S136	025206	001004			BNE	4\$		
S137	025210	126767	177744	177743	CMPB	2\$ 2\$+1		; CHECK ASH & ASR CC'S
S138	025216	001401			BEQ	.+4		
S139	025220	104000			4\$: HLT			; DECREMENT PASS SHIFT COUNT
S140	025222	005300			DEC	R0		
S141	025224	020003			CMP	R0, R3		
S142	025226	001346			BNE	1\$		
S143								
S144	025230	012746	000037		ASHCLO: MOV	#31, -(SP)		; PUT MAX SHIFT COUNT ON STACK
S145	025234	012746	000001		MOV	#1, -(SP)		; PUT LEFT SHIFT COUNT ON STACK
S146	025240	011600			1\$: MOV	(SP), R0		; GET PASS SHIFT COUNT
S147	025242	010705			MOV	PC, R5		; CURRENT PC IS DATA TO BE SHIFTED
S148	025244	010503			MOV	R5, R3		; ASHC SHIFTS R4, R5; ASL, ROL SHIFTS R2, R3
S149	025246	005004			CLR	R4		
S150	025250	005002			CLR	R2		
S151	025252	073400			ASHC	R0, R4		; SHIFT R4 LEFT AS SPECIFIED BY R0
S152	025254	006303			2\$: ASL	R3		; SHIFT R2, R3 LEFT
S153	025256	006102			ROL	R2		; AS SPECIFIED BY R0
S154	025260	077003			SOB	R0, 2\$		
S155	025262	020402			CMP	R4, R2		; CHECK RESULTS
S156	025264	001002			BNE	3\$		
S157	025266	020503			CMP	R5, R3		
S158	025270	001401			BEQ	.+4		
S159	025272	104000			3\$: HLT			; INCREMENT NEXT PASS SHIFT COUNT
S160	025274	005216			INC	(SP)		; REACHED MAX COUNT (31.)
S161	025276	021666	000002		CMP	(SP), 2(SP)		
S162	025302	001356			BNE	1\$		
S163	025304	022626			CMP	(SP)+, (SP)+		; RESTORE STACK PTR
S164								
S165	025306	012746	177740		ASHCRO: MOV	#-32, -(SP)		; PUT MAX RIGHT SHIFT COUNT ON STACK
S166	025312	012746	177777		MOV	#-1, -(SP)		; PUT PASS SHIFT COUNT ON STACK
S167	025316	011600			1\$: MOV	(SP), R0		; GET PASS SHIFT COUNT

```

5169 025322 010204      MOV      R2,R4      ;TO BE SHIFTED BY TEST
5170 025324 005003      CLR      R3
5171 025326 005005      CLR      R5
5172 025328 000265      SEV
5173 025330 073200      ASHC     R0,R2      ;SET V BIT IN PSW
5174 025332 102410      BVS     J$         ;SHIFT R2,R3 RIGHT R0 TIMES
5175 025334 005400      NEG     R0         ;SHIFT RIGHT CLEARS V
5176 025336 006204      ASR     R4         ;NEGATE SHIFT COUNT FOR SOB
5177 025338 006005      ROR     R5         ;SHIFT R4,R5 RIGHT R0 TIMES
5178 025340 077003      SOB     R0,R2$
5179 025342 020204      CMP     R2,R4      ;CHECK RESULT
5180 025344 001002      BNE     J$
5181 025346 020305      CMP     R3,R5
5182 025348 001401      BEQ     .+4
5183 025350 104000      HLT
5184 025360 005316      DEC     (SP)       ;SET SHIFT COUNT FOR NEXT PASS
5185 025362 021666 000002      CMP     (SP),2(SP) ;CHECK IF MAX SHIFT COUNT
5186 025364 001353      BNE     1$
5187 025370 022626      CMP     (SP)+,(SP)+ ;RESTORE STACK PTR
5188
5189
5190
5191
5192
5193
5194 025372 112737 000054 001202 1$T54: MOVB  #54,2$STSTM ;LOAD TEST NUMBER
5195 025400 000004      SCOPE
5196 025402 012700 000001      MULO:  MOV  #1,R0      ;R0 CONTAINS MULTIPLIER FOR MUL
5197 025406 012706 000700      MOV  #SUPSTK,SP     ;SETUP THE STACK
5198 025412 005016      CLR  (SP)           ;(SP) CONTAINS SHIFT VALUE FOR ASHC
5199 025414 010702      MOV  PC,R2         ;R3,R2 & R5,R4 ARE DATA REGISTERS
5200 025416 010227      MOV  R2,(PC)+     ;SAVE MULTIPICAND
5201 025420 000000      .WORD 0           ;CONTAINS ORIGINAL MULTIPICAND
5202 025422 005003      CLR  R3
5203 025424 005004      CLR  R4
5204 025426 010205      MOV  R2,R5         ;FOR MUL AND ASHC
5205 025430 100001      BPL  .+4           ;IF MULTIPICAND IS NEG THEN SET R4 = -1
5206 025432 005104      COM  R4            ;FOR ASHC
5207 025434 000277      SCC  R4            ;PRESET CC'S
5208 025436 070200      MUL  R0,R2         ;MULTIPLY R2 BY R0 LEAVE PRODUCT
5209
5210 025440 102406      BVS  2$           ;IN R2,R3 MSH IN R2,LSH IN R3
5211 025442 001405      BEQ  2$           ;PRODUCT WILL NEVER BE = 0
5212 025444 073416      ASHC (SP),R4      ;'MULTIPLY' R4,R5 BY (SP) LEAVE PRODUCT
5213
5214 025446 020204      CMP  R2,R4        ;IN R4,R5 MSH IN R4,LSH IN R5
5215 025450 001002      BNE  2$           ;CHECK MSH RESULT
5216 025452 020305      CMP  R3,R5
5217 025454 001401      BEQ  .+4
5218 025456 104000      HLT
5219 025460 005216      INC  (SP)         ;INCREMENT ASHC SHIFT COUNT
5220 025462 006300      ASL  R0           ;SHIFT MUL MULTIPLIER
5221 025464 102353      BVC  1$
5222
5223 025466 010702 ;CHECK MUL INST WITH MULTIPLIER (R0) = 100000
      MOV  PC,R2      ;R2 = MULTIPICAND

```



```

5225 025472 010227      MOV      R2,(PC)+      ;SAVE MULTIPICAND
5226 025474 000000      .WORD      0          ;CONTAINS ORIGINAL MULTIPICAND
5227 025476 005103      COM      R2,R3
5228 025500 010204      MOV      R2,R4        ;R4 WILL BE MSH 'PRODUCT'
5229 025502 006204      ASR      R4          ;FORM 'PRODUCT'
5230 025504 005104      COM      R4          ;COMPLEMENT MSH 'PRODUCT'
5231 025506 070200      MUL      R0,R2        ;MULTIPLY R2 BY 10000 LEAVING
5232 025510 020204      CMP      R2,R4        ;R2 = MSH, R3 = LSH PRODUCT
5233 025512 001002      BNE      3$          ;COMPARE MSH PRODUCTS
5234 025514 020003      CMP      R0,R3
5235 025516 001401      BEQ      .+4         ;CHECK LSH PRODUCT
5236 025520 104000      BEQ      .+4
5237 025520 104000      HLT
5238
5239
5240
5241
5242
5243
5244
5245 025522 112737 000055 001202 3$: *****
5246 025530 000004      TEST 55      CHECK THE DIV INSTRUCTION
5247 025532 012700 000001      *THE BELOW TEST OF THE DIV INSTRUCTION DIVIDES THE CURRENT PC BY
5248 025536 010716      *1,2,4,8 ETC LEAVING THE QUOTIENT/REMAINDER IN R2/R3. NEXT THE QUOTIENT
5249 025540 011603      *IS MULTIPLIED BY 1,2,4,8 ETC AND THE REMAINDER ADDED. THE RESULT IS
5250 025542 005002      *THEN COMPARED WITH THE ORIGINAL CURRENT PC.
5251 025544 000277      *****
5252 025546 071200      TST55: MOVB  #55,#STSTNM ;LOAD TEST NUMBER
5253
5254 025550 103417      SCOPE
5255 025552 100416      DIV0: MOV     #1,R0      ;R0=DIVISOR
5256 025554 102007      MOV     PC,(SP)      ;SAVE DATA ON STACK
5257 025556 022700 000001      1$: MOV     (SP),R3    ;GET DATA
5258 025560 001012      CLR     R2          ;CLEAR MSH DIVIDEND
5259 025564 032716 100000      SCC     R2
5260 025570 001407      DIV     R0,R2      ;DIVIDE R2 BY R0 LEAVING QUOTIENT IN R2
5261 025572 000407      ;AND REMAINDER IN R3
5262 025574 010204      BCS     2$
5263 025576 070400      BMI     2$
5264 025600 060305      BVC     20$        ;BRANCH IF DIVIDE WORKED
5265 025602 103402      CMP     #1,R0      ;V BIT SHOULD ONLY SET IF DIVIDING BY 1
5266 025604 021605      BNE     2$        ;AND THE LSH OF DIVIDEND
5267 025606 001401      BIT     #100000,(SP) ;IS NEGATIVE
5268 025610 104000      BEQ     2$
5269
5270
5271
5272 025612 006300      BR      3$
5273 025614 102351      20$: MOV     R2,R4      ;GET QUOTIENT
5274
5275
5276 025616 005016      MUL     R0,R4      ;MULTIPLY QUOTIENT BY DIVISOR
5277 025620 005000      ADD     R3,R5      ;ADD REMAINDER TO LSH PRODUCT
5278 025622 012702 000020      BCS     2$        ;SHOULD BE NO CARRY
5279 025626 005067 000012      CMP     (SP),R5    ;CHECK RESULT
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000

```

```

5281 025634 010304      MOV      R3,R4
5282 025636 072316      ASH      (SP),R3      ;SHIFT R3 LEFT (SP) TIMES
5283 025640 013727 177776      MOV      2#PSW,(PC)+ ;SAVE CC'S
5284 025644 000000      .WORD   0           ;CONTAINS ASH (SP),R3 CC'S IN EVEN BYTE
5285                                     ;AND ASH R0,R4 CC'S IN ODD BYTE
5286                                     ;SHIFT R4 LEFT R0 TIMES
5287 025646 072400      ASH      R0,R4      ;SAVE CC'S IN ODD BYTE OF 2$
5288 025650 113767 177776 177767      MOV      2#PSW,2$+1 ;COMPARE RESULTS
5289 025656 020304      CMP      R3,R4      ;BRANCH IF THEY DO NOT COMPARE
5290 025660 001004      BNE      3$        ;CHECK CC'S AFTER ASH INSTRUCTIONS
5291 025662 126767 177756 177755      CMP      2$,2$+1
5292 025670 001'01      BEQ      .+4
5293 025672 104000      HLT
5294                                     ;ERROR! EITHER RESULTS OF SHIFT OR
5295                                     ;RESULT CC'S ARE INCORRECT
5296 025674 005200      INC      R0        ;INCREMENT SHIFT COUNT FOR ASH R0,R4
5297 025676 005216      INC      (SP)      ;INCREMENT SHIFT COUNT FOR ASH (SP),R3
5298 025700 020200      CMP      R2,R0
5299 025702 001351      BNE      1$
5300 ASHR1: CLR      (SP)      ;(SP) = SHIFT COUNT FOR ASH (SP),R4
5301 CLR      R0        ;R0 = SHIFT COUNT FOR ASH R0,R4
5302 NEG      R2        ;R2 = MAX RIGHT SHIFT COUNT (SET BY
5303                                     ;ABOVE TEST TO 16. NOW = -16.
5304 025712 005067 000012      CLR      2$        ;CLEAR CC'S HOLDING ADDRESS
5305 025716 010704      MOV      PC,R4      ;R4,R5 = DATA TO BE SHIFTED RIGHT
5306 025720 010405      MOV      R4,R5
5307 025722 072416      ASH      (SP),R4    ;SHIFT R4 RIGHT (SP) TIMES
5308 025724 013727 177776      MOV      2#PSW,(PC)+ ;SAVE CC'S
5309 025730 000000      .WORD   0           ;CONTAINS ASH (SP),R4 CC'S IN EVEN BYTE
5310 025732 072500      ASH      R0,R5      ;SHIFT R5 RIGHT R0 TIMES
5311 025734 113767 177776 177767      MOV      2#PSW,2$+1 ;SAVE CC'S IN ODD BYTE 2$
5312 025742 020405      CMP      R4,R5      ;CHECK RESULTS
5313 025744 001004      BNE      3$
5314 025746 126767 177756 177755      CMP      2$,2$+1    ;CHECK RESULT CC'S
5315 025754 001401      BEQ      .+4
5316 025756 104000      HLT
5317                                     ;ERROR! EITHER RESULTS OR RESULT CC'S
5318 025760 005300      DEC      R0        ;DID NOT COMPARE
5319 025762 005316      DEC      (SP)      ;DECREMENT SHIFT COUNT
5320 025764 020002      CMP      R0,R2      ;DECREMENT SHIFT COUNT FOR ASH (SP),R4
5321 025766 001351      BNE      1$        ;CHECK FOR MAX RIGHT SHIFT
5322                                     ;*****
5323                                     ;*TEST 56      DIVIDE AGAIN
5324                                     ;*
5325                                     ;* THE BELOW TEST CHECKS THE DIVIDE INSTRUCTION BY DIVIDING
5326                                     ;* THE CURRENT PC BY ITSELF+1. THE QUOTIENT (IN R2) ALWAYS = 0,
5327                                     ;* AND THE REMAINDER (IN R3) ALWAYS = THE CURRENT PC.
5328                                     ;*****
5328 025770 112737 000056 001202      ST56: MOV      #56,2#STSTNM ;LOAD TEST NUMBER
5329 025776 000004      SCOPE
5330 026000 010703      DIV1: MOV      PC,R3      ;CURRENT PC IS LSH DIVIDEND
5331 026002 006702      SXT      R2        ;EXTEND SIGN TO R2 (MSH DIVIDEND)
5332 026004 010304      MOV      R3,R4      ;SAVE ORIGINAL DIVIDEND
5333 026006 010316      MOV      R3,(SP)    ;PUT ON STACK
5334 026010 005216      INC      (SP)      ;ADD 1 (WILL BE DIVISOR)
5335 026012 100002      BPL      1$        ;BRANCH IF POSITIVE

```

```

5337 026020 071216          15:  DIV      (SP),R2          ;DIVIDE R2 BY (SP)
5338 026022 103410          BCS      25$              ;CHECK CONDITION CODES
5339 026024 102407          BVS      25$
5340 026026 001006          BNE      25$
5341 026030 100405          BMI      25$
5342 026032 005702          TST      R2              ;CHECK QUOTIENT (R2 = 0)
5343 026034 001361          BNE      DIV1
5344 026036 010416          MOV      R4,(SP)         ;GET ORIGINAL DIVISOR
5345 026040 020316          CMP      R3,(SP)         ;CHECK REMAINDER
5346 026042 001401          BEQ      .+4
5347 026044 104000          25:  HLT                    ;REPORT ERROR
5348
5349
5350
5351 026046 112737 000057 001202  *TEST 57 CHECK SPL INSTRUCTION
5352 026054 000004          *TEST 57:  MOV      #57,2#STSTMM ;LOAD TEST NUMBER
5353 026056 012702          SCOPE
5354 026060 000237          SPLTST: MOV      (PC)+,R2    ;R2 CONTAINS OP CODE FOR SPL 7
5355 026062 005004          SPL      7
5356 026064 042744 000340          CLR      R4
5357 026070 011403          BIC      #PR7,-(R4)      ;CLEAR PRIORITY LEVEL BITS IN PSW
5358 026072 042703 177757          MOV      (R4),R3        ;GET CURRENT PSW
5359
5360 026076 012767 000230 000010          BIC      #177757,R3     ;R3 CONTAINS CORRECT PSW AFTER SPL
5361 026104 012767 000237 000050          MOV      #SPL+0,2$      ;INITIALIZE SPL INSTRUCTIONS
5362 026112 000257          MOV      #SPL+7,5$
5363 026114 000230          15:  CCC                    ;CLEAR CONDITION CODES
5364 026116 121403          25:  SPL      0            ;SET PRIORITY LEVEL (NOTE: SPL=NOP IF USER/SUPER MODE)
5365 026120 001401          CMPB     (R4),R3        ;CHECK RESULT OF SPL ABOVE
5366 026122 104000          BEQ      .+4
5367 026124 032714 140000          HLT
5368 026130 001002          BIT      #UM,(R4)       ;ERROR! SPL ABOVE FAILED
5369 026132 062703 000040          BNE      3$             ;IF NOT IN KERNEL MODE THEN SPL
5370 026136 005267 177752          ADD      #40,R3         ;ACTS AS A NOP
5371 026142 026702 177746          35:  INC      2$,R3       ;SET NEXT CORRECT PSW RESULT
5372 026146 002761          CMP      2$,R2          ;SET NEXT SPL INSTRUCTION
5373 026150 012702          BLT      1$             ;CHECK IF DONE
5374 026152 000230          MOV      (PC)+,R2      ;LOOP UNTIL DONE CHANGING SPL EACH PASS
5375 026154 052703 000017          SPL      0            ;R2 CONTAINS SPL INSTRUCTION BELOW
5376 026160 000277          BIS      #17,R3        ;SET CONDITION CODE RESULT INTO R3
5377 026162 000237          45:  SCC                    ;SET CONDITION CODES
5378 026164 121403          55:  SPL      7            ;SET PRIORITY LEVEL
5379 026166 001401          CMPB     (R4),R3        ;CHECK RESULT OF SPL ABOVE
5380 026170 104000          BEQ      .+4
5381 026172 032714 140000          HLT
5382 026176 001002          BIT      #UM,(R4)       ;ERROR! SPL ABOVE FAILED
5383 026200 162703 000040          BNE      6$             ;CHECK IF IN KERNEL MODE
5384 026204 005367 177752          SUB      #40,R3         ;SET NEXT CORRECT PSW RESULT
5385 026210 026702 177746          65:  DEC      5$,R3       ;SET NEXT SPL
5386 026214 002361          CMP      5$,R2          ;CHECK IF DONE ALL SPL'S
5387
5388
5389
5390
5391
*****
*TEST 60 CHECK PIRQ LOGIC
* THIS TEST CHECKS THAT WHEN A REQUEST IS MADE AT A LEVEL = TO THE
* CURRENT PROCESSOR PRIORITY LEVEL THAT NO INTERRUPT TAKES PLACE, AND
* THAT WHEN A REQUEST IS MADE AT A LEVEL 1 GREATER THAN THE CURRENT PRO-

```

F10

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 103  
T60 CHECK PIRQ LOGIC

SEQ 0122

```

5393          ;*****
5394 026216 112737 000060 001202 †ST60:  MOVB  #60,‡STSTNM          ;LOAD TEST NUMBER
5395 026224 000004          SCOPE
5396 026226 012700 026372  PIRQ0:  MOV   #4‡,R0          ;R0 POINTS TO A TABLE OF CORRECT PIRQ
5397          ;CONTENTS AFTER AN INTERRUPT
5398 026232 012702 000400          MOV   #400,R2          ;R2 CONTAINS INTERRUPT REQUEST LEVEL
5399 026236 005003          CLR   R3              ;R3 CONTAINS PROCESSOR PRIORITY LEVEL
5400 026240 012704 177772          MOV   ‡PIRQ,R4        ;R4 CONTAINS ADDRESS OF PIRQ REGISTER
5401 026244 005014          CLR   (R4)           ;INITIALZE REQUEST LEVEL TO 0
5402 026246 013737 177776 000242 ‡‡PSW,‡‡PIRQVEC+2    ;RETAIN MODE & REG SET ON TRAP
5403 026254 112737 000340 000242 ‡‡PR7,‡‡PIRQVEC+2    ;ASSUME LEVEL 7 ON INTERRUPT
5404 026262 112737 000340 000016 ‡‡PR7,‡‡TBITVEC+2    ;PRIORITY LEVEL 7 ON TRAP
5405 026270 012737 026330 000240 1‡:  MOV   ‡‡‡PIRQVEC    ;SET PIRQ ERROR INTERRUPT VECTOR
5406 026276 063737 001506 000240 ‡‡FACTOR,‡‡PIRQVEC  ;ADD RELOCATION FACTOR
5407 026304 110337 177776          MOVB  R3,‡‡PSW        ;SET CP PRIORITY LEVEL
5408 026310 050214          BIS   R2,(R4)         ;MAKE REQUEST AT LEVEL = TO CP LEVEL
5409 026312 100436          BMI   5‡            ;BRANCH WHEN DONE
5410 026314 062737 000002 000240 ‡‡‡-2‡,‡‡PIRQVEC    ;SET PIRQ INTERRUPT VECTOR TO 3‡
5411 026322 006302          ASL  R2
5412 026324 050214          BIS   R2,(R4)        ;MAKE REQUEST AT LEVEL 1 HIGHER
5413 026326 000240          NOP
5414 026330 104000          2‡:  HLT            ;ERROR! EITHER AN INTERRUPT OCCURED
5415          ;WHEN RQST LEVEL = CP LEVEL (PIRQVEC)=2‡
5416          ;OR INTERRUPT FAILED (PIRQVEC)=3‡
5417 026332 022014          3‡:  CMP   (R0)+,(R4)    ;CHECK CONTENTS OF PIRQ REGISTER
5418 026334 001406          BEQ  6‡
5419 026336 013737 177772 001276 ‡‡PIRQ,‡‡STMPD        ;SAVE PIRQ
5420 026344 005037 177772          CLR  ‡‡PIRQ
5421 026350 104000          HLT            ;ERROR! INCORRECT PIRQ CONTENTS
5422 026352 062703 000040          6‡:  ADD   #40,R3        ;SET NEXT CP PRIORITY LEVEL
5423 026356 040214          BIC  R2,(R4)        ;LOWER LEVEL BY 1
5424 026360 012716 026270          MOV  #1‡,(SP)       ;ADJUST RETURN ADDRESS
5425 026364 063716 001506          ADD  ‡‡FACTOR,(SP)  ;TO RETURN TO 1‡
5426 026370 000006          30‡: RTT
5427          ;
5428          ;TABLE OF CORRECT PIRQ REGISTER CONTENTS ON INTERRUPT
5429 026372 001042          4‡:  1042          ;PIR1+PIA1
5430 026374 003104          3104          ;PIR2+PIR1+PIA2
5431 026376 007146          7146          ;PIR3+PIR2+PIR1+PIA3
5432 026400 017210          17210         ;PIR4+PIR3+PIR2+PIR1+PIA4
5433 026402 037252          37252         ;PIR5+PIR4+PIR3+PIR2+PIR1+PIA5
5434 026404 077314          77314         ;PIR6+PIR5+PIR4+PIR3+PIR2+PIR1+PIA6
5435 026406 177356          177356        ;PIR7+PIR6+PIR5+PIR4+PIR3+PIR2+PIR1+PIA7
5436          ;
5437 026410 005014          5‡:  CLR   (R4)          ;CLEAR PIRQ REGISTER
5438 026412 012737 000242 000240 ‡‡PIRQVEC+2,‡‡PIRQVEC  ;RESET PIRQVEC TO HALT AT PIRQVEC+2
5439 026420 005037 000242          CLR  ‡‡PIRQVEC+2
5440 026424 105037 177776          CLRB ‡‡PSW
5441 026430 042737 000340 000016 ‡‡PR7,‡‡TBITVEC+2
5442          ;*****
5443          ;*TEST 61 CHECK MICRO-BREAK REGISTER
5444          ;* THIS TEST SHIFTS A '0' BIT THRU ALL BIT POSITIONS.
5445          ;*****
5446 026436 112737 000061 001202 †ST61:  MOVB  #61,‡STSTNM          ;LOAD TEST NUMBER
5447 026444 000004          SCOPE

```

```

5449 026452 011246          MOV      (R2),-(SP)      ;SAVE ORIG CONTENTS
5450 026454 012700 000376    MOV      #376,R0        ;SET DATA PATTERN
5451 026460 010012          MOV      R0,(R2)        ;LOAD REGISTER WITH PATTERN
5452 026462 021200          CMP      (R2),R0        ;AND CHECK
5453 026464 001004          BNE      3$             ;BRANCH IF INCORRECT
5454 026466 000261          SEC                     ;SET 'C'
5455 026470 006100          ROL      R0             ;SHIFT DATA
5456 026472 103772          BCS      1$
5457 026474 000402          BR       4$
5458 026476 104000          3$:      HLT                     ;ERROR DATA IN R0 NOT IN UBREAK REG
5459 026500 000772          BR       2$             ;CONTINUE TEST
5460 026502 012612          4$:      MOV      (SP)+,(R2)    ;RESTORE ORIG UBREAK CONTENTS
5461 *****
5462 *****
5463 *****
5464 026504 112737 000062 001202 1$T62:  MOVB    #62,#STSTNM      ;LOAD TEST NUMBER
5465 026512 000004          SCOPE
5466 026514 032737 140000 177776 MPI:    BIT     #UM,#PSW        ;KERNEL MODE?
5467 026522 001545          BEQ     ENDCP          ;YES EXIT TEST
5468 026524 010746          MOV     PC, -(SP)
5469 026526 062716 000134          ADD     #5$-.,(SP)
5470 026532 012637 000250          MOV     (SP)+,#MMVEC   ;SET MEM MGMT ABORT VECTOR
5471 026536 005046          CLR     -(SP)         ;CLEAR CHECK WORD
5472 026540 010603          MOV     SP,R3
5473 026542 010346          MOV     R3, -(SP)     ;PUT ADDRESS OF CHECK WORD ON THE STACK
5474 026544 105737 001503          TSTB   #MMON          ;CHECK IF MEM MGMT IS ENABLED
5475 026550 001417          BEQ     1$            ;BRANCH IF OFF
5476 026552 013737 177640 177654          MOV     #UIPAR0,#UIPAR6 ;SET UP USER PAGE ADDR. REG.
5477 026560 012737 006006 177614          MOV     #6006,#UIPOR6  ;SET USER PAGE DESC REG R/W UP 6 PAGES
5478 026566 013737 172240 172254          MOV     #SIPAR0,#SIPAR6
5479 026574 012737 006006 172214          MOV     #6006,#SIPOR6  ;SET SUPER PAGE DESC. REG.
5480 026602 062706 140000          10$:    ADD     #140000,SP   ;SET CURRENT MODE'S STACK POINTER
5481 026606 000240          NOP
5482 026610 010746          1$:      MOV     PC, -(SP)
5483 026612 062716 000024          ADD     #3$-.,(SP)
5484 026616 012637 000020          MOV     (SP)+,#IOTVEC  ;SET IOT TRAP VECTOR
5485 026622 000004          IOT
5486 026624 005266 000002          INC     2(SP)         ;INCREMENT CHECK WORD
5487 026630 001417          BEQ     6$
5488 026632 104000          4$:      HLT                     ;ERROR! MFPI,MTPI FAILURE-FOR BETTER
5489 026634 000415          BR      6$            ;ISOLATION SUGGEST RUNNING MFPI DIAG. DCKTD/E
5490 026636 000240          3$:      NOP
5491 026640 006506          MFPI   SP             ;PSW=KERNEL MODE,PREV USER OR SUPER MODE
5492 026642 006536          MFPI   2(SP)+         ;GET PREV. MODES STACK POINTER
5493 026644 006576 000000          MFPI   2(SP)         ;GET DATA (AN ADDRESS) ON PREV MODE'S STACK
5494 026650 000240          NOP
5495 026652 001367          BNE    4$            ;SPACE AND PUSH ONTO KERNEL STACK
5496 026654 005116          COM    (SP)          ;ERROR IF BRANCH TAKEN! SHOULD HAVE A ZERO ON THE STACK
5497 026656 006636          MTPI   2(SP)+        ;COMPLEMENT OPERAND
5498
5499 026660 000002          RTI
5500 026662 104000          5$:      HLT                     ;RETURN TO INST FOLLOWING IOT ABOVE
5501 026664 105037 177776          CLRB   #PSW          ;ERROR! MEMORY MANG. ABORT
5502 026670 012737 053274 000250          6$:      MOV     #KTABRT,#MMVEC ;SET PRIORITY LEVEL BACK TO 0
5503 026676 012737 044534 000020          MOV     #SSCOPE,#IOTVEC ;RESTORE VECTOR

```

```

5505 ;*****
5506 ;*TEST 63 CHECK ILLEGAL HALT
5507 ;*****
5508 026710 112737 000063 001202 †ST63: MOV  #63,‡STSTNM ;LOAD TEST NUMBER
5509 026716 000004 SCOPE
5510 026720 010746 HALT1: MOV PC, -(SP) ;GET CURRENT PC
5511 026722 062716 000022 ADD #2‡, -(SP)
5512 026726 011637 000004 MOV (SP),‡ERRVEC ;SET ERROR TRAP VECTOR TO 2‡ BELOW
5513 026732 012637 000010 MOV (SP)+,‡RESVEC ;LOAD RESERVED INST TRAP VECTOR (11/40)
5514 026736 000000 HALT ;SHOULD TRAP TO 4 IN USER/SUPER MODE
5515 026740 104000 1‡: HLT ;ERROR! HALT ABOVE FAILED IN USER/SUPER MODE
5516 026742 000404 BR 3‡
5517 026744 010716 2‡: MOV PC, (SP) ;REPLACE RETURN PC WITH
5518 026746 062716 000006 ADD #3‡-, (SP) ;ADDRESS OF 3‡ BELOW
5519 026752 000002 RTI ;RETURN (TO 3‡)
5520
5521 026754 012737 053442 000004 3‡: MOV #ERPRT,‡ERRVEC ;RESTORE ERROR TRAP VECTOR
5522 026762 012737 053370 000010 MOV #RESERR,‡RESVEC
5523 026770 105037 177776 CLR‡PSW
5524 026774 005037 177766 CLR‡CPUERR
5525 ;*****
5526 ;*TEST 64 CHECK RESET IN SUPER/USER MODE
5527 ;*****
5528 027000 112737 000064 001202 †ST64: MOV  #64,‡STSTNM ;LOAD TEST NUMBER
5529 027006 000004 SCOPE
5530 027010 000277 RESET1: SCC
5531 027012 013700 177776 MOV‡PSW, R0 ;GET CURRENT PSW
5532 027016 000277 SCC
5533 027020 000005 RESET
5534 027022 023700 177776 CMP‡PSW, R0 ;CHECK THAT PSW UNCHANGED BY RESET ABOVE
5535 027026 001401 BEQ .+4
5536 027030 104000 HLT ;ERROR! RESET CLEARED MODE BITS IN PSW
5537 027032 010037 177776 MOV R0,‡PSW ;RESTORE PSW (FOR ERROR)
5538 027036 ENDCP:
5539 027036 000004 RELE6: SCOPE
5540 027040 010702 MOV PC, R2
5541 027042 062702 000012 ADD #1‡, R2
5542 027046 012707 034242 MOV #RELOC, PC ;GO RELOCATE PROGRAM CODE
5543 027052 000000 REL66: .WORD 0
5544 ;66666666666666 LAST ADDRESS OF CODE TO BE RELOCATED 666666666666
5545
5546 ;*****
5547 ;*TEST 65 TEST STACK LIMIT REGISTER
5548 ;*****
5549 027054 112737 000065 001202 †ST65: MOV  #65,‡STSTNM ;LOAD TEST NUMBER
5550 027062 012767 000001 152226 MOV #1,‡TIMES ;;DO 1 ITERATION
5551 027070 000004 SCOPE
5552
5553 .SBTTL START OF SECTION 7
5554 .7777777777777777 FIRST ADDRESS TO BE RELOCATED 7777777777
5555 027072 010700 REL7: MOV PC, R0 ;GET PC
5556 027074 005740 TST -(R0) ;R0 CONTAINS THE ADDRESS OF REL7
5557 027076 010037 001512 MOV R0,‡FRSTAD ;SAVE
5558 027102 010700 MOV PC, R0 ;GET CURRENT PC
5559 027104 162700 027104 SUB #., R0 ;SUBTRACT RELOCATION FACTOR

```

```

5561 027114 010737 001212      MOV      PC, 2#SLPERR      ;SET LOOP ADDRESS
5562 027120 062737 000030 001212      ADD      #30, 2#SLPERR    ;ADJUST
5563 027126 013737 001212 001210      MOV      2#SLPERR, 2#SLPADR
5564 027134 105737 001502      TSTB    2#NEXEC          ;BR IF TEST CODE TO BE EXECUTED
5565 027140 001402          BEQ      +6
5566 027142 000167 001206      JMP      RELE7
5567          ;THIS TEST SHIFTS A '1' BIT THROUGH ALL BIT POSITIONS
5568 027146 012702 177774      MOV      #STKLM, R2      ;GET ADDRESS OF STACK LIM REG
5569 027152 005022          CLR      (R2)+          ;CLEAR STACK LIMIT REG
5570 027154 032712 000020      BIT      #20 (R2)       ;EXIT TEST IF 'T' BIT IS SET
5571 027160 001116          BNE     101$
5572 027162 052712 000340      BIS      #340, (R2)     ;SET PRIORITY LEVEL 7 TO PREVENT
5573          ;ANY INTERRUPTS FROM OCCURRING
5574 027166 012700 000400      MOV      #400, R0      ;SET CHECK DATA
5575 027172 010042 1$:      MOV      R0, -(R2)     ;MOVE TO STACK LIMIT REG
5576 027174 022200          CMP      (R2)+, R0     ;AND CHECK RESULT
5577 027176 001401          BEQ     2$
5578 027200 104000          HLT
5579          ;ERROR! STACK LIMIT DID NOT
5580          ;LOAD CORRECTLY. CORRECT RESULT
5581          ;IS IN R0
5582 027202 006300 2$:      ASL      R0            ;SHIFT '1' BIT LEFT
5583 027204 103372          BCC     1$            ;LOOP UNTIL 1 BIT SHIFTS OUT
5584 027206 005042          CLR      -(R2)        ;CLEAR STACK LIMIT REG
5585          ;THIS TEST CHECKS THAT A PROPER 'RED' ZONE VIOLATION OCCURS, NOTE THAT
5586          ;NO 'RED ZONE' VIOLATION WILL OCCUR IF IN USER/SUPER MODES.
5587          ;A RED ZONE VIOLATION PUSHES THE CURRENT PSW, PC ON A STACK AT 2 AND 0
5588          ;AND TAKES THE NEXT INSTRUCTION FROM THE PC IN LOCATION 4. THE INST-
5589          ;RUCTION CAUSING THE RED ZONE VIOLATION IS 'ABORTED'.
5590 027210 010746          MOV      PC, -(SP)     ;GET CURRENT PC
5591 027212 062716 000060      ADD      #4$, -(SP)    ;FORM ADDRESS OF 4$ BELOW
5592 027216 012637 000004          MOV      (SP)+, 2#ERRVEC ;SET ERROR TRAP VECTOR TO 4$ BELOW
5593 027222 013737 177776 000006      MOV      2#PSW, 2#ERRVEC+2 ;RETAIN CURRENT STATUS ON TRAP
5594 027230 010712          MOV      PC, (R2)     ;SET STACK LIMIT TO CURRENT PC
5595          ;+400
5596 027232 011206          MOV      (R2), SP     ;AND STACK PTR = STACK LIMIT REG
5597 027234 010603          MOV      SP, R3      ;SAVE STACK PTR
5598 027236 016304 000336      MOV      336(R3), R4 ;SAVE MEMORY LOC CONTENTS
5599          ;AT 'RED ZONE' BOUNDARY
5600 027242 032737 140000 177776      BIT      #UM, 2#PSW    ;BRANCH IF IN KERNEL MODE
5601 027250 001403          BEQ     20$
5602 027252 010466 000336      MOV      R4, 336(SP)  ;SHOULD NOT CAUSE TRAP
5603 027256 000432          BR      100$
5604          ;
5605 027260 005066 000336 20$:      CLR      336(SP)      ;SHOULD CAUSE 'RED ZONE' TRAP
5606 027264 012706 000700 3$:      MOV      #SUPSTK, SP  ;RESTORE THE STACK
5607 027270 104000          HLT                  ;ERROR! FAILED TO TRAP
5608          ;
5609 027272 032737 140000 000002 4$:      BIT      #UM, 2#2     ;CHECK IF TRAPPED WHEN IN USER
5610          ;/SUPER MODES (2 CONTAINS OLD PSW)
5611 027300 001013          BNE     99$          ;GO TO ERROR CALL
5612 027302 010600          MOV      SP, R0      ;STACK PTR SHOULD = 0
5613 027304 001011          BNE     99$          ;GO TO ERROR CALL IF NOT 0
5614 027306 026304 000336      CMP      336(R3), R4 ;CHECK THAT INST WAS ABORTED
5615 027312 001006          BNE     99$          ;GO REPORT ERRPR

```

```

5617 027316 010705          MOV    PC,R5          ;GET CURRENT PC
5618 027320 062705 177744  ADD    #3$,R5        ;FORM ADDRESS OF 3$ ABOVE
5619 027324 020516          CMP    R5,(SP)       ;CHECK THAT RETURN PC IS ON
5620                                ;THE STACK (AT 0)
5621 027326 001406          BEQ    100$         ;EXIT TEST
5622
5623                                ;ERROR
5624 027330 005012          99$:  CLR    (R2)        ;CLEAR STACK LIMIT REG
5625 027332 010463 000336  MOV    R4,336(R3)   ;RESTORE MEM LOCATION
5626 027336 012706 000700  MOV    #SUPSTK,SP   ;SET STACK PTR
5627 027342 104000          HLT                                ;ERROR!
5628 027344 010463 000336  100$: MOV    R4,336(R3)  ;RESTORE MEM LOCATION
5629 027350 005022          CLR    (R2)+        ;CLEAR STACK LIM REG
5630 027352 012706 000700  MOV    #SUPSTK,SP   ;SET STACK PTR
5631 027356 042712 000340  BIC    #340,(R2)    ;SET PRIORITY LEVEL BACK TO 0
5632 027362 012737 053442 000004  MOV    #ERPAT,#ERRVEC ;RESTORE ERROR TRAP VECTOR
5633 027370 013737 177776 000006  MOV    #PSW,#ERRVEC+2
5634 027376 112737 000340 000006  MOVB   #PR7,#ERRVEC+2
5635 027404 042737 000020 000006  BIC    #BIT4,#ERRVEC+2
5636 027412 005037 177766          CLR                                ;CLEAR ERROR REG
5637 027416
5638                                101$:
5639                                ;*****
5640                                ;*TEST 66 MEMORY MANAGEMENT REGISTER TESTS
5641                                ;* PDR TEST - THIS TEST WRITES 64. RANDOM #'S INTO EACH PDR REGISTER
5642                                ;* NOTE: IF MEM MGMT IS ENABLED ONLY PDR/PAR PAIRS 3-5 ARE TESTED.
5643                                ;*****
5643 027416 112737 000066 001202 1$T66: MOVB   #66,#STSTNM ;LOAD TEST NUMBER
5644 027424 000004          SCOPE
5645
5646 027426 012702 027652  KTPDR: MOV    #PORTBL,R2 ;SET TABLE ADDRESS OF PDR'S
5647 027432 012705 100360  MOV    #100360,R5   ;SET BIT MASK (11/45)
5648 027436 012200          1$:  MOV    (R2)+,R0    ;GET PDR ADDRESS
5649 027440 001435          BEQ    100$         ;EXIT ON '0' TERMINATOR
5650 027442 012716 000010  2$:  MOV    #8,(SP)   ;SET LOOP COUNT (FOR 8 REGS)
5651 027446 105737 001503  TSTB   #MMON        ;BRANCH IF MEM MGMT DISABLED
5652 027452 001404          BEQ    3$
5653 027454 062700 000006  ADD    #6,R0        ;SET R0 TO PDR3
5654 027460 012716 000003  MOV    #3,(SP)     ;AND LIMIT TO TEST 3 PDRS
5655 027464 012703 000040  3$:  MOV    #32.,R3    ;SET DATA COUNT
5656 027470 005004          CLR    R4          ;INITIALIZE DATA TO BE WRITTEN
5657 027472 040504          4$:  BIC    R5,R4       ;CLEAR NON-SETTABLE BITS
5658 027474 010410          MOV    R4,(R0)    ;WRITE INTO PDR
5659 027476 021004          CMP    (R0),R4    ;AND CHECK DATA READ BACK
5660 027500 001013          BNE   99$         ;GO TO ERROR CALL
5661 027502 005104          COM    R4         ;COMPLEMENT DATA
5662 027504 040504          BIC    R5,R4       ;CLEAR NON-SETTABLE BITS
5663 027506 010410          MOV    R4,(R0)    ;WRITE COMPLEMENT DATA INTO PDR
5664 027510 021004          CMP    (R0),R4    ;AND CHECK
5665 027512 001006          BNE   99$         ;GO TO ERROR CALL
5666 027514 060104          ADD    R1,R4      ;STEP DATA
5667 027516 077313          SOB   R3,4$      ;STEP TO NEXT REGISTER
5668 027520 005020          5$:  CLR    (R0)+      ;DECREMENT REGISTER COUNT
5669 027522 005316          DEC   (SP)
5670 027524 001357          BNE   3$
5671 027526 000743          BR   1$          ;GET NEXT SET OF 8 REGISTERS

```



# K10

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 108  
T66 MEMORY MANAGEMENT REGISTER TESTS

SEQ 0127

```

5673 027530 104000          99$:   HLT                ;ERROR! INCORRECT DATA READ
5674                                     ;BACK FROM PDR. ADDRESS OF
5675                                     ;PDR IS IN R0, DATA IS IN R4
5676 027532 000772          BR      5$                ;STEP TO NEXT REGISTER
5677 027534
5678                                     100$:
5679                                     ;*****
5680                                     ;*TEST 67 PAR TEST
5681                                     ;* PAR TEST - THIS TEST WRITES 64. COMPLEMENTING RANDOM #'S INTO EACH PAR.
5682                                     ;*****
5682 027534 112737 000067 001202 1ST67:  MOVB    #67,2#STSTM      ;LOAD TEST NUMBER
5683 027542 000004                                     SCOPE
5684 027544 012702 027670  KTPAR:  MOV     #PARTBL,R2      ;GET TABLE ADDRESS OF PAR'S
5685 027550 005005                                     CLR     R5
5686 027552 012200          1$:   MOV     (R2)+,R0      ;GET PAR ADDRESS
5687 027554 001435          BEQ     100$          ;EXIT ON '0' TERMINATOR
5688 027556 012716 000010  2$:   MOV     #8,(SP)      ;SET LOOP COUNT (FOR 8 REGS.)
5689 027562 105737 001503          TSTB   2#MMON        ;BRANCH IF MEM MGMT DISABLED
5690 027566 001404          BEQ     3$
5691 027570 062700 000006          ADD     #6,R0        ;SET R0 TO PAR3
5692 027574 012716 000003          MOV     #3,(SP)     ;AND LIMIT TEST TO 3 PARS
5693 027600 012703 000040  3$:   MOV     #32.,R3     ;SET DATA COUNT
5694 027604 005004          CLR     R4          ;INITIALIZE DATA
5695 027606 040504  4$:   BIC     R5,R4        ;CLEAR NON-SETTABLE BITS
5696 027610 010410          MOV     R4,(R0)     ;WRITE INTO PAR
5697 027612 021004          CMP     (R0),R4     ;AND CHECK
5698 027614 001013          BNE    99$         ;TAKE ERROR EXIT
5699 027616 005104          COM    R4          ;COMPLEMENT DATA
5700 027620 040504          BIC     R5,R4        ;CLEAR NON-SETTABLE BITS
5701 027622 010410          MOV     R4,(R0)     ;WRITE COMPLEMENT DATA
5702 027624 021004          CMP     (R0),R4     ;AND CHECK
5703 027626 001006          BNE    99$         ;TAKE ERROR EXIT
5704 027630 060104          ADD     R1,R4        ;STEP DATA
5705 027632 077313          SOB    R3,4$       ;LOOP UNTIL FINISHED
5706
5707 027634 005020  5$:   CLR     (R0)+
5708 027636 005316          DEC    (SP)         ;DECREMENT REGISTER COUNT
5709 027640 001357          BNE    3$          ;BRANCH IF 8 REGS NOT DONE
5710 027642 000743          BR     1$
5711
5712 027644 104000          99$:   HLT                ;ERROR! INCORRECT DATA READ BACK
5713                                     ;FROM PAR. ADDRESS OF PAR IS IN
5714                                     ;R0, DATA IS IN R4
5715 027646 000772          BR      5$                ;DO NEXT REGISTER
5716 027650
5717 027650 000416          100$:  BR      TST70          ;GO TO NEXT TEST
5718                                     ;TABLES FOR PDR & PAR TESTS ABOVE
5719 027652 172300          PORTBL: .WORD  KIPDRO
5720 027654 177600          .WORD  UIPDRO
5721 027656 172200          .WORD  SIPDRO      ;CHANGED TO '0' IF 11/40
5722 027660 172320          .WORD  KDPDRO
5723 027662 177620          .WORD  UDPDRO
5724 027664 172220          .WORD  SDPDRO
5725 027666 000000          .WORD  0           ;TERMINATOR
5726
5727 027670 172340          PARTBL: .WORD  KIPARO

```

```

5729 027674 172240 .WORD SIPARO ;CHANGED TO '0' IF 11/40
5730 027676 172360 .WORD KOPARO
5731 027700 177660 .WORD UDPARO
5732 027702 172260 .WORD SDPARO
5733 027704 000000 .WORD 0 ;TERMINATOR
5734
5735
5736
5737
5738
5739
5740
5741 027706 112737 000070 001202 TST70: MOVB #70,2#STSTNM ;LOAD TEST NUMBER
5742 027714 000004 SCOPE
5743 027716 105737 001503 KTABT: TSTB 2#MMON ;BRANCH IF MEM MGMT DISABLED
5744 027722 001515 BEQ KTEX
5745 027724 005037 172350 CLR 2#KIPAR4 ;SET UP MEM MGMT REGISTERS
5746 027730 005037 172310 CLR 2#KIPDR4 ;TO ABORT IF A MEMORY
5747 027734 005037 177650 CLR 2#UIPAR4 ;REFERENCE IS MADE TO
5748 027740 005037 177610 CLR 2#UIPDR4 ;ADDRESSES (VIRTUAL) BETWEEN
5749 027744 005037 172250 CLR 2#SIPAR4
5750 027750 005037 172210 CLR 2#SIPDR4
5751 027754 013746 000250 1S: MOV 2#MMVEC, -(SP) ;SAVE MEM MGMT VECTOR
5752 027760 013746 000252 MOV 2#MMVEC+2, -(SP) ;AND PRIORITY
5753 027764 010746 MOV PC, -(SP) ;SET MEM MGMT
5754 027766 062716 000040 ADD #4, -(SP) ;VECTOR TO 4S BELOW
5755 027772 012637 000250 MOV (SP)+, 2#MMVEC
5756 027776 013737 177776 000252 MOV 2#PSW, 2#MMVEC+2
5757 030004 005000 CLR R0 ;CLEAR ABORT INDICATOR
5758 030006 010702 MOV PC, R2 ;SET R2 AND R3 NOTE:
5759 030010 012703 100000 MOV #100000, R3 ;THE REF VIA R3 CAUSES THE
5760 030014 014223 2S: MOV -(R2), (R3)+ ;ABORT
5761 030016 005700 3S: TST R0 ;BRANCH IF THE ABORT OCCURRED
5762 030020 001001 BNE .+4
5763 030022 104000 HLT ;REPORT ERROR
5764 030024 000445 BR 100S
5765
5766 030026 013700 177776 ;ABORT HERE
5767 030032 000300 4S: MOV 2#PSW, R0 ;SRO SHOULD CONTAIN
5768 030034 006200 SWAB R0 ;CAUSE FOR ABORT AND
5769 030036 042700 177637 ASR R0 ;ALSO WHICH SEGMENT
5770 030042 062700 100011 BIC #177637, R0 ;WAS IN USE WHEN ABORT
5771 030046 020037 177572 ADD #100011, R0 ;OCCURRED.
5772 030052 001025 CMP R0, 2#SRO
5773 030054 012700 030014 BNE 99$
5774 030060 020037 177576 MOV #2S, R0 ;GET ADDRESS OF INST THAT ABORTED
5775 030064 001020 CMP R0, 2#SR2 ;THAT ABORTED
5776 030066 012700 000362 BNE 99$
5777 030072 120037 177574 MOV #362, R0
5778 030076 001013 CMPB R0, 2#SR1 ;SR1 (11/45) CONTAINS REGISTER
5779 030100 012700 000023 BNE 99$ ;MODIFICATIONS MADE
5780 030104 120037 177575 MOV #23, R0
5781 030110 001006 CMPB R0, 2#SR1+1
5782 030112 012700 030014 BNE 99$
5783 030116 005720 5S: MOV #2S, R0
TST (R0)+ ;R0=ADDRESS OF INST FOLLOWING ABORT

```

```

5785 030122 001001      BNE      99$
5786 030124 000002      RYI
5787      ;ENTER HERE ON ERROR ;RETURN
5788 030126 104000      99$: HLT      ;REPORT ERROR
5789 030130 010716      MOV      PC,(SP)
5790 030132 062716 177664      ADD      #3$-.,(SP)
5791 030136 000002      RTI      ;RETURN
5792 030140 012637 000252      100$: MOV      (SP)+,2#MMVEC+2 ;RESTORE ABORT VECTOR
5793 030144 012637 000250      MOV      (SP)+,2#MMVEC ;& PRIORITY.
5794 030150 012737 000001 177572      MOV      #1,2#SRO ;CLEAR ERROR CONDITIONS
5795 030156
5796
5797
5798
5799
5800 030156 112737 000071 001202      KTEX:
5801 030164 000004      ;*****
5802 030166 032737 000040 172516      ;*TEST 71 MAPPING REGISTER TESTS
5803 030174 001053      ;* THIS TEST LOADS RANDOM #'S INTO EACH MAPPING REGISTER
5804 030176 012700 170200      ;*****
5805 030202 012706 000770      TST71: MOV      #71,2#STSTNM ;LOAD TEST NUMBER
5806 030206 012716 000001      SCOPE
5807 030212 012702 177700      BIT      #BITS 2#MMR3 ;IS MAP ON?
5808 030216 012703 000040      BNE      MAPTH ;BRANCH IF YES
5809 030222 005005      MAPTST: MOV      #MAPL,R0 ;SET ADRS OF FIRST MAP REGISTER
5810 030224 010504      MOV      #SUPSTK,SP ;SETUP THE SP
5811 030226 041604      MOV      #1,(SP) ;SET BIT MASK FOR MAPLO <15-01>
5812 030230 010410      MOV      #177700,R2 ;AND ALSO FOR MAPHO <21-16>
5813 030232 021004      1$: MOV      #32.,R3 ;SET DATA COUNT
5814 030234 001032      2$: CLR      R5 ;SET INITIAL DATA
5815 030236 005105      MOV      R5,R4 ;GET DATA
5816 030240 010504      BIC      (SP),R4 ;CLEAR UNUSED BITS
5817 030242 041604      MOV      R4,(R0) ;LOAD DATA INTO MAPLO <15-01>
5818 030244 010410      CMP      (R0),R4 ;CHECK DATA
5819 030246 021004      BNE      99$ ;BRANCH IF INCORRECT
5820 030250 001024      COM      R5 ;COMPLEMENT TEST DATA
5821 030252 005720      MOV      R5,R4 ;GET TEST DATA
5822 030254 010504      BIC      R2,R4 ;CLEAR UNUSED BITS
5823 030256 040204      MOV      R4,(R0) ;LOAD TEST DATA INTO MAPHO <21-16>
5824 030260 010410      CMP      (R0),R4 ;AND CHECK
5825 030262 021004      BNE      99$
5826 030264 001016      COM      R5 ;COMPLEMENT TEST DATA
5827 030266 005105      MOV      R5,R4 ;GET TEST DATA
5828 030270 010504      BIC      R2,R4 ;CLEAR UNUSED BITS
5829 030272 040204      MOV      R4,(R0) ;LOAD TEST DATA
5830 030274 010410      CMP      (R0),R4 ;AND CHECK
5831 030276 021004      BNE      99$
5832 030300 001010      ADD      PC,R5 ;FORM NEXT TEST DATA
5833 030302 060705      TST      -(R0) ;RESET PTR TO REGISTER <15-01>
5834 030304 005740      SOB      R3,2$ ;AND TEST UNTIL ALL #'S USED
5835 030306 077332      CMP      (R0)+,(R0)+ ;STEP TO NEXT REGISTER PAIR
5836 030310 022020      CMP      #MAPLO+128.,R0 ;BRANCH IF NOT LAST PAIR
5837 030312 022700 170400      BNE      1$
5838 030316 001337      BR
5839 030320 000401      MAPTWO

```



```

5897 030550 004767 002116          JSR    PC,FLTMPY      ;DO THE MULTIPLY
5898 030554 174137 001306          STF    AC1,2#STMP4   ;SAVE RESULT
5899 030560 013737 001404 001262    MOV    2#SAC1,2#SREG2 ;AND SOFTWARE EXP
5900
5901      ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
5902      ;DO THE A*A FIRST
5903 030566 013737 001256 001402    MOV    2#SREG0,2#SAC0 ;GET EXT EXPONENT
5904 030574 172437 001276          LDF    2#STMP0,AC0   ;LOAD OPERAND A
5905 030600 013737 001402 001404    MOV    2#SAC0,2#SAC1 ;SET OPERAND B EXT EXPONENT
5906 030606 172500          LDF    AC0,AC1       ;LOAD B OPERAND
5907 030610 004767 002056          JSR    PC,FLTMPY     ;EXECUTE THE MULTIPLY
5908 030614 174102          STF    AC1,AC2       ;SAVE RESULT
5909 030616 013737 001404 001406    MOV    2#SAC1,2#SAC2
5910
5911      ;NOW DO THE B*B
5912 030624 172437 001302          LDF    2#STMP2,AC0   ;LOAD B OPERAND
5913 030630 172500          LDF    AC0,AC1       ;LOAD THE A OPERAND
5914 030632 013737 001260 001402    MOV    2#SREG1,2#SAC0 ;AND EXT EXPONENT
5915 030640 013737 001402 001404    MOV    2#SAC0,2#SAC1
5916 030646 004767 002020          JSR    PC,FLTMPY     ;DO THE MULTIPLY
5917 030652 174103          STF    AC1,AC3       ;SAVE THE RESULT
5918 030654 013737 001404 001410    MOV    2#SAC1,2#SAC3
5919
5920      ;NOW DO THE 2*B*A
5921 030662 012701 001302          MOV    2#STMP2,R1    ;LOAD THE B OPERAND
5922 030666 172411          LDF    (R1),AC0      ;LOAD THE A OPERAND
5923 030670 172541          LDF    -(R1),AC1     ;AND THE EXT EXPONENTS
5924 030672 013737 001260 001402    MOV    2#SREG1,2#SAC0
5925 030700 013737 001256 001404    MOV    2#SREG0,2#SAC1
5926 030706 004767 001760          JSR    PC,FLTMPY     ;DO THE MULTIPLY
5927 030712 172427 040000          LDF    #1040000,AC0 ;SETUP TO MULTIPLY BY TWO
5928 030716 012737 000002 001402    MOV    #2,2#SAC0
5929 030724 004767 001742          JSR    PC,FLTMPY     ;DO THE MULTIPLY
5930
5931      ;NOW SUM THE RESULTS
5932 030730 013737 001410 001402    MOV    2#SAC3,2#SAC0 ;GET RESULT OF B*B
5933 030736 172403          LDF    AC3,AC0       ;ADD THE RESULT
5934 030740 004767 001776          JSR    PC,FLTADD     ;GET RESULT OF A*A
5935 030744 172402          LDF    AC2,AC0
5936 030746 013737 001406 001402    MOV    2#SAC2,2#SAC0
5937 030754 004767 001762          JSR    PC,FLTADD     ;ADD THIS RESULT
5938 030760 174137 001312          STF    AC1,2#STMP6   ;SAVE FINAL RESULT
5939 030764 013737 001404 001264    MOV    2#SAC1,2#SREG3
5940
5941      ;NOW CHECK BOTH SIDES OF THE EQUATION
5942      ;CALCULATE THE NUMBER OF CORRECT BITS
5943      ;PUT LARGEST EXPONENT OF A**2 OR B**2 IN SAC2
5944 030772 023737 001406 001410    CMP    2#SAC2,2#SAC3
5945 031000 002003          BGE    15            ;BRANCH IF SAC2 ALREADY HAS LARGEST
5946 031002 013737 001410 001406    MOV    2#SAC3,2#SAC2 ;SAC3 WAS LARGER
5947 031010 163737 001404 001406 15:  SUB    2#SAC1,2#SAC2 ;NOW CALCULATE NUMBER
5948 031016 162737 000024 001406    SUB    #20,2#SAC2    ;OF CORRECT BITS WITHIN 2
5949 031024 005437 001406          NEG    2#SAC2        ;MAKE RESULT POSITIVE
5950 031030 172437 001306          LDF    2#STMP4,AC0   ;LOAD RESULT OF LEFT HAND SIDE
5951 031034 013737 001262 001402    MOV    2#SREG2,2#SAC0 ;AND EXTENDED EXPONENT

```

C11

CEQKCC PDP 11 70 CPU EXERCISOR MACY11 30A(1052) 03-MAR-78 13:15 PAGE 113  
CEQKCC.P11 03-MAR-78 13:13 START OF SECTION 8

SEQ 0132

```

5954                                     ;ACTUAL EXP'S ARE EQUAL TO 200
5955 031054 100002                       BPL      35          ;ENSURE RESULT IS POSITIVE
5956 031056 005437 001404                NEG      2#SAC1
5957 031062 023737 001406 001404 35:   CMP      2#SAC2,2#SAC1 ;ANSWERS WITHIN ALLOWABLE NUMBER?
5958 031070 003401                       BLE      SECT2     ;BRANCH IF YES
5959 031072 104014                       45:   ERROR    14    ;RESULTS ARE WRONG
5960                                     ;*****
5961 031074 170127 000000                SECT2: LDFPS    80
5962                                     ;DO A+B
5963 031100 172537 001276                 LDF      2#STMP0,AC1 ;LOAD A OPERAND
5964 031104 172437 001302                 LDF      2#STMP2,AC0 ;LOAD B OPERAND
5965 031110 013737 001256 001404        MOV      2#SREG0,2#SAC1
5966 031116 013737 001260 001402        MOV      2#SREG1,2#SAC0
5967 031124 004767 001612                 JSR      PC,FLTADD   ;ADD THEM
5968 031130 174102                       STF      AC1,AC2     ;SAVE IN AC2
5969 031132 013737 001404 001406        MOV      2#SAC1,2#SAC2 ;AND EXT EXPONENT
5970                                     ;NOW DO THE A-B
5971 031140 172537 001276                 LDF      2#STMP0,AC1 ;LOAD OPERAND A
5972 031144 013737 001256 001404        MOV      2#SREG0,2#SAC1 ;AND EXT EXPONENT
5973 031152 172437 001302                 LDF      2#STMP2,AC0 ;LOAD OPERAND B
5974 031156 013737 001260 001402        MOV      2#SREG1,2#SAC0
5975 031164 004767 001546                 JSR      PC,FLTSUB   ;SUBTRACT THEM
5976                                     ;NOW DO (A+B)*(A-B)
5977 031170 172402                       LDF      AC2,AC0     ;GET RESULT OF (A+B)
5978 031172 013737 001406 001402        MOV      2#SAC2,2#SAC0
5979 031200 004767 001466                 JSR      PC,FLTMPLY  ;FORM THE PRODUCT
5980 031204 174137 001306                 STF      AC1,2#STMP4 ;SAVE RESULT
5981 031210 013737 001404 001262        MOV      2#SAC1,2#SREG2 ;AND EXT EXPONENT
5982                                     ;NOW DO THE B*B
5983 031216 172437 001302                 LDF      2#STMP2,AC0 ;LOAD OPERAND B
5984 031222 013737 001260 001402        MOV      2#SREG1,2#SAC0
5985 031230 172500                       LDF      AC0,AC1     ;B OPERAND IS IN AC0
5986 031232 013737 001402 001404        MOV      2#SAC0,2#SAC1 ;AND EXT EXPONENT
5987 031240 004767 001426                 JSR      PC,FLTMPLY  ;
5988 031244 174102                       STF      AC1,AC2     ;SAVE RESULT IN AC2
5989 031246 013737 001404 001406        MOV      2#SAC1,2#SAC2
5990                                     ;NOW DO THE A*A
5991 031254 172437 001276                 LDF      2#STMP0,AC0 ;LOAD OPERAND A
5992 031260 013737 001256 001402        MOV      2#SREG0,2#SAC0
5993 031266 172500                       LDF      AC0,AC1
5994 031270 013737 001402 001404        MOV      2#SAC0,2#SAC1
5995 031276 004767 001370                 JSR      PC,FLTMPLY  ;EXECUTE THE MULTIPLY
5996 031302 013737 001404 001410        MOV      2#SAC1,2#SAC3 ;SAVE EXT EXPO OF A*A
5997                                     ;NOW DO A**2-B**2
5998 031310 172402                       LDF      AC2,AC0     ;GET B*B
5999 031312 013737 001406 001402        MOV      2#SAC2,2#SAC0 ;A*A IN AC1
6000 031320 004767 001412                 JSR      PC,FLTSUB   ;
6001 031324 174137 001312                 STF      AC1,2#STMP6 ;SAVE IN MEMORY
6002 031330 013737 001404 001264        MOV      2#SAC1,2#SREG3
6003                                     ;NOW COMPUTE THE RESULTS
6004                                     ;CALCULATE THE NUMBER OF CORRECT BITS
6005 031336 023737 001406 001410        CMP      2#SAC2,2#SAC3 ;DETERMINE WHICH EXP IS LARGER
6006 031344 002003                       BGE      25
6007 031346 013737 001410 001406        MOV      2#SAC3,2#SAC2 ;BRANCH IF AC2 LARGER
6008 031354 163737 001404 001406 25:   SUB      2#SAC1,2#SAC2 ;PUT LARGEST IN AC2

```

```

6010 031370 005437 001406      NEG      2#SAC2
6011 031374 172437 001306      LDF      2#STMP4,AC0      ;GET LEFT HAND SIDE
6012 031400 013737 001262 001402      MOV      2#SREG2,2#SAC0
6013 031406 004767 001324      JSR      PC,FLTSUB      ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6014 031412 163737 001264 001404      SUB      2#SREG3,2#SAC1  ;SUB EXT EXPONENTS
6015                                     ;ACTUAL EXPONENTS ARE EQUAL
6016 031420 100002                                     BPL      15              ;MAKE SURE RESULT IS POSITIVE
6017 031422 005437 001404      NEG      2#SAC1
6018 031426 023737 001406 001404 15:      CMP      2#SAC2,2#SAC1  ;RESULTS WITHIN RANGE ALLOWED?
6019 031434 003401                                     BLE      SECT3          ;BRANCH IF YES
6020 031436 104014                                     ERROR    14              ;RESULTS WRONG

```

```

6021                                     ;*****
6022 SECT3:                                     ;LOAD OPERAND A
6023 031440 172537 001276      LDF      2#STMP0,AC1
6024 031444 172437 001202      LDF      2#STMP2,AC0      ;AND OPERAND B
6025 031450 013737 001256 001404      MOV      2#SREG0,2#SAC1
6026 031456 013737 001260 001402      MOV      2#SREG1,2#SAC0
6027 031464 004767 001224      JSR      PC,FLTDIV      ;GO DIVIDE THEM
6028 031470 004767 001176      JSR      PC,FLTMPY      ;MULTIPLY RESULT BY B
6029 031474 174137 001306      STF      AC1,2#STMP4      ;SAVE RESULT
6030 031500 013737 001404 001262      MOV      2#SAC1,2#SREG2
6031 031506 172437 001276      LDF      2#STMP0,AC0      ;LOAD OPERAND A
6032 031512 174037 001312      STF      AC0,2#STMP6      ;SAVE INCASE TYPE OUT
6033 031516 013737 001256 001402      MOV      2#SREG0,2#SAC0
6034 031524 013737 001256 001264      MOV      2#SREG0,2#SREG3
6035 031532 004767 001200      JSR      PC,FLTSUB      ;SUBTRACT RIGHT AND LEFT HAND SIDES
6036 031536 163737 001256 001404      SUB      2#SREG0,2#SAC1  ;SEE IF RESULT OK
6037 031544 100002                                     BPL      15              ;ENSURE DIFFERENCE IS POSITIVE
6038 031546 005437 001404      NEG      2#SAC1
6039 031552 022737 000027 001404 15:      CMP      #23.,2#SAC1     ;RESULTS WITHIN 2 BITS?
6040 031560 003001                                     BGT      25              ;BRANCH IF NO
6041 031562 000401                                     BR       TST73          ;GO TO NEXT TEST
6042 031564 104014                                     25:      ERROR    14              ;RESULTS WRONG

```

```

6043                                     ;*****
6044 *TEST 73      FLOATING POINT TEST 2
6045 *
6046 *      THIS TEST TAKES TWO RANDOM NUMBERS (A AND B) AND
6047 *      COMPARES THE RESULTS OF TWO EQUAL CALCULATIONS.
6048 *      EACH SECTION EVALUATES A DIFFERENT EQUATION AS DESCRIBED BELOW:
6049 *      SECT1      (A+B)**2=A**2+2*A*B+B**2
6050 *      SECT2      (A+B)*(A-B)=A**2-B**2
6051 *      SECT3      A/B*B=A
6052 *      *****

```

```

6053 TST73: MOV# 73,2#STSTNM      ;LOAD TEST NUMBER
6054 031566 112737 000073 001202      SCOPE
6055 031574 000004                                     MOV      #1,2#STIMES
6056 031576 012737 000001 001316      JSR      PC,2#FLTD8L     ;GET RANDOM OPERANDS
6057 031604 004737 050702 1005:      LDFPS   #200            ;INIT FPS
6058 031610 170127 000200      LDF      2#STMP0,AC1     ;LOAD A OPERAND
6059 031614 172537 001276      LDF      2#STMP4,AC0     ;LOAD B OPERAND
6060 031620 172437 001306      MOV      2#SREG0,2#SAC1  ;SETUP EXTENDED
6061 031624 013737 001256 001404      MOV      2#SREG1,2#SAC0  ;EXPONENTS
6062 031632 013737 001260 001402      JSR      PC,FLTADD      ;PERFORM THE ADD
6063 031640 004767 001076      STF      AC1,AC0        ;SETUP AC0 TO
6064 031644 174100

```



```

6066 031654 004767 001012 JSR PC,FLTMPY ;DO THE MULTIPLY
6067 031660 174137 001422 STF AC1,2#FLTMPD ;SAVE RESULT
6068 031664 013737 001404 001262 MOV 2#SAC1,2#SREG2 ;AND SOFTWARE EXP
6069
6070 ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
6071 ;DO THE A*A FIRST
6072 031672 013737 001256 001402 MOV 2#SREG0,2#SAC0 ;GET EXT EXPONENT
6073 031700 172437 001276 LDF 2#STMPD,AC0 ;LOAD OPERAND A
6074 031704 013737 001402 001404 MOV 2#SAC0,2#SAC1 ;SET OPERAND B EXT EXPONENT
6075 031712 172500 LDF AC0,AC1 ;LOAD B OPERAND
6076 031714 004767 000752 JSR PC,FLTMPY ;EXECUTE THE MULTIPLY
6077 031720 174102 STF AC1,AC2 ;SAVE RESULT
6078 031722 013737 001404 001406 MOV 2#SAC1,2#SAC2
6079
6080 ;NOW DO THE B*B
6081 031730 172437 001306 LDF 2#STMP4,AC0 ;LOAD B OPERAND
6082 031734 172500 LDF AC0,AC1
6083 031736 013737 001260 001402 MOV 2#SREG1,2#SAC0 ;AND EXT EXPONENT
6084 031744 013737 001402 001404 MOV 2#SAC0,2#SAC1
6085 031752 004767 000714 JSR PC,FLTMPY ;DO THE MULTIPLY
6086 031756 174103 STF AC1,AC3 ;SAVE THE RESULT
6087 031760 013737 001404 001410 MOV 2#SAC1,2#SAC3
6088
6089 ;NOW DO THE 2*B*A
6090 031766 012701 001306 MOV 2#STMP4,R1
6091 031772 172411 LDF (R1),AC0 ;LOAD THE B OPERAND
6092 031774 172541 LDF -(R1),AC1 ;LOAD THE A OPERAND
6093 031776 013737 001260 001402 MOV 2#SREG1,2#SAC0 ;AND THE EXT EXPONENTS
6094 032004 013737 001256 001404 MOV 2#SREG0,2#SAC1
6095 032012 004767 000654 JSR PC,FLTMPY ;DO THE MULTIPLY
6096 032016 172427 040000 LDF #1040000,AC0 ;SETUP TO MULTIPLY BY TWO
6097 032022 012737 000002 001402 MOV #2,2#SAC0
6098 032030 004767 000636 JSR PC,FLTMPY ;DO THE MULTIPLY
6099
6100 ;NOW SUM THE RESULTS
6101 032034 013737 001410 001402 MOV 2#SAC3,2#SAC0
6102 032042 172403 LDF AC3,AC0 ;GET RESULT OF B*B
6103 032044 004767 000672 JSR PC,FLTADD ;ADD THE RESULT
6104 032050 172402 LDF AC2,AC0 ;GET RESULT OF A*A
6105 032052 013737 001406 001402 MOV 2#SAC2,2#SAC0
6106 032060 004767 000656 JSR PC,FLTADD ;ADD THIS RESULT
6107 032064 174137 001432 STF AC1,2#FLTMP1 ;SAVE FINAL RESULT
6108 032070 013737 001404 001264 MOV 2#SAC1,2#SREG3
6109
6110 ;NOW CHECK BOTH SIDES OF THE EQUATION
6111 ;CALCULATE THE NUMBER OF CORRECT BITS
6112 ;PUT LARGEST EXPONENT OF A**2 OR B**2 IN SAC2
6113 032076 023737 001406 001410 CMP 2#SAC2,2#SAC3
6114 032104 002003 BGE 1$ ;BRANCH IF SAC2 ALREADY HAS LARGEST
6115 032106 013737 001410 001406 MOV 2#SAC3,2#SAC2 ;SAC3 WAS LARGER
6116 032114 163737 001404 001406 1$: SUB 2#SAC1,2#SAC2 ;NOW CALCULATE NUMBER
6117 032122 162737 000065 001406 SUB #53,2#SAC2 ;OF CORRECT BITS WITHIN 2
6118 032130 005437 001406 NEG 2#SAC2 ;MAKE RESULT POSITIVE
6119 032134 172437 001422 LDF 2#FLTMPD,AC0 ;LOAD RESULT OF LEFT HAND SIDE
6120 032140 013737 001262 001402 MOV 2#SREG2,2#SAC0 ;AND EXTENDED EXPONENT
    
```

```

6122 032152 163737 001264 001404 SUB 2#SREG3,2#SAC1 ;GET DIFFERENCE IN EXT EXPONENTS
6123 032160 100002 001404 BPL 35 ;ACTUAL EXP'S ARE EQUAL TO 200
6124 032162 005437 001404 NEG 2#SAC1 ;ENSURE RESULT IS POSITIVE
6125 032166 023737 001404 3S: CMP 2#SAC2,2#SAC1 ;ANSWERS WITHIN ALLOWABLE NUMBER?
6126 032174 003401 001406 4S: BLE SECT2D ;BRANCH IF YES
6127 032176 104016 001406 ERROR 16 ;RESULTS ARE WRONG
6128 *****
6129 SECT2D: LDFPS #200
6130 ;DO A+B
6131 LDF 2#STMP0,AC1 ;LOAD A OPERAND
6132 LDF 2#STMP4,AC0 ;LOAD B OPERAND
6133 MOV 2#SREG0,2#SAC1
6134 MOV 2#SREG1,2#SAC0
6135 JSR PC FLTA00 ;ADD THEM
6136 STF AC1,AC2 ;SAVE IN AC2
6137 MOV 2#SAC1,2#SAC2 ;AND EXT EXPONENT
6138 ;NOW DO THE A-B
6139 LDF 2#STMP0,AC1 ;LOAD OPERAND A
6140 MOV 2#SREG0,2#SAC1 ;AND EXT EXPONENT
6141 LDF 2#STMP4,AC0 ;LOAD OPERAND B
6142 MOV 2#SREG1,2#SAC0
6143 JSR PC FLTSUB ;SUBTRACT THEM
6144 ;NOW DO (A+B)*(A-B)
6145 LDF AC2,AC0 ;GET RESULT OF (A+B)
6146 MOV 2#SAC2,2#SAC0
6147 JSR PC FLTMPY ;FORM THE PRODUCT
6148 STF AC1,2#FLTMP0 ;SAVE RESULT
6149 MOV 2#SAC1,2#SREG2 ;AND EXT EXPONENT
6150 ;NOW DO THE B*B
6151 LDF 2#STMP4,AC0 ;LOAD OPERAND B
6152 MOV 2#SREG1,2#SAC0
6153 LDF AC0,AC1 ;B OPERAND IS IN AC0
6154 MOV 2#SAC0,2#SAC1 ;AND EXT EXPONENT
6155 JSR PC FLTMPY
6156 STF AC1,AC2 ;SAVE RESULT IN AC2
6157 MOV 2#SAC1,2#SAC2
6158 ;NOW DO THE A*A
6159 LDF 2#STMP0,AC0 ;LOAD OPERAND A
6160 MOV 2#SREG0,2#SAC0
6161 LDF AC0,AC1
6162 MOV 2#SAC0,2#SAC1
6163 JSR PC FLTMPY ;EXECUTE THE MULTIPLY
6164 MOV 2#SAC1,2#SAC3 ;SAVE EXT EXPO OF A*A
6165 ;NOW DO A**2-B**2
6166 LDF AC2,AC0 ;GET B*B
6167 MOV 2#SAC2,2#SAC0 ;A*A IN AC1
6168 JSR PC FLTSUB
6169 STF AC1,2#FLTMP1 ;SAVE IN MEMORY
6170 MOV 2#SAC1,2#SREG3
6171 ;NOW COMPUTE THE RESULTS
6172 ;CALCULATE THE NUMBER OF CORRECT BITS
6173 CMP 2#SAC2,2#SAC3 ;DETERMINE WHICH EXP IS LARGER
6174 BGE 25 ;BRANCH IF AC2 LARGER
6175 MOV 2#SAC3,2#SAC2 ;PUT LARGEST IN AC2
6176
    
```

```

6178 032466 162737 000066 001406 SUB 154,2#SAC2
6179 032474 005437 001406 NEG 2#SAC2
6180 032500 172437 001422 LDF 2#FLTMPD,AC0 ;GET LEFT HAND SIDE
6181 032504 013737 001262 001402 MOV 2#SREG2,2#SAC0
6182 032512 004767 000220 JSR PC,FLTSUB ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6183 032516 163737 001264 001404 SUB 2#SREG3,2#SAC1 ;SUB EXT EXPONENTS
6184 ;ACTUAL EXPONENTS ARE EQUAL
6185 032524 100002 BPL 15 ;MAKE SURE RESULT IS POSITIVE
6186 032526 005437 001404 NEG 2#SAC1
6187 032532 023737 001406 15: CMP 2#SAC2,2#SAC1 ;RESULTS WITHIN RANGE ALLOWED?
6188 032540 003401 BLE SECT30 ;BRANCH IF YES
6189 032542 104016 ERROR 16 ;RESULTS WRONG
6190
6191 ;*****
6192 032544 172537 001276 SECT30: LDF 2#STMPD,AC1 ;LOAD OPERAND A
6193 032550 172437 001306 LDF 2#STMP4,AC0 ;AND OPERAND B
6194 032554 013737 001256 001404 MOV 2#SREG0,2#SAC1
6195 032562 013737 001260 001402 MOV 2#SREG1,2#SAC0
6196 032570 004767 000120 JSR PC,FLTDIV ;GO DIVIDE THEM
6197 032574 004767 000072 JSR PC,FLTMPY ;MULTIPLY RESULT BY B
6198 032600 174137 001422 STF AC1,2#FLTMPD ;SAVE RESULT
6199 032604 013737 001404 001262 MOV 2#SAC1,2#SREG2
6200 032612 172437 001276 LDF 2#STMPD,AC0 ;LOAD OPERAND A
6201 032616 174037 001432 STF AC0,2#FLTMP1 ;SAVE INCASE TYPE OUT
6202 032622 013737 001256 001402 MOV 2#SREG0,2#SAC0
6203 032630 013737 001256 001264 MOV 2#SREG0,2#SREG3
6204 032636 004767 000074 JSR PC,FLTSUB ;SUBTRACT RIGHT AND LEFT HAND SIDES
6205 032642 163737 001256 001404 SUB 2#SREG0,2#SAC1 ;SEE IF RESULT OK
6206 032650 100002 BPL 15 ;ENSURE DIFFERENCE IS POSITIVE
6207 032652 005437 001404 NEG 2#SAC1
6208 032656 022737 000067 001404 15: CMP 2#SAC1,2#SAC1 ;RESULTS WITHIN 2 BITS?
6209 032664 003505 BLE RELEB ;BRANCH IF YES
6210 032666 104016 ERROR 16 ;RESULTS WRONG
6211 032670 000503 BR RELEB
6212
6213 ;*****
6214 ;SBTTL FLOATING POINT MULTIPLY ROUTINE
6215 ;* THIS ROUTINE MULTIPLIES THE CONTENTS OF AC0 AND AC1
6216 ;* AND LEAVES THE RESULT IN AC1. IT ALSO TAKES CARE OF
6217 ;* THE SOFTWARE EXPONENTS THAT ARE KEPT IN SAC0 AND SAC1.
6218 ;*****
6219 032672 063737 001402 001404 FLTMPY: ADD 2#SAC0,2#SAC1 ;ADD SOFTWARE EXPONENTS
6220 032700 171100 MULF AC0,AC1 ;DO THE MULTIPLY
6221 032702 012746 100400 MOV #100400,-(SP) ;PUT CONTROL WORD ON STACK
6222 032706 004737 051042 JSR PC,2#EXPEXT ;CALCULATE EXT EXPONENT
6223 032712 000207 15: RTS PC ;RETURN
6224
6225 ;*****
6226 ;SBTTL FLOATING POINT DIVIDE ROUTINE
6227 ;* THIS ROUTINE DIVIDES THE CONTENTS OF AC1 BY AC0
6228 ;* AND LEAVES THE RESULT IN AC1.
6229 ;*****
6230 032714 163737 001402 001404 FLTDIV: SUB 2#SAC0,2#SAC1 ;ADJUST SOFTWARE EXPONENTS
6231 032722 174500 DIVF AC0,AC1 ;EXECUTE THE DIVIDE
6232 032724 012746 100400 MOV #100400,-(SP) ;PUT CONTROL WORD ON STACK

```

```

6234 032734 000207      1$:      RTS      PC      ;RETURN
6235
6236      ;*****
6237      ;SBTTL  FLOATING POINT ADD ROUTINE
6238      ;*      THIS ROUTINE ADDS THE CONTENTS OF ACO TO AC1.
6239      ;*      THIS CAN ONLY BE DONE IF THE SOFTWARE EXPONENTS
6240      ;*      ARE CLOSE ENOUGH TOGETHER SUCH THAT AN ADJUSTMENT
6241      ;*      OF THE REAL EXPONENT LEAVES A NON-ZERO NUMBER.
6242      ;*****
6243 032736 010667 000134 001404 001404  FLTADD: MOV      SP,SUBFLG      ;SET SUBTRACT FLAG
6244 032742 023737 001402 001404  FLTADD: CMP      2(S),2(S)      ;CHECK SOFTWARE EXPONENTS
6245 032750 003016          BGT      1$
6246 032752 001434          BEQ      2$
6247      ;ACCUMULATOR 1 IS LARGER THAN ACCUMULATOR 0
6248 032754 013702 001404          MOV      2(S),R2      ;GET OPERAND B SOFTWARE EXP
6249 032760 163702 001402          SUB      2(S),R2      ;GET DIFFERENCE IN SOFTWARE EXP'S
6250 032764 020227 000071          CMP      R2,#57.      ;EXP WITHIN DBL PREC RANGE?
6251 032770 002003          BGE      7$          ;BRANCH IF ADD NOT REQUIRED
6252      ;RESULT IS OPERAND B
6253 032772 005402          NEG      R2
6254 032774 176402          LDEXP   R2,ACO      ;RELOAD THE EXPONENT
6255 032776 000422          BR      2$
6256 033000 176427 177703 7$:      LDEXP   #-75,ACO      ;FAKE EXPONENT SO HARDWARE
6257 033004 000417          BR      2$          ;WILL DETECT OUT OF RANGE
6258
6259      ;ACCUMULATOR 0 IS LARGER THAN ACCUMULATOR 1
6260 033006 013702 001402 001404 001404  1$:      MOV      2(S),R2      ;GET SOFTWARE EXP OF OPERAND A
6261 033012 163702 001404          SUB      2(S),R2      ;GET DIFFERENCE IN EXP'S
6262 033016 013737 001402          MOV      2(S),2(S)      ;MAKE SOFTWARE EXP'S EQUAL
6263 033024 020227 000071          CMP      R2,#57.      ;EXP WITHIN DBL PREC RANGE?
6264 033030 002003          BGE      4$          ;BRANCH IF NO
6265 033032 005402          NEG      R2
6266 033034 176502          LDEXP   R2,AC1      ;RELOAD THE EXPONENT
6267 033036 000402          BR      2$
6268
6269      ;ACCUMULATOR 0 IS MUCH LARGER THAN ACCUMULATOR 1 SO RESULT IS 0
6270 033040 176527 177703 4$:      LDEXP   #-75,AC1      ;FAKE EXPONENT SO HARDWARE
6271      ;WILL DETECT OUT OF RANGE
6272 033044 005767 000026 2$:      TST      SUBFLG
6273 033050 001402          BEQ      5$          ;ADD OR SUBTRACT?
6274 033052 173100          SUBF    ACO,AC1
6275 033054 000401          BR      6$
6276 033056 172100          ADDF    ACO,AC1      ;EXECUTE THE ADD
6277 033060 012746 100400 6$:      MOV      #100400,-(SP) ;PUT CONTROL WORD ON STACK
6278 033064 004737 051042          JSR     PC,2(EXP)EXT ;CALCULATE EXT EXPONENT
6279 033070 005067 000002 3$:      CLR      SUBFLG
6280 033074 000207          RTS     PC          ;INIT SUBTRACT FLAG
6281 033076 000000          SUBFLG: .WORD
6282 033100 000004          RELEB: SCOPE
6283 033102 010702          MOV     PC,R2
6284 033104 062702 000012          ADD     #12,R2
6285 033110 012707 034242          MOV     #RELOC,PC      ;GO RELOCATE PROGRAM CODE
6286 033114 000000          REL88: .WORD 0
6287      ;8888888888888888 LAST ADDRESS OF CODE TO BE RELOCATED 888888888888
6288

```

```

6290 ;*TEST 74 TELETYPE AND CLOCK TESTS
6291 ;*****
6292 033116 112737 000074 001202 †ST74: MOVB #74,‡STSTNM ;LOAD TEST NUMBER
6293 033124 000240 NOP
6294 033126 113737 001202 177570 MOVB ‡STSTNM,‡SWR
6295 033134 005037 001506 TTYCHK: CLR ‡FACTOR
6296 033140 012704 000100 MOV #100,R4 ;SET R4 = CONSTANT 100
6297 033144 032737 000400 001470 BIT ‡TOPT,‡OPT.CP ;BRANCH IF TTY
6298 033152 001002 BNE 1$ ;ON SYSTEM
6299 033154 000167 000204 JMP ARBFIN ;JUMP IF NOT
6300 033160 122777 000200 146056 1$: CMPB ‡200,‡STPS ;CHECK IF TTY IS READY
6301 033166 001374 BNE 1$
6302 033170 012737 001525 001270 MOV ‡NULLS-1,‡$REG5 ;SET ADDRESS OF ASCII STRING TO TYPE
6303 033176 106277 146042 ASRB ‡STPS ;SET IE BIT. SEE TPISR FOR INT SERVICE.
6304 033202 000001 WAIT ;WAIT FOR INTERRUPT
6305
6306
6307 033204 DUMMY:
6308 ;ROUTINE TO CHECK PRIORITY ARBITRATION LOGIC
6309 ;THE BELOW TEST WILL INHIBIT INTERRUPTS ON LEVEL 6 AND ABOVE (LOCKING
6310 ;OUT THE LINE CLOCK) AND THEN SET UP THE TTY TO INTERRUPT. NEXT THE
6311 ;PRIORITY LEVEL WILL BE SET TO 0 ALLOWING INTERRUPTS IN WHICH CASE
6312 ;THE LINE CLOCK (AT LEVEL 6) SHOULD INTERRUPT BEFORE THE TTY (AT LEVEL 4).
6313 033204 132737 000020 177776 1$: BITB #20,‡PSW
6314 033212 001071 BNE ARBEX ;EXIT TEST IF 'T' BIT SET
6315 033214 030477 146024 2$: BIT R4,‡STPS ;WAIT FOR TTY TO BE NOT
6316 033220 001375 BNE 2$ ;BUSY
6317 033222 112737 000300 177776 MOVB #300,‡PSW ;SET PRIORITY LEVEL 6
6318 033230 150477 146010 3$: BISB R4,‡STPS ;SET IE BIT
6319 033234 100375 BPL 3$ ;AND WAIT FOR READY
6320 033236 032737 001000 001470 BIT ‡LKOPT,‡OPT.CP ;LINE CLOCK AVAILABLE?
6321 033244 001447 BEQ ARBFIN ;BRANCH IF NO
6322 033246 012737 033336 000064 MOV #7$,‡TPVEC ;SET TTY VECTOR
6323 033254 012737 033350 000100 MOV #8$,‡LKVEC ;SET CLOCK VECTORS
6324 033262 012737 000340 000102 MOV #PR‡,‡LKVEC+2
6325 033270 005027 CLR (PC)+ ;CLEAR CHECK WORD
6326 033272 000000 4$: .WORD 0
6327 033274 000240 NOP
6328 033276 000240 NOP
6329 033300 000240 NOP
6330 033302 010437 177546 MOV R4,‡LKS
6331 033306 113700 5$: MOVB ‡(PC)+,R0 ;GET CLOCK STATUS & BRANCH IF TTDY
6332 033310 177546 6$: .WORD LKS ;CONTAINS ADDRESS OF L CLOCK STAT
6333 033312 100375 BPL 5$
6334 033314 000240 NOP ;AT THIS TIME BOTH THE CLOCK
6335 ;ARE READY TO INTERRUPT
6336 033316 105037 177776 CLRB ‡PSW ;SET PRIORITY LEVEL 0
6337 ;A CLOCK INTERRUPT WILL OCCUR (8$) AND LOC 4$ WILL BE INCREMENTED
6338 ;AFTER THE CLOCK SERVICE A TTY INTERRUPT WILL OCCUR. THE TTY INT SERV
6339 ;ICE WILL SHIFT LEFT 4$.
6340
6341 033322 022767 000002 177742 CMP #2,4$ ;CHECK THAT THE CLOCK
6342 033330 001415 BEQ ARBFIN ;& TTY INTERRUPTED IN
6343 033332 104000 HLT ;THE PROPER SEQUENCE
6344 033334 000413 BR ARBFIN

```



```

6402
6403
6404
6405
6406
6407
6408
6409
6410
6411
6412 033614 112737 000076 001202
6413 033622 000240
6414 033624 113737 001202 177570
6415 033632 032737 000100 177570
6416 033640 001402
6417 033642 000167 002450
6418
6419
6420
6421
6422
6423
6424 033646 013727 177776
6425 033652 000000
6426 033654 012737 000200 177776
6427 033662 004767 016452
6428
6429
6430 033666 012700 077406
6431 033672 010037 172300
6432 033676 010037 172302
6433 033702 010037 172304
6434 033706 010037 172306
6435 033712 010037 172310
6436 033716 010037 172312
6437 033722 010037 172316
6438
6439 033726 005037 172340
6440 033732 012737 000200 172342
6441 033740 012737 000400 172344
6442 033746 013737 001520 172346
6443 033754 013737 172346 172350
6444 033762 062737 000200 172350
6445 033770 013737 172346 172352
6446 033776 062737 000400 172352
6447 034004 012737 177600 172356
6448
6449 034012 010037 177600
6450 034016 010037 177602
6451 034022 010037 177604
6452 034026 010037 177616
6453 034032 016737 145462 177640
6454 034040 013737 177640 177642
6455 034046 062737 000200 177642
6456 034054 013737 177640 177644

```

```

.SBTTL STMM ROUTINE
ROUTINE TO SET UP MEMORY MANAGEMENT TO RELOCATE PROGRAM CODE ABOVE 16K
CHECK IF PROGRAM IS TO BE RELOCATED.
SW6=1=NO RELOCATION

*****PLEASE NOTE: THE RELOCATION ROUTINES WILL BE REFERED TO
AS TEST 76 IN ERROR TYPEOUTS DURING
RELOCATION.

*****
STMM:  MOVB  #76,2#STSTM ;LOAD TEST NUMBER
      NOP
      MOVB  2#STSTM,2#SWR
      BIT   #SW6,2#SWR ;RELOCATION DISABLED?
      BEQ  3$ ;BRANCH IF NO
      JMP  ENDM

;THE PROGRAM IS GOING TO RELOCATE.
;RELOCATION WILL BE PERFORMED IN KERNEL MODE WITH PSW SET AT PRIORITY
;LEVEL 4 (TO PREVENT TTY INTERRUPT-WHICH CHANGES DATA IN PROGRAM)
;THE 'T' BIT IS CLEARED (IF SET). AFTER THE DATA HAS BEEN WRITTEN IT IS
;VERIFIED BEFORE EXECUTION.
3$:  MOV  2#PSW,(PC)+ ;SAVE CURRENT PSW
OLDPSW: .WORD 0
      MOV  #PR4,2#PSW
      JSR  PC,CLRTBIT ;CO CLEAR 'T' BIT IF SET

;NOW SETUP MEMORY MANAGEMENT REGISTERS
      MOV  #77406,RO ;SET CONSTANT=R/W UP 4K WORDS
      MOV  RO,2#KIPOR0 ;SET KIPOR0,1,2,3,& 7 R/W UP 4K WORDS
      MOV  RO,2#KIPOR1
      MOV  RO,2#KIPOR2
      MOV  RO,2#KIPOR3
      MOV  RO,2#KIPOR4
      MOV  RO,2#KIPOR5
      MOV  RO,2#KIPOR7
      CLR  2#KIPAR0 ;NOTE: THESE 2 INSTRUCTIONS EFFECTIVELY
      MOV  #200,2#KIPAR1 ;RELOCATE PROGRAM EXECUTION
      MOV  #400,2#KIPAR2
      MOV  2#NEXPAR,2#KIPAR3 ;SET UP KIPAR3 & KIPAR4 & 5
      MOV  2#KIPAR3,2#KIPAR4
      ADD  #200,2#KIPAR4
      MOV  2#KIPAR3,2#KIPAR5
      ADD  #400,2#KIPAR5
      MOV  #177600,2#KIPAR7;AND OF COUSE THE I/O PAGE

;NOW SETUP USER MEM MGMT REGISTERS
3$:  MOV  RO,2#UIPOR0 ;SET UP USER MEM MGMT REGS
      MOV  RO,2#UIPOR1
      MOV  RO,2#UIPOR2
      MOV  RO,2#UIPOR7
      MOV  NEXPAR,2#UIPAR0
      MOV  2#UIPAR0,2#UIPAR1
      ADD  #200,2#UIPAR1
      MOV  2#UIPAR0,2#UIPAR2

```

```

6458 034070 013737 172356 177656      MOV      @#KIPAR7,@#UIPAR7
6459
6460 034076 010037 172200      MOV      R0,@#SIPDR0      ;SET UP SUPERVISOR MEM MGMT REGS
6461 034102 010037 172202      MOV      R0,@#SIPDR1
6462 034106 010037 172204      MOV      R0,@#SIPDR2
6463 034112 010037 172216      MOV      R0,@#SIPDR7
6464 034116 016737 145376 172240      MOV      NEXPAR,@#SIPAR0
6465 034124 013737 172240 172242      MOV      @#SIPAR0,@#SIPAR1
6466 034132 062737 000200 172242      ADD      @200,@#SIPAR1
6467 034140 013737 172240 172244      MOV      @#SIPAR0,@#SIPAR2
6468 034146 062737 000400 172244      ADD      @400,@#SIPAR2
6469 034154 013737 172356 172256      MOV      @#KIPAR7,@#SIPAR7
6470 034162 012737 000001 172572      MOV      #1,@#SR0      ;ENABLE MEM MGMT
6471 034170 012737 000060 172516      MOV      #60,@#SR3      ;SETUP SR3
6472 034176 110637 001503      MOV      SP,@#MMON      ;SET MEM MGMT ON IND = ON
6473 C34202 005037 000006      RETRY:   CLR      @#ERRVEC+2
6474 034206 012737 036314 000004      MOV      @ENDMEM,@#ERRVEC ;SET TIME OUT TRAP VECTOR
6475 034214 012702 060000      MOV      #60000,R2      ;SETUP GENERAL REGISTERS
6476 034220 005000      CLR      R0      ;DATA WILL BE RELOCATED FROM
6477                                     ;ADDRESS IN R0 TO ADDRESS IN R2
6478 034222 012703 137776      MOV      #137776,R3      ;GET 12K WORDS TO RELOCATE
6479 034226 010013      MOV      R0,(R3)      ;TRAP TO ENDMEM IF INSUFFICIENT MEMORY
6480 034230 012737 053442 000004      MOV      @ERRPT,@#ERRVEC ;RESTORE ERROR TRAP VECTOR
6481 034236 000137 034436      JMP      @#IOMON
6482
6483      ;*****
6484      ;SBTTL RELOCATION ROUTINE
6485      ;THIS ROUTINE IS USED TO RELOCATE THE 9 SUBTESTS UP TO 28K.
6486      ;IF RELOCATION BY AN I/O DEVICE IS SELECTED, CONTROL IS PASSED
6487      ;TO THE I/O MONITOR.
6488      ;ENTER WITH:
6489      ;FRSTAD=PHYSICAL ADDRESS OF FIRST CODE
6490      ;FACTOR=NUM ER OF BYTES ABOVE BASE CODE
6491      ;R2 =LAST PHYSICAL ADDRESS OF THE SECTION
6492      ;EXIT TO I/O MONITOR WITH:
6493      ;OLDBASE=FIRST PHYSICAL ADDRESS TO BE RELOCATED
6494      ;NWBASL =FIRST PHYSICAL ADDRESS TO RELOCATE TO
6495      ;IOWC =TWO'S COMPLIMENT WORD COUNT
6496      ;*****
6496 034242 032737 000100 177570 RELOC:   BIT      @SW6,@#SWR      ;IS RELOCATION DISABLED?
6497 034250 001067      BNE      EXITRE      ;BRANCH IF YES
6498 034252 105737 001503      TSTB     @#MMON      ;IS MEMORY MGMT ON?
6499 034256 001064      BNE      EXITRE      ;BRANCH IF YES
6500 034260 013700 001512      MOV      @#FRSTAD,R0      ;GET FIRST ADDRESS TO BE RELOCATED
6501 034264 010005      MOV      R0,R5
6502                                     ;LAST ADDRESS IS IN R2
6503 034266 010203      MOV      R2,R3      ;SAVE LAST ADDRESS
6504 034270 010204      MOV      R2,R4
6505 034272 160004      SUB      R0,R4      ;R4 NOW HAS BYTE COUNT
6506 034274 010437 001510      MOV      R4,@#SFACOR      ;SAVE BYTE COUNT
6507 034300 005737 001506      TST      @#FACTOR      ;FIRST RELOC IS TO ENDTAG+2
6508 034304 001004      BNE      IS      ;BRANCH IF NOT EXECUTING BASE CODE
6509 034306 010237 034434      MOV      R2,@#RETPC      ;SAVE RETURN PC TO NEXT SECTION
6510 034312 013702 001514      MOV      @#FRSTMEM,R2      ;GET FIRST ADDRESS TO RELOCATE TO
6511 034316 060204      IS:      ADD      R2,R4      ;R4 NOW CONTAINS LAST MEM ADDRESS
6512 034320 020437 001516      CMP      R4,@#LSTMEM      ;ENOUGH MEMORY?

```



```

6514 034326 160204          SUB      R2,R4          ;R4 NOW HAS BYTE COUNT
6515 034330 005037 001506      CLR      @#FACTOR
6516 034334 032737 000400 177570  BIT      @SWB,@SWR      ;INHIBIT RELOC BY I/O DEVICE?
6517 034342 001014          BNE      RELNIO        ;BRANCH IF YES
6518 034344 010037 001540      MOV      R0,@#OLDBASE ;SAVE START ADDRESS
6519 034350 010237 001542      MOV      R2,@#NWBASL  ;SAVE NEW BASE ADDRESS
6520 034354 005037 001544      CLR      @#NWBASH
6521 034360 006204          ASR      R4            ;MAKE IT A WORD COUNT
6522 034362 005404          NEG      R4            ;GET TWO'S COMPLIMENT
6523 034364 010437 001546      MOV      R4,@#IOWC    ;SAVE R4 AS WORDCOUNT
6524 034370 000167 000120      JMP      ENTER2        ;GO TO I/O MONITOR
6525                                     ;RELOCATE BY CPU-MEMORY MANAGEMENT OFF
6526 034374 012022  RELNIO: MOV      (R0)+,(R2)+ ;RELOCATE CODE
6527 034376 020003          CMP      R0,R3         ;DONE YET?
6528 034400 001375          BNE      RELNIO        ;BRANCH IF NO
6529 034402 004737 051746      JSR      PC,@#CHKDAT  ;GO CHECK DATA
6530 034404 102010          BVC      EXITRE
6531 034410 010037 001276      MOV      R0,@#STMP0    ;SAVE R0 FOR TYPEOUT
6532 034414 010237 001474      MOV      R2,@#VADR     ;SAVE R2
6533 034420 004737 051644      JSR      PC,@#CNVADR   ;CONVERT R2 TO A PHYSICAL ADR
6534 034424 104006          ERROR   6
6535 034426 000401          BR      NOMEM
6536 034430 010207  EXITRE: MOV      R2,PC      ;GO EXECUTE RELOCATED CODE
6537 034432 011707  NOMEM:  MOV      (PC),PC   ;GO TO NEXT SECTION
6538 034434 000000  RETPC:  .WORD  0         ;CONTAINS PC OF NEXT SECTION
6539                                     ;*****
6540                                     ;SBTTL I/O RELOCATION MONITOR
6541                                     ;* THIS ROUTINE IS USED TO SCHEDULE I/O DEVICES FOR SUBTEST
6542                                     ;* RELOCATION AND PROGRAM RELOCATION. THE I/O DEVICE UNIT
6543                                     ;* NUMBER IS DETERMINED, THE BUS ADDRESS CALCULATED, THE WORD
6544                                     ;* COUNT CALCULATED AND PASSED TO THE DEVICE HANDLER.
6545                                     ;*****
6546 034436 012737 034444 001212  IOMON: MOV      @1$,@#SLPERR ;SETUP ERROR LOOP
6547 034444 005037 001540      CLR      @#OLDBASE
6548 034450 013705 172346      MOV      @#KIPAR3,R5
6549 034454 005004          CLR      R4
6550 034456 073427 000006      ASHC    @6,R4          ;TO FORM 22 BIT ADDRESS
6551 034462 010537 001542      MOV      R5,@#NWBASL  ;FORM 22 BIT ADDRESS
6552 034466 010437 001544      MOV      R4,@#NWBASH  ;SAVE LOWER 16 BITS
6553 034472 032737 000400 177570  BIT      @SWB,@SWR      ;SAVE UPPER 6 BITS
6554 034500 001402          BEQ      2$           ;RELOCATE VIA I/O?
6555 034502 000167 001436      JMP      RELOCP       ;BRANCH IF YES
6556 034506 012737 174000 001546  2$:  MOV      @174000,@#IOWC ;GO RELOCATE VIA CP
6557 034514 005037 001302  ENTER2: CLR      @#STMP2    ;SET WORD COUNT TO 2K
6558 034520 012737 177776 001556      MOV      @-2,@#RNTBINX ;SETUP RUN TABLE INDEX
6559 034526 005037 001276      CLR      @#STMP0
6560 034532 005002          CLR      R2
6561 034534 032737 000040 177570  41$: BIT      @SW5,@SWR      ;CLEAR LEGAL DEV FLAG
6562 034542 001416          BEQ      50$           ;INHIBIT ROUND ROBIN?
6563 034544 005737 001276      TST      @#STMP0
6564 034550 001027          BNE      43$           ;BRANCH IF NO
6565 034552 113737 177570 001552      MOVB    @#SWR,@#DEVINDX ;FLAG SET?
6566 034560 042737 177770 001552      BIC     @177770,@#DEVINDX ;BRANCH IF YES
6567 034566 006337 001552      ASL     @#DEVINDX      ;GET DEVICE FROM SWITCHES
6568 034572 005237 001276      INC     @#STMP0        ;MASK LOWER 3 BITS
                                           ;ADJUST FOR WORD INDEX
                                           ;SET FLAG

```

6570	034600	012705	000010		50\$:	MOV	#10,R5	:	SET SOB COUNT
6571	034604	022737	000016	001552	40\$:	CMP	#16,#DEVINDX	:	LAST DEVICE YET?
6572	034612	001003				BNE	42\$	:	BRANCH IF NO
6573	034614	012737	177776	001552	48\$:	MOV	#-2,#DEVINDX	:	INIT DEVICE INDEX
6574	034622	062737	000002	001552	42\$:	ADD	#2,#DEVINDX	:	INCREMENT INDEX
6575	034630	013703	001552		43\$:	MOV	#DEVINDX,R3	:	GET INDEX
6576	034634	012737	000401	001300		MOV	#401,#STMP1	:	INIT UNIT MASK
6577	034642	012704	000010			MOV	#10,R4	:	SET SOB COUNT
6578	034646	133763	001300	001606	44\$:	BITB	#STMP1,SYSSIZE(R3)	:	IS THIS UNIT EXISTENT?
6579	034654	001406				BEQ	52\$	:	BRANCH IF NO
6580	034656	005202				INC	R2	:	SET LEGAL DEVICE FLAG
6581	034660	133763	001301	001607		BITB	#STMP1+1,SYSSIZE+1(R3)	:	HAS IT BEEN USED?
6582	034666	001520				BEQ	11\$	:	BRANCH IF NO
6583	034670	006337	001300		52\$:	ASL	#STMP1	:	SELECT NEXT UNIT
6584	034674	077414				SOB	R4,44\$	:	CONTINUE
6585	034676	075737	001276			TST	#STMP0	:	INHIBIT ROUND ROBIN?
6586	034702	001013				BNE	45\$	:	BRANCH IF YES
6587	034704	077541				SOB	R5,40\$	:	CONTINUE
6588	034706	005702				TST	R2	:	ANY DEVICES AT ALL?
6589	034710	001442				BEQ	46\$	:	BRANCH IF NO
6590	034712	012704	000010			MOV	#10,R4	:	SET SOB COUNT
6591	034716	012701	001607			MOV	#SYSSIZE+1,R1	:	GET ADR OF SIZE TABLE
6592	034722	105021			47\$:	CLRB	(R1)+	:	CLEAR ALL USED BITS
6593	034724	005201				INC	R1	:	IN ALL DEVICES
6594	034726	077403				SOB	R4,47\$	:	CONTINUE
6595	034730	000701				BR	41\$	:	
6596	034732	005702			45\$:	TST	R2	:	HAS IT A LEGAL DEVICE?
6597	034734	001403				BEQ	49\$	:	BRANCH IF NO
6598	034736	105063	001607			CLRB	SYSSIZE+1(R3)	:	CLEAR ALL USED BITS THIS DEV
6599	034742	000732				BR	43\$	:	
6600	034744	010367	000016		49\$:	MOV	R3,60\$	:	
6601	034750	062767	053612	000010		ADD	#MSGINX,60\$	:	GEN MESSAGE ADR
6602	034756	017767	000004	000002		MOV	#60\$,60\$	:	
6603	034764	104400				TYPE		:	
6604	034766	000000			60\$:	.WORD		:	
6605	034770	104400	034776			TYPE		:	
6606	034774	000407				BR	65\$	:	TYPE ASCIZ STRING
6607						BR	64\$	:	GET OVER THE ASCIZ
6608	035014				65\$:	.ASCIZ	/UNAVAILABLE/<<CRLF>	:	
6609	035014	000637			64\$:			:	
6610	035016	105737	001504		46\$:	BR	ENTER2	:	
6611	035022	001016				TSTB	#OV	:	ACT11?
6612	035024	005227	177777			BNE	51\$	:	BRANCH IF YES
6613	035030	001013				INC	#-1	:	
6614	035032	104400	035040			BNE	51\$	:	
6615	035036	000410				TYPE	67\$	:	TYPE ASCIZ STRING
6616						BR	66\$	:	GET OVER THE ASCIZ
6617	035060				67\$:	.ASCIZ	?NO I/O DEVICES?<<CRLF>	:	
6618	035060	105737	001503		66\$:			:	
6619	035064	001012			51\$:	TSTB	#MMON	:	MGMT ON?
6620	035066	013700	001540			BNE	61\$	:	BRANCH IF YES
6621	035072	013702	001542			MOV	#OLDBASE,R0	:	RESTORE R0
6622	035076	013703	001510			MOV	#NMBASL,R2	:	RESTORE R2
6623	035102	060003				MOV	#SFACOR,R3	:	GET RELOCATION FACTOR
6624	035104	010005				ADD	R0,R3	:	FORM LAST ADDRESS
						MOV	R0,R5	:	SETUP R5

6626	035112	012702	060000		61\$:	MOV	#60000,R2	:	SETUP REGISTERS
6627	035116	012703	137776			MOV	#137776,R3	:	WITH FROM
6628	035122	005000				CLR	R0	:	AND TOO ADDRESS
6629	035124	000137	036144			JMP	@RELOCP	:	RELOCATE VIA CP
6630	035130	105763	001676		11\$:	TSTB	RP3HSTAT(R3)	:	IS HANDLER BUSY?
6631	035134	100405				BMI	R5	:	BRANCH IF NO
6632	035136	005737	001276			TST	@STMP0	:	ROUND ROBIN?
6633	035142	001372				BNE	11\$	:	BRANCH IF NO
6634	035144	000167	177364			JMP	41\$		
6635	035150	005763	001676		8\$:	TST	RP3HSTAT(R3)	:	DID HANDLER FAIL?
6636	035154	100005				BPL	62\$	:	BRANCH IF NO
6637	035156	005737	001276			TST	@STMP0	:	ROUND ROBIN
6638	035162	001402				BEQ	62\$	:	BRANCH IF YES
6639	035164	000137	036124			JMP	@15\$		
6640	035170	153763	001301	001607	62\$:	BISB	@STMP1+1,SYSSIZE+1(R3)	:	SET UNIT USED BIT
6641	035176	005002				CLR	R2		
6642	035200	006037	001300		30\$:	ROR	@STMP1	:	ENCODE THE BIT POSITION
6643	035204	005202				INC	R2	:	INTO A UNIT NUMBER
6644	035206	103374				BCC	30\$		
6645	035210	005302				DEC	R2		
6646	035212	010237	001554			MOV	R2,@UNITNO	:	SAVE UNIT NUMBER
6647	035216	013763	001546	001716	10\$:	MOV	@10WC,RP3HWC(R3)	:	GIVE WORD COUNT TO HANDLER
6648	035224	010304				MOV	R3,R4		
6649	035226	072427	000003			ASH	#3,R4	:	ENCODE DEVICE FOR RUNTABLE
6650	035232	053704	001554			BIS	@UNITNO,R4	:	ENCODE UNIT NUMBER
6651	035236	006304				ASL	R4		
6652	035240	062737	000002	001556		ADD	#2,@RANTBINX	:	INCREMENT RUN TABLE INDEX
6653	035246	013702	001556			MOV	@RANTBINX,R2	:	GET RUN TABLE INDEX
6654	035252	110462	001627			MOV	R4,RUNTABL+1(R2)	:	ENTER DEV & UNIT IN TABLE
6655	035256	013763	001554	002012		MOV	@UNITNO,RP3UNIT(R3)	:	GIVE HANDLER UNIT NUMBER
6656	035264	012737	000240	000012		MOV	@PRS,@RESVEC+2	:	SETUP RESERVED VECTOR PSW
6657	035272	016337	002026	000010		MOV	RP3HANA(R3),@RESVEC	:	SETUP RESERVED VECTOR
6658	035370	006303				ASL	R3	:	ADJUST INDEX
6659	035362	013763	001540	001732		MOV	@OLDBASE,RP3OLD(R3)	:	GIVE HANDLER OLD BASE ADDRESS
6660	035310	013763	001542	001762		MOV	@NEWASL,RP3NWL(R3)	:	GIVE HANDLER
6661	035316	013763	001544	001764		MOV	@NEWASH,RP3NWH(R3)	:	NEW BASE ADDRESS
6662	035324	005063	001734			CLR	RP3OLD+2(R3)	:	ENSURE OLD BASE HIGH IS CLR
6663	035330	000210				CALL	HANDLER		
6664	035332	105737	001503			TSTB	@MMON	:	IS MEMORY MANAGEMENT ON?
6665	035336	001416				BEQ	13\$	:	BRANCH IF NO
6666	035340	022737	000012	001556		CMF	#12,@RANTBINX	:	TRANSFERED 12K YET?
6667	035344	001412				BEQ	13\$	:	BRANCH IF YES
6668	035350	062737	010000	001540		ACD	#10000,@OLDBASE	:	ADD 2K
6669	035356	062737	010000	001542		ADD	#10000,@NEWASL	:	TO BASE
6670	035364	005537	001544			ADC	@NEWASH	:	ADDRESSES
6671	035370	000137	034534			JMP	@41\$		
6672	035374	113705	001603		13\$:	MOVB	@LTICKS+1,R5	:	GET SECOND COUNT
6673	035400	062705	000002			ADD	#2,R5	:	INCREMENT BY TWO
6674	035404	162705	000074			SUB	#60.,R5	:	ENSURE RESULT IS 59 OR LESS
6675	035410	100002				BPL	31\$		
6676	035412	062705	000074			ADD	#60.,R5	:	COUNT WAS LESS THAN 58-RESTORE
6677	035416	012700	000010		31\$:	MOV	#10,R0	:	SET SOB COUNT
6678	035422	005002				CLR	R2		
6679	035424	005003				CLR	R3		
6680	035426	005004				CLR	R4		

```

6682 035434 005504          ADC      R4
6683 035436 062702 000002  ADD      #2,R2
6684 035442 077006          SOB      R0,14$
6685 035444 006103          ROL      R3
6686 035446 005504          ADC      R4
6687 035450 022703 004000  CMP      #4000,R3
6688 035454 001406          BEQ      32$
6689 035456 123705 001603  CMPB    @#LTICKS+1,R5
6690 035462 001355          BNE      31$
6691 035464 104015          ERROR   15
6692 035466 000177 143520  JMP      @#LPERR
6693 035472 005704 32$:      TST      R4
6694 035474 001402          BEQ      82$
6695 035476 000167 000422  JMP      15$
6696 035502 105737 001503 82$:      TSTB    @#MMON
6697 035506 001012          BNE      25$
6698 035510 013705 001540  MOV      @#OLDBASE,R5
6699 035514 010500          MOV      R5,R0
6700 035516 063700          ADD      @#$FACTOR,R0
6701          ; GET LAST ADDRESS
6702          ; OF GOOD DATA
6703 035522 013702 001542  MOV      @#NWBASL,R2
6704 035526 063702 001510  ADD      @#$FACTOR,R2
6705          ; GET LAST ADDRESS
6706          ; OF DATA TO BE CHECKED
6707          ; CONTINUE
6708 035532 000406 057776 25$:      BR       22$
6709 035534 012700 137776          MOV      #57776,R0
6710 035540 012702 002100          MOV      #137776,R2
6711 035544 012705 051746          MOV      #2100,R5
6712 035550 004737 22$:      JSR      PC,@#CHKDAT
6713 035554 102413          BVS      81$
6714 035556 105737 001204  TSTB    @#SERFLG
6715 035562 001002          BNE      83$
6716 035564 000167 000412          JMP      EXIT
6717 035570 032737 001000 177570 83$:     BIT      #549,@#SWR
6718 035576 001473          BEQ      100$+2
6719 035600 000167 000244          JMP      20$
6720 035604 005001 81$:      CLR      R1
6721 035606 010037 001276          MOV      R0,@#STMPD
6722 035612 010237 001474          MOV      R2,@#VAOR
6723 035616 010203          MOV      R2,R3
6724 035620 005004          CLR      R4
6725 035622 105737 001503  TSTB    @#MMON
6726 035626 001406          BEQ      16$
6727 035630 162703 010000 17$:     SUB      #10000,R3
6728 035634 100403          BMI      16$
6729 035636 062704 000002  ADD      #2,R4
6730 035642 000772          BR       17$
6731          ; R4 NOW CONTAINS INDEX OF ERROR FOR RUN
6732          ; TIME TABLE
6733 035644 116404 001627 16$:     MOVB    RUNTBL+1(R4),R4
6734 035650 042704 177400          BIC      #177400,R4
6735 035654 006004          ROR      R4
6736 035656 005005          CLR      R5
6737 035660 073427 177775          ASHC    #-3,R4
6738 035664 010500          MOV      R5,R0
6739 035666 072027 177764          ASH     #-14,R0
6740 035672 042700 177770          BIC     #177770,R0

```

```

; STATUS WORDS. WHEN ALL
; TRANSFERS ARE FINISHED
; RESULT WILL BE 2000
; (WITHOUT ROTATE)
; ALL DONE?
; BRANCH IF YES
; TWO SECONDS ELAPSED YET?
; BRANCH IF NO
; DEVICE HUNG
; RESTART RELOCATION
; ANY DEVICE ERRORS?
; BRANCH IF NO
; ERROR
; MEM MGMT ON?
; BRANCH IF YES
; SETUP R5 FOR DATA CHECK
; GET LAST ADDRESS
; OF GOOD DATA
; GET LAST ADDRESS
; OF DATA TO BE CHECKED
; CONTINUE
; GET LAST ADR OF GOOD DATA
; GET LAST ADR OF DATA TO BE CHECKED
; DON'T CHECK FIRST 2100 LOCATIONS
; GO CHECK DATA
; BRANCH IF ERROR
; ANY ERRORS?
; BRANCH IF YES
; RETURN
; LOOP ON ERROR?
; BRANCH IF NO
; GO DO FUNCTION AGAIN
; SAVE ERROR ADDRESS
; IS MEM MGMT ON?
; BRANCH IF NO
; SUBTRACT 2K FROM ERROR ADDRESS
; BRANCH IF BLOCK IS FOUND
; COUNT ONE MORE BLOCK
; CONTINUE
; GET DEVICE THAT FAILED
; ENSURE HIGH BYTE CLEAR
; THROW AWAY LSB
; ENSURE R5 CLEAR
; GET UNIT NUMBER IN R5

```

6738	035702	010403				MOV	R4,R3		;AND DEVICE INDEX IN R4 & R3
6739	035704	010337	001302			MOV	R3,@STMP2		
6740	035710	012737	000001	001300		MOV	#1,@STMP1		;ENCODE 3 BIT UNIT NO INTO
6741	035716	162705	020000		19\$:	SUB	#20000,R5		;ONE BIT IN THE LOW BYTE OF STMP1
6742	035722	103403				BCS	18\$		;BRANCH IF DONE
6743	035724	006137	001300			ROL	@STMP1		;SELECT NEXT UNIT
6744	035730	000772				BR	19\$		;CONTINUE
6745	035732	012737	036050	001212	18\$:	MOV	#20\$,@SLPERR		;SETUP LOOP RETURN
6746	035740	005701				TST	R1		;DEVICE ERROR?
6747	035742	001010				BNE	100\$		;BRANCH IF YES
6748	035744	104010				ERROR	10		;DATA CHECK ERROR
6749	035746	105737	001503			TSTB	@MMON		;MGMT ON?
6750	035752	001002				BNE	70\$		;BRANCH IF YES
6751	035754	000137	034514		71\$:	JMP	@ENTER2		
6752	035760	000137	034436		70\$:	JMP	@IOMON		
6753	035764	104007			100\$:	ERROR	7		
6754	035766	042763	100000	001676		BIC	#BIT15,RP3HSTAT(R3)		;CLEAR THE ERROR
6755	035774	022703	000002			CMP	#2,R3		;RK05 ERROR?
6756	036000	002405				BLT	90\$		;BRANCH IF RH70
6757	036002	003016				BGT	92\$		;BRANCH IF RPO3
6758	036004	112777	000001	144114		MOVB	#1,@RKCS		;RK CONTROLLER CLEAR
6759	036012	000412				BR	92\$		
6760	036014	022703	000012		90\$:	CMP	#12,R3		;RS04?
6761	036020	001004				BNE	91\$		;BRANCH IF NO
6762	036022	052777	000040	144164		BIS	#BIT5,@RSCS2		;CLEAR RS CONTROLLER
6763	036030	000403				BR	92\$		
6764	036032	052777	000040	144114	91\$:	BIS	#BIT5,@RP4CS2		;CLEAR RPO4 CONTROLLER
6765	036040	105737	001503		92\$:	TSTB	@MMON		;MGMT ON?
6766	036044	001345				BNE	70\$		;BRANCH IF YES
6767	036046	000742				BR	71\$		
6768	036050	052763	000400	001676	20\$:	BIS	#BIT8,RP3HSTAT(R3)		;SET REPEAT FLAG IN HANDLER
6769	036056	016337	002026	000010		MOV	RP3HANA(R3),@RESVEC		;SETUP RESERVED INSTRUCTION VECTOR
6770	036064	000010				CALLHANDLER			
6771	036066	105763	001676		21\$:	TSTB	RP3HSTAT(R3)		;HANDLER FINISHED?
6772	036072	100375				BPL	21\$		;BRANCH IF NO
6773	036074	005763	001676			TST	RP3HSTAT(R3)		;ANY ERROR?
6774	036100	100714				BMI	18\$		;BRANCH IF YES
6775	036102	005701				TST	R1		;DEVICE ERROR?
6776	036104	001002				BNE	80\$		;BRANCH IF YES
6777	036106	000167	177364			JMP	32\$+4		;GO CHECK DATA
6778	036112	032737	001000	177570	80\$:	BIT	#BIT9,@SWR		;STILL LOOPING?
6779	036120	001353				BNE	20\$		;BRANCH IF YES
6780	036122	000721				BR	100\$+2		;CONTINUE TEST
6781									ON RELOCATION
6782	036124	005004							;SET INDEX
6783	036126	010601			15\$:	CLR	R4		
6784	036130	005764	001642		24\$:	MOV	SP,R1		;SEARCH FOR DEVICE ERROR
6785	036134	100643				TST	RUNTRAK(R4)		;BRANCH IF ERROR
6786	036136	062704	000002			BMI	16\$		;INCREMENT INDEX
6787	036142	000772				ADD	#2,R4		;CONTINUE SEARCH
6788						BR	24\$		
6789	036144	012022							;RELOCATE BY CPU-MEMORY MANAGEMENT ON
6790	036146	020302			RELOCP:	MOV	(R0)+,(R2)+		;RELOCATE CODE
6791	036150	001375				CMP	R3,R2		;DONE YET?
6792	036152	012705	001700			BNE	RELOCP		;BRANCH IF NO
						MOV	#1700,R5		

```

6794 036162 102007
6795 036164 010037 001276
6796 036170 010237 001474
6797 036174 104006
6798 036176 000167 176000
6799 036202 105737 001503
6800 036206 001002
6801 036210 000137 034430
6802 036214 062737 000077 001520
6803 036222 013737 172346 172340
6804 036230 013737 172350 172342
6805 036236 013737 172352 172344
6806
6807
6808
6809
6810 036244 013700 172340
6811 036250 072027 177771
6812 036254 110037 001203
6813 036260 012706 001200
6814 036264 005037 177776
6815 036270 016746 175356
6816 036274 012746 005372
6817 036300 105737 001502
6818 036304 001402
6819 036306 012716 033614
6820 036312 000002
6821
6822
6823 036314 022626
6824 036316 005037 177572
6825 036322 042737 000020 172516
6826
6827
6828
6829
6830 036330 012737 000600 001520
6831 036336 005737 003244
6832 036342 001403
6833 036344 012737 001600 001520
6834 036352 105037 001503
6835
6836
6837
6838
6839
6840
6841 036356
6842 036356 012737 053442 000004
6843 036364 005037 177776
6844 036370 004767 013744
6845 036374 012706 001200
6846 036400 032777 000100 142636
6847 036406 001374
6848 036410 105237 001532

```

```

EXIT: BVC EXIT
MOV R0, @STMPD
MOV R2, @VADR
ERROR 6
JMP RETRY
TSTB @MMON ; MEM MGMT ON?
BNE +6 ; BRANCH IF YES
JMP @EXITRE
ADD #77, @NEXPAR ; SET VALUE FOR NEXT RELOCATION
MOV @KIPAR3, @KIPAR0
MOV @KIPAR4, @KIPAR1
MOV @KIPAR5, @KIPAR2
*****
: PROGRAM IS NOW EXECUTING IN KERNEL MODE RELOCATED TO ADDRESS AS SPEC-
: IFIED IN KIPAR0. FOR EX. IF KIPAR0=1600 THEN PROGRAM EXECUTING AT
: ADDRESS 160000+(PC)
MOV @KIPAR0, R0 ; GET PAR0
ASH #7, R0 ; GET BITS <14:7> IN LOW BYTE
MOVB R0, @STSTNM+1 ; PUT IN DSPLY REG HIGH BYTE
MOV @KERSTK, SP ; SET KERNEL STACK PTR
CLR @PSW
MOV OLDPSW, -(SP) ; RESTORE OLD PSW
MOV @LOOP, -(SP)
TSTB @NEXEC ; BRANCH IF TEST CODE TO
BEQ 1$ ; BE EXECUTED
MOV @STMM, (SP)
1$: RTI ; RESTART PROGRAM AT LOOP

: WHEN RELOCATION ABOVE 28K IS COMPLETE PROGRAM TRAPS TO ENDMEM.
ENMEM: CMP (SP)+, (SP)+ ; POP STACK TWICE
ENDM: CLR @SRO ; DISABLE MEM MGMT
BIC @BIT4, @MMR3 ; CLEAR 22 BIT MODE
*****
: AT THIS TIME A 'SUB-PASS' HAS BEEN COMPLETED.
: PROGRAM NOW EXECUTING IN KERNEL MODE AT PC AS SHOWN (NO RELOCATION)
MOV #600, @NEXPAR ; RESET NEXT VALUE FOR PAR REGISTERS
TST @PROT
BEQ 2$
MOV #1600, @NEXPAR
2$: CLRB @MMON ; SET MEM MGMT ON IND = OFF
*****
: SBTTL END OF SUB-PASS ROUTINE
: * THIS ROUTINE SETSUP THE PSW AND MAINTENANCE REGISTERS
: * FOR THE NEXT SUB-PASS. IT THEN STARTS THE PRINTER
: * (IF NOT ON ACT11) FOR TYPING THE END OF SUB-PASS MESSAGE.
: *****
END:
END1: MOV @ERPRT, @ERRVEC
CLR @PSW ; CLEAR MODE BITS IN PSW
JSR PC, CLRTBIT ; GO CLEAR 'T' BIT IF SET
MOV @KERSTK, SP ; SET KERNEL STACK PTR
BIT #100, @STPS ; CHECK IF OUTPUT DEVICE IS BUSY
BNE -6 ; IS AVAILABLE
1$: INCB @SUBPASS

```

```

6850 036420 162702 000060 SUB #60,R2
6851 036424 022702 000006 CMP #6,R2
6852 036430 001013 BNE 2$ ;BRANCH ;END OF TEST?
6853 036432 012737 000060 001532 MOV #60,2$SUBPASS ;INIT SUBPASS COUNT TO ASCII 0
6854 036440 005037 177750 CLR 2$MAINT ;CLEAR MAINTENANCE REG
6855 036444 005037 001604 CLR 2$SMAINT ;CLEAR SOFTWARE VALUE
6856 036450 005046 CLR -(SP)
6857 036452 012746 036560 MOV #SEOP,-(SP)
6858 036456 000002 RTI
6859 036460 006302 2$: ASL R2
6860 036462 012737 001472 000014 MOV #RTRN,2$TBITVEC ;SET 'T' TRAP VECTOR
6861 036470 012737 001531 001270 MOV #SUBPASS-1,2$SREGS
6862 036476 106277 142542 ASRB 2$STPS
6863 036502 016246 053536 MOV PSWTAB(2),-(SP) ;PUSH NEXT PASS PSW ON STACK
6864 036506 012746 005372 MOV #LOOP,-(SP) ;RESART PROGRAM AT LOOP
6865 036512 016237 053552 001604 MOV MRGTAB(R2),2$SMAINT
6866 036520 016237 053552 177750 MOV MRGTAB(R2),2$MAINT
6867 036526 105737 001504 3$: TSTB 2$OV ;OV PASS?
6868 036532 001011 BNE RTI1 ;BRANCH IF YES
6869 036534 122777 000200 142502 CMPB #200,2$STPS ;IS PRINTER READY?
6870 036542 001371 BNE 3$ ;BRANCH IF NO
6871 036544 012737 054602 001270 MOV #MSG20-1,2$SREGS
6872 036552 106277 142466 ASRB 2$STPS
6873 036556 000002 RTI1: RTI ;TYPE END SUBPASS MESSAGE
; ;RESTART PROGRAM AT LOOP WITH NEW PSW
; ;(FROM TABLE BELOW)
; ;*****
; .SBTTL END OF PASS ROUTINE
; *INCREMENT THE PASS NUMBER ($PASS)
; *TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYY"
; *WHERE XXXXX AND YYYY ARE DECIMAL NUMBERS
; *IF THERES A MONITOR GO TO IT
; *IF THERE ISN'T JUMP TO LOOP
SEOP:
036560 JSR PC,2$TYPTIME
036564 CLR $STNM ;: ZERO THE TEST NUMBER
036570 CLR $TIMES ;: ZERO THE NUMBER OF ITERATIONS
036574 INC $PASS ;: INCREMENT THE PASS NUMBER
036600 BIC #100000,$PASS ;: DON'T ALLOW A NEG. NUMBER
036606 DEC (PC)+ ;: LOOP?
SEOPCT: .WORD 1
036610 BGT $DOAGN ;: YES
036612 MOV (PC)+,2(PC)+ ;: RESTORE COUNTER
SENDCT: .WORD 1
036620 SEOPCT
036622 TYPE 65$ ;: TYPE ASCII STRING
036626 BR 64$ ;: GET OVER THE ASCII
; 65$: .ASCIIZ <12><15>/END PASS #/
64$: MOV $PASS,-(SP) ;: SAVE $PASS FOR TYPEOUT
;: TYPE PASS NUMBER
;: GO TYPE--DECIMAL ASCII WITH SIGN
TYPDS

```

```

6906 036660 000421          BR      66$          ;;GET OVER THE ASCIZ
6907          ;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
6908 036724          66$:          MOV      SERTTL,-(SP)      ;;SAVE SERTTL FOR TYPEOUT
6909 036724 016746 142264          TYPDS          ;;TOTAL NUMBER OF ERRORS
6910          TYPE          ;;GO TYPE--DECIMAL ASCII WITH SIGN
6911 036730 104410          CLR      $SCLF          ;;TYPE CARRIAGE RETURN, LINE FEED
6912 036732 104400 001327          CLR      SERTTL          ;;CLEAR ERROR TOTAL
6913 036736 005067 142252          SGET42: MOV      2#42,RO      ;;GET MONITOR ADDRESS
6914 036742 013700 000042          BEQ      $DOAGN          ;;BRANCH IF NO MONITOR
6915 036746 001405          RESET          ;;CLEAR THE WORLD
6916 036750 000005          SENDAD: JSR     PC,(RO)      ;;GO TO MONITOR
6917 036752 004710          NOP          ;;SAVE ROOM
6918 036754 000240          NOP          ;;FOR
6919 036756 000240          NOP          ;;ACT11
6920 036760 000240          $DOAGN:
6921 036762          JMP      2#LOOP          ;;RETURN
6922 036762 000137 005372          SNULL: .BYTE  -1,-1,0      ;;NULL CHARACTER STRING
6923 036766          .EVEN
6924          036772
6925          ;;*****
6926          ;SBTTL RP11/RP03 HANDLER
6927          ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
6928          ;*****
6929 036772 104412          RP3DRV: SAVREG
6930 036774 105037 001676          CLRB      2#RP3HSTA          ;CLEAR DONE FLAG
6931 037000 032737 000400 201676          BIT      #BIT8,2#RP3HSTA      ;REPEAT FLAG SET?
6932 037006 001403          BEQ      8$                  ;BRANCH IF NO
6933 037010 104414          RESREG
6934 037012 000137 040644          JMP      2#RP3RPT
6935 037016 013737 001556 001566 8$: MOV      2#RNTBINX,2#RP311      ;SAVE RUN TABLE INDEX
6936 037024 032737 000020 177570          BIT      #SW4,2#SWR          ;INHIBIT RND DSK ADR?
6937 037032 001403          BEQ      1$                  ;BRANCH IF NO
6938 037034 005000          CLR      RO
6939 037036 005001          CLR      R1
6940 037040 000410          BR      4$
6941 037042 004737 050604          1$: JSR     PC,2#$RAND          ;GO GET RANDOM NUMBER
6942 037046 013700 001524          MOV      2#$HINUM,RO          ;GET HI NUMBER
6943 037052 013701 001522          MOV      2#$LONUM,R1          ;GET LO NUMBER
6944 037056 073027 177771          ASHC     #-7,RO              ;ADJUST TO FORM CYL ADR
6945 037062 042700 177000          4$: BIC     #177000,RO          ;GET RID OF UNUSED BITS
6946 037066 022700 000624          CMP      #624,RO              ;LEGAL CYL?
6947 037072 100003          BFL     5$                  ;BRANCH IF YES
6948 037074 062700 000624          ADD      #624,RO              ;MAKE IT LEGAL
6949 037100 000770          BR      4$
6950 037102 013702 001566          5$: MOV      2#RP311,R2          ;GET RUN TABLE INDEX
6951 037106 016203 001626          MOV      RUNTBL(R2),R3        ;GET DEVICE ID
6952 037112 042703 000777          BIC     #777,R3              ;ID ONLY
6953 037116 050300          BIS     R3,RO                ;COMBINE WITH CYL ADR
6954 037120 010062 001626          MOV      RO,RUNTBL(R2)        ;PUT BACK IN TABLE
6955 037124 072127 177775          ASH     #-3,R1                ;GEN TRK-SECT ADR
6956 037130 010103          MOV      R1,R3                ;SAVE
6957 037132 042701 160377          6$: BIC     #160377,R1          ;GET RID OF ALL BUT TRK
6958 037136 022701 011400          CMP      #11400,R1            ;LEGAL TRAK?
6959 037142 100003          BPL     2$                  ;BRANCH IF YES
6960 037144 062701 011400          ADD      #11400,R1            ;MAKE IT LEGAL

```



```

6962 037152 042703 177760      2$:   BIC      #177760,R3      ;GET SECTOR ADR
6963 037156 022703 000011      CMP      #11,R3        ;IS IT LEGAL?
6964 037162 100003                BPL      JS          ;BRANCH IF YES
6965 037164 062703 000011      ADD      #11,R3        ;MAKE IT LEGAL
6966 037170 000770                BR       2$
6967 037172 050301      3$:   BIS      R3,R1        ;COMBINE TRK-SECT
6968 037174 010162 001642      MOV      R1,RUNTRAK(R2);PUT IN TABLE
6969 037200 010037 002044      MOV      R0,@#RP3HDC   ;SAVE DESIRED CYL
6970 037204 010137 002112      MOV      R1,@#RP3DA    ;SAVE DSK ADR
6971 037210 112737 177775 002072  MOVB     #-3,@#RP3TRY   ;INIT TRY COUNT
6972 037216 032737 000040 172516  BIT      #BITS,@#MMR3   ;MAP ON?
6973 037224 001405                BEQ      7$            ;BRANCH IF NO
6974 037226 005046                CLR      -(SP)         ;PUT DEVICE ID ON STACK
6975 037230 013746 001732      MOV      @#RP3CLD,-(SP);PUT ADR OF BUS ADR ON STK
6976 037234 004737 052054      JSR      PC,@#GETMAP   ;GET MAP REGISTER
6977 037240 012737 000103 001564  7$:   MOV      #103,@#RP310  ;GET FUNCTION
6978 037246 013700 001734      MOV      @#RP3OLD+2,R0;GET BAE BITS
6979 037252 072027 000004      ASH      #4,R0         ;SHIFT TO BITS 4 & 5
6980 037256 050037 001564      BIS      R0,@#RP310   ;COMBINE WITH FUNCTION
6981 037262 010037 001734      MOV      R0,@#RP3OLD+2
6982 037266 013700 002012      MOV      @#RP3UNIT,R0
6983 037272 072027 000010      ASH      #10,R0        ;SHIFT UNIT NO TO RIGHT BITS
6984 037276 050037 001564      BIS      R0,@#RP310   ;COMBINE WITH FUNC & BAE
6985 037302 010037 002012      MOV      R0,@#RP3UNIT
6986 037306 104414                RESREG
6987 037310 005777 142564      RP3WTRY:TST @#RP3DS      ;IS DRIVE READY?
6988 037314 100375                BPL      RP3WTRY       ;BRANCH IF NO
6989 037316 053777 002012 142560  BIS      @#RP3UNIT,@#RP3CS;SET UNIT BITS
6990 037324 004737 037356      JSR      PC,@#LDRP3    ;LOAD RP3 REGISTERS
6991 037330 012777 040674 142560  MOV      @#RP3SRV,@#RP3VEC;SET VECTOR
6992 037336 005077 142556      CLR      @#RP3PSW
6993 037342 005037 002060      CLR      @#RP3FUN
6994 037346 013777 001564 142530  MOV      @#RP310,@#RP3CS;SET FUNCTION TO WRITE
6995 037354 000002                RTI
6996 037356 013777 002042 142526  LDRP3: MOV      @#RP3HDA,@#RP3DA ;LOAD DSK ADR
6997 037364 013777 002044 142522  MOV      @#RP3HDC,@#RP3DC ;LOAD CYL ADR
6998 037372 013777 001716 142506  MOV      @#RP3HWC,@#RP3WC ;LOAD WORD COUNT
6999 037400 013777 001732 142502  MOV      @#RP3OLD,@#RP3BA ;LOAD BUS ADR
7000 037406 000207                RTS      PC            ;RETURN
7001
7002
7003 :*****
7004 :SBTTL RK11/RK05 HANDLER
7005 :* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
7006 :*****
7006 037410 104412      RKDRV: SAVREG
7007 037412 105037 001700      CLR      @#ARKHSTAT   ;CLEAR DONE FLAG IN HANDLER STAT
7008 037416 032737 000400 001700  BIT      #BIT8,@#ARKHSTAT;REPEAT FLAG SET?
7009 037424 001403                BEQ      5$            ;BRANCH IF NO
7010 037426 104414                RESREG
7011 037430 000137 041462      JMP      @#ARKRPT
7012 037434 013737 001556 001572  5$:   MOV      @#RANTBINX,@#RK11 ;SAVE RUN TABLE INDEX
7013 037442 105037 001700      CLR      @#ARKHSTAT   ;CLEAR DONE FLAG IN HANDLER STAT
7014 037446 032737 000020 177570  BIT      #SW4,@#SWR    ;RANDOM DSK ADDRESS?
7015 037454 001403                BEQ      6$            ;BRANCH IF YES
7016 037456 005000                CLR      R0            ;CLEAR REGISTERS

```

7018	037462	000404	
7019	037464	004737	050604
7020	037470	013700	001524
7021	037474	072027	177775
7022			
7023	037500	010001	
7024	037502	042701	160037
7025	037506	022701	014300
7026	037512	100003	
7027	037514	062701	014340
7028	037520	000770	
7029	037522	072127	177773

	6\$:	BR	7\$
		JSR	PC,2\$SRAND
		MOV	2\$SHINUM,RO
	7\$:	ASH	#-3,RO
		MOV	RO,R1
	4\$:	BIC	#160037,R1
		CMP	#14300,R1
		BPL	3\$
		ADD	#14340,R1
		BR	4\$
	3\$:	ASH	#-5,R1

```

:FOR ADDRESS CHECKING
:GET RANOOM NUMBER
:GET HIGH NUMBER
:ADJUST TO FORM
:CYLINDER ADDRESS
:SAVE IN R1
:GET RID OF SURF-SECT BITS
:IS IT A LEGAL CYLINDER?
:BRANCH IF YES
:ADD MAXIMUM CYLINDER
:TRY AGAIN
:ADJUST CYLINDER ADDRESS

```

K12

7031 037532 016203 001626  
7032 037536 042703 000777  
7033 037542 050103  
7034 037544 010362 001626  
7035 037550 072027 177770  
7036 037554 042700 177740  
7037 037560 010003

MOV RUNTBL(R2),R3  
BIC #777,R3  
BIS R1,R3  
MOV R3,RUNTBL(R2)  
ASH #-10,R0  
BIC #177740,R0  
MOV R0,R3

:GET RUN TABLE ENTRY  
:SAVE ID AND UNIT NO.  
:INSERT CYLINDER ADDR  
:ENTER CYLINDER ADR IN RUN TABLE  
:GENER SECTOR-SURF ADDRESS  
:GET RID OF EXTRA BITS  
:SAVE

```

7039 037566 022700 000012          CMP      #12,RO          ; IS SECTOR ADDRESS LEGAL?
7040 037572 100004          BPL      1$             ; BRANCH IF YES
7041 037574 062700 000012          ADD      #12,RO        ; MAKE IT LEGAL
7042 037600 042700 000020          BIC      #BIT4,RO      ; GET RID OF CARRY FROM ADD
7043 037604 042703 000017          BIC      #17,R3        ; GET SURFACE ADDRESS
7044 037610 050300          BIS      R3,RO         ; GENER COMP SECT-SURF ADDRESS
7045 037612 010062 001642          MOV      RO,RUNTRAK(R2); SAVE IN RUN TRAK TABLE
7046 037616 072127 000005          ASH     #5,R1          ; ADJUST CYLINDER ADDRESS
7047 037622 050100          BIS      R1,RO         ; CONCATINATE TRK & SECT ADDR
7048 037624 013701 002014          MOV      @#RKUNIT,R1  ; GET UNIT NUMBER
7049 037630 072127 000015          ASH     #15,R1        ; ADJUST
7050 037634 050100          BIS      R1,RO         ; CONCATINATE UNIT,TRK,SURF,SECT
7051 037636 010037 002046          MOV      RO,@#RKHDA   ; SAVE
7052 037642 112737 177775 002073  MOVVB   #-3,@#RKTRY   ; SET RETRY COUNT
7053 037650 032737 000040 172516  BIT     #BIT5,@#MMR3  ; MAP ON?
7054 037656 001406          BEQ     2$             ; BRANCH IF NO
7055 037660 012746 000001          MOV     #1,-(SP)      ; PUT DEVICE ID ON STACK
7056 037664 012746 001736          MOV     @#KOLD,-(SP)  ; PUT ADDRESS OF ADR ON STACK
7057 037670 004737 052054          JSR     PC,@#GETMAP   ; GET MAP REG
7058 037674 012767 000103 141666 25:    MOV     #103,RK10    ; SET FUNCTION
7059 037702 013700 001740          MOV     @#KOLD+2,RO   ; GET BA EXTENDED
7060 037706 072027 000004          ASH     #4,RO         ; ADJUST
7061 037712 050037 001570          BIS     RO,@#RK10    ; PUT IN WITH FUNCTION
7062 037716 010037 001740          MOV     RO,@#KOLD+2  ; SAVE IN MEMORY
7063 037722 104414          RESREG
7064 037724 013777 002046 142202  RKWTRY: MOV   @#RKHDA,@#RKDA ; LOAD DISK ADDRESS
7065 037732 032777 000100 142162  BIT     #BIT6,@#RKDS ; UNIT READY?
7066 037740 001774          BEQ     -6            ; BRANCH IF NO
7067 037742 013777 001720 142160  MOV     @#RKHWC,@#RKWC ; LOAD WORD COUNT
7068 037750 013777 001736 142154  MOV     @#KOLD,@#KBA   ; LOAD BUS ADDRESS
7069 037756 012777 041512 142152  MOV     @#RKSrv,@#RKVEC ; LOAD INTERRUPT VECTOR
7070 037764 005077 142150          CLR     @#RKPSW
7071 037770 005037 002062          CLR     @#RKFUN
7072 037774 013777 001570 142124  MOV     @#RK10,@#RKCS ; SET FUNCTION TO WRITE
7073 040002 000006          RTT
7074
7075
7076
7077
7078
7079 040004 104412          ;*****
7080 040006 105037 001706          ;SBTTL RH70/RP04 HANDLER
7081 040012 032737 000400 001706  ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
7082 040020 001403          ;*****
7083 040022 104414          RP4DRV: SAVREG
7084 040024 000137 042340          CLR     @#RP4HSTA    ; CLEAR DONE FLAG
7085 040030 013737 001556 001574 65:    BIT     #BIT8,@#RP4HST ; REPEAT FLAG SET?
7086 040036 105037 001706          BEQ     6$           ; BRANCH IF NO
7087 040042 032737 000020 177570  RESREG
7088 040050 001403          JMP     @#RP4RPT
7089 040052 005000          MOV     @#RNTBINX,@#RP411 ; SAVE RUN TABLE INDEX
7090 040054 005001          CLR     @#RP4HSTA    ; CLEAR DONE FLAG
7091 040056 000410          BIT     #SW4,@#SWR   ; RANDOM DSK ADDRESS?
7092 040060 004737 050604          BEQ     1$           ; BRANCH IF YES
7093 040064 013700 001524          CLR     RO
7093 040064 013700 001524          CLR     R1
7093 040064 013700 001524          BR      4$
7093 040064 013700 001524          JSR     PC,@#SRAND   ; GET RANDOM NUMBER
7093 040064 013700 001524          MOV     @#SHINUM,RO  ; GET HI NUMBER

```

```

7095 040074 073027 177771          ASHC      #-7,R0          ;ADJUST TO FORM CYL. ADR.
7096 040100 042700 177000          BIC      #177000,R0 ;GET RID OF UNUSED BITS
7097 040104 022700 000631          CMP      #631,R0   ;LEGAL CYLINDER
7098 040110 100003          BPL      $        ;BRANCH IF YES
7099 040112 062700 000631          ADD      #631,R0   ;MAKE IT LEGAL
7100 040116 000770          BR      4$
7101
7102 040120 013702 001574          5$:      MOV      @#RP411,R2 ;GET RUN TABLE INDEX
7103 040124 016203 001626          MOV      RUNTBL(R2),R3 ;GET DEVICE ID
7104 040130 042703 000777          BIC      #777,R3   ;SAVE ID ONLY
7105 040134 050003          BIS      R0,R3    ;COMBINE WITH CYL ADR
7106 040136 010362 001626          MOV      R3,RUNTBL(R2) ;PUT IN RUN TABLE
7107 040142 072127 177775          ASH      #-3,R1   ;GEN TRAK-SECT ADR
7108 040146 042701 160340          BIC      #160340,R1 ;GET RID OF UNUSED BITS
7109 040152 010103          MOV      R1,R3    ;SAVE
7110 040154 042701 000037          BIC      #37,R1   ;GET RID OF SECT BITS
7111 040160 022701 011000          CMP      #11000,R1 ;LEGAL TRAK?
7112 040164 100004          BPL      2$      ;BRANCH IF YES
7113 040166 062701 011000          ADD      #11000,R1 ;MAKE IT LEGAL
7114 040172 042701 020000          BIC      #BIT13,R1 ;GET RID OF ADD CARRY
7115 040176 042703 177740          2$:      BIC      #177740,R3 ;GET SECTOR ADR
7116 040202 022703 000025          CMP      #25,R3   ;LEGAL SECTOR
7117 040206 100004          BPL      3$      ;BRANCH IF YES
7118 040210 062703 000025          ADD      #25,R3   ;MAKE IT LEGAL
7119 040214 042703 000040          BIC      #BIT5,R3 ;GET RID OF ADD CARRY
7120 040220 050301          3$:      BIS      R3,R1   ;COMBINE TRAK-SECTOR
7121 040222 010162 001642          MOV      R1,RUNTRAK(R2) ;PUT TRAK-SECT IN TABLE
7122 040226 010037 002054          MOV      R0,@#RP4HDC ;SAVE CYLINDER ADR
7123 040232 010137 002052          MOV      R1,@#RP4HDA ;SAVE TRAK-SECTOR ADR
7124 040236 112737 177775          002075    MOVB     #-3,@#RP4TRY ;SET TRY COUNT
7125 040244 104414          RESREG
7126 040246 004767 000026          RP4WTRY: JSR      PC,LDRP4   ;LOAD RP4 REGISTERS
7127 040252 012777 042364          141716    MOV      #RP4SRV,@RP4VEC ;LOAD INTERRUPT VECTOR
7128 040260 005077 141714          CLR      @RP4PSW
7129 040264 005037 002066          CLR      @#RP4FSW
7130 040270 112777 000161          141644    MOVB     #161,@RP4CS1 ;SET FUNCTION TO WRITE
7131 040276 000002          RTI      ;LOAD FUNCTION AND GO
7132
7133 040300 013777 002022          141646    LDRP4:  MOV      @#RP4UNIT,@RP4CS2 ;LOAD UNIT NUMBER
7134 040306 012777 010000          MOV      #BIT12,@RP4OF ;SET FORMAT TO 16 BIT
7135 040314 013777 002054          MOV      @#RP4HDC,@RP4DC ;LOAD CYLINDER ADR
7136 040322 013777 002052          MOV      @#RP4HDA,@RP4DA ;LOAD TRAK-SECTOR
7137 040330 013777 001726          MOV      @#RP4HWC,@RP4WC ;LOAD WORD COUNT
7138 040336 013777 001754          MOV      @#RP4OLD+2,@RP4BAE ;LOAD EXTENDED ADR BITS
7139 040344 013777 001752          MOV      @#RP4OLD,@RP4BA ;LOAD BUS ADR
7140 040352 000207          RTS      PC       ;RETURN
7141
7142
7143
7144
7145
7146 040354 104412          RSDRV:  SAVREG
7147 040356 105037 001710          CLRB     @#RSHSTAT ;CLEAR DONE FLAG
7148 040362 032737 000400          BIT      #BIT8,@#RSHSTAT ;REPEAT FLAG SET?
7149 040370 001403          BEQ     3$      ;BRANCH IF NO

```

```

7151 040374 000137 043050          JMP      @RSRPT
7152 040400 013737 001556 001576 3S:  MOV      @RANTBINX,@RS11      ;SAVE RUN TABLE INDEX
7153 040406 032737 000020 177570  BIT      @SW4,@SWR           ;RANDOM DSK ADR?
7154 040414 001403          BEQ      1$                 ;BRANCH IF YES
7155 040416 005000          CLR      R0
7156 040420 005001          CLR      R1
7157 040422 000407          BR      4$
7158 040424 004737 050604          JSR      PC,@SRAND           ;GET RANDOM NUMBER
7159 040430 013700 001524          MOV      @SHINUM,R0
7160 040434 072027 177774          ASH      #-4,R0
7161 040440 010001          MOV      R0,R1              ;SAVE RANDOM NUMBER
7162 040442 042700 170077          BIC      @170077,R0         ;GET TRACK ADR
7163 040446 022700 007600          CMP      @7600,R0          ;IS IT LEGAL?
7164 040452 100003          BPL      5$                 ;BRANCH IF YES
7165 040454 062700 007600          ADD      @7600,R0          ;MAKE IT LEGAL
7166 040460 000770          BR      4$
7167 040462 013702 001576          MOV      @RS11,R2          ;GET RUN TABLE INDEX
7168 040466 072027 177772          ASH      #-6,R0           ;ADJUST TRACK ADR
7169 040472 110062 001626          MOV      R0,RUNTABL(R2)    ;SAVE TRAK ADR IN RUN TBL
7170 040476 042701 177700          BIC      @177700,R1        ;GET SECTOR ADR
7171 040502 022701 000077          CMP      @77,R1           ;IS IT LEGAL?
7172 040506 100003          BPL      2$                 ;BRANCH IF YES
7173 040510 062701 000077          ADD      @77,R1           ;MAKE IT LEGAL
7174 040514 000770          BR      6$
7175 040516 010162 001642          MOV      R1,RUNTRAK(R2)    ;SAVE IN RUN TRAK TABLE
7176 040522 072027 000006          ASH      @6,R0            ;ADJUST TRACK ADDR
7177 040526 050100          BIS      R1,R0            ;COMBINE SECTOR TRAK
7178 040530 010037 002056          MOV      R0,@RSHDA        ;SAVE AS DSK ADR
7179 040534 112737 177775 002076  MOV      @-3,@RSTRY        ;SET TRY COUNT
7180 040542 104414          RESREG
7181 040544 004737 040604          JSR      PC,@LDRS          ;GO LOAD REGISTERS
7182 040550 012777 043074 141446  MOV      @RSSRV,@RSVEC     ;SET INTERRUPT VECTOR
7183 040556 005077 141444          CLR      @RSPSW
7184 040562 005037 002070          CLR      @RSFUN           ;SET FUNCTION TO WRITE
7185 040566 105777 141426          TSTB    @RSDS             ;IS DRIVE READY?
7186 040572 001775          BEQ      1$                 ;BRANCH IF NO
7187 040574 112777 000161 141400  MOV      @161,@RSCS1       ;LOAD FUNCTION AND GO
7188 040602 000002          RTI
7189
7190 040604 013777 002024 141402  LDRS:  MOV      @RSUNIT,@RSCS2   ;LOAD UNIT NUMBER
7191 040612 013777 002056 141372  MOV      @RSHDA,@RSDA      ;LOAD DSK ADR
7192 040620 013777 001730 141356  MOV      @RSHWC,@RSHWC     ;LOAD WORD COUNT
7193 040626 013777 001760 141354  MOV      @RSOLD+2,@RSBAE    ;LOAD EXTENDED ADDRESS
7194 040634 013777 001756 141344  MOV      @RSOLD,@RSBA      ;LOAD BUS ADDRESS
7195 040642 000207          RTS      PC                ;RETURN
7196
7197  ;*****
7198  ;SBTTL RP11/RP03 SERVICE ROUTINE
7199  ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
7200  ;*****
7201  RP3RPT: RESET
7202 040644 000005          DEC      @003FUN          ;RESTORE FUNCTION
7203 040646 005337 002060          CMP      @1,@RP3FUN        ;WHAT IS IT?
7204 040652 022737 000001 002060  BEQ      RP31              ;BRANCH IF WC
7205 040660 001472          BLT     1$                 ;BRANCH IF WRITE

```

7207	040670	000167	000414		1\$: JMP	RP33	
7208	040674	005237	002060		RP3SRV: INC	@RP3FUN	: INCREMENT FUNCTION
7209	040700	022737	000002	002060	CMP	#2, @RP3FUN	: WHAT IS IT?
7210	040706	001501			BEQ	RP3WCK	: BRANCH TO WRITE CHECK
7211	040710	100002			BPL	+6	
7212	040712	000137	041350		JMP	@RP3READ	
7213							
7214							
7215	040716	032737	000400	001676			: FUNCTION JUST EXECUTED WAS A WRITE
7216	040724	001036			BIT	#BIT8, @RP3HSTAT	: REPEAT FLAG SET?
7217	040726	005777	141152		BNE	RP3LOOP	: BRANCH IF YES
7218	040732	100045			TST	@RP3CS	: ANY ERRORS?
7219	040734	105737	002072		BPL	RP31	: BRANCH IF NO
7220	040740	001415			TSTB	@RP3TRY	: TRIED 3 TIMES?
7221	040742	112777	000001	141134	BEQ	RP3ERR	: BRANCH IF YES
7222	040750	105777	141130		MOVB	#BIT0, @RP3CS	: CLEAR THE DRIVE
7223	040754	100375			TSTB	@RP3CS	: CONTROLLER READY?
7224	040756	105237	002072		BPL	-4	: BRANCH IF NO
7225	040762	013746	177776		INCB	@RP3TRY	: INCREMENT TRY COUNT
7226	040766	012746	037310		MOV	@PSW, -(SP)	: MAINTAIN SAME PSW
7227	040772	000002			MOV	@RP3TRY, -(SP)	: SET RETRY ADDRESS
7228	040774	012737	100200	001676	RTI		: RETURN
7229	041002	010046			RP3ERR: MOV	#100200, @RP3HSTA	: SET ERROR BIT IN HAND. STA
7230	041004	013700	001566		MOV	RO, -(SP)	: SAVE RO
7231	041010	052760	100000	001642	MOV	@RP311, RO	: GET RUNTABLE INDEX
7232	041016	012600			BIS	#BIT15, @UNTRAK(RO)	: SET ERROR BIT
7233	041020	000002			MOV	(SP)+, RO	: RESTORE RO
7234					RTI		: RETURN
7235	041022	012737	100200	001676	RP3LOOP: MOV	#100200, @RP3HSTAT	: SET DONE AND ERROR
7236	041030	005777	141050		TST	@RP3CS	: ANY ERRORS?
7237	041034	100403			BMI	1\$	: BRANCH IF YES
7238	041036	042737	100000	001676	BIC	#BIT15, @RP3STAT	: CLEAR ERROR BIT
7239	041044	000002			1\$: RTI		: RETURN
7240							: WRITE WAS OK- NOW DO A WRITE CHECK
7241	041046	112737	177775	002072	RP31: MOVB	#-3, @RP3TRY	: INIT TRY COUNT
7242	041054	012737	000107	001564	MOV	#107, @RP310	: SET FUNCTION
7243	041062	053737	001734	001564	BIS	@RP3CLO+2, @RP310	: SET BAE BITS
7244	041070	053737	002012	001564	BIS	@RP3UNIT, @RP310	: SET UNIT BITS
7245	041076	004737	037356		RP32: JSR	PC, @LDRP3	: LOAD RP3 REGISTERS
7246	041102	013777	001564	140774	MOV	@RP310, @RP3CS	: LOAD FUNCTION AND GO
7247	041110	000002			RTI		: RETURN
7248							
7249							
7250	041112	032737	000400	001676			: FUNCTION JUST EXECUTED WAS A WRITE CHECK
7251	041120	001340			RP3WCK: BIT	#BIT8, @RP3HSTAT	: REPEAT FLAG SET?
7252	041122	005777	140756		BNE	RP3LOOP	: BRANCH IF YES
7253	041126	100031			TST	@RP3CS	: ANY ERRORS?
7254	041130	005737	001566		BPL	1\$	: BRANCH IF NO
7255	041134	001422			TST	@RP311	: FIRST 2K?
7256	041136	105737	002072		BEQ	4\$	: BRANCH IF YES
7257	041142	001714			SS: TSTB	@RP3TRY	: TRIED 3 TIMES?
7258	041144	005337	002060		BEQ	RP3ERR	: BRANCH IF YES
7259	041150	112777	000001	140726	DEC	@RP3FUN	: RESTORE FUNCTION
7260	041156	105777	140722		MOVB	#BIT0, @RP3CS	: CLEAR THE DRIVE
7261	041162	100375			TSTB	@RP3CS	: CONTROLLER READY?
					BPL	-4	: BRANCH IF NO

```

7263 041170 013746 177776          MOV      @RPSW, -(SP)
7264 041174 012746 041076          MOV      @RP32, -(SP)
7265 041200 000002                    RTI
7266 041202 032777 000010 140672 4S:  BIT      @BIT3, @RP3ER
7267 041210 001752                    BEQ      SS
7268
7269
7270 041212 112737 177775 002072 1S:  WRITE CHECK OK- NOW DO A READ
7271 041220 032737 000040 172516  MOVB    #-3, @RP3TRY
7272 041226 001407                    BIT      @BIT5, @MMR3
7273 041230 005046                    BEQ      2S
7274 041232 004737 052320          CLR      -(SP)
7275 041236 012746 001762          JSR     PC, @GIVEMAP
7276 041242 004737 052054          MOV     @RP3NL, -(SP)
7277 041246 010046                    JSR     PC, @GETMAP
7278 041250 013700 001764 2S:  MOV     RO, -(SP)
7279 041254 072027 000004          MOV     @RP3NH, RO
7280 041260 010037 001764          ASH    #4, RO
7281 041264 012600                    MOV     RO, @RP3NH
7282 041266 012737 000105 001564  MOV     (SP)+, RO
7283 041274 053737 001764 001564  MOV     @105, @RP310
7284 041302 053737 002012 001564  BIS     @RP3NH, @RP310
7285 041310 013777 002042 140574  RP33:  BIS     @RP3UNIT, @RP310
7286 041316 013777 002044 140570  MOV     @RP3DA, @RP30A
7287 041324 013777 001716 140554  MOV     @RP3DOC, @RP30C
7288 041332 013777 001762 140550  MOV     @RP3MC, @RP30C
7289 041340 013777 001564 140536  MOV     @RP3NL, @RP3BA
7290 041346 000002                    MOV     @RP310, @RP3CS
7291
7292
7293 041350 032737 000400 001676  .FUNCTION JUST EXECUTED WAS A READ
7294 041356 001221 140520  RP3READ: BIT @BIT8, @RP3HSTAT
7295 041360 005777                    BNE     RP3LOOP
7296 041364 100022                    TST    @RP3CS
7297 041366 105737 002072          BPL     1S
7298 041372 001600                    TSTB   @RP3TRY
7299 041374 005337 002060          BEQ     RP3ERR
7300 041400 112777 000001 140476  DEC     @RP3FUN
7301 041406 105777 140472          MOVB   @BIT0, @RP3CS
7302 041412 100375                    TSTB   @RP3CS
7303 041414 105237 002072          BPL     -4
7304 041420 013746 177776          INCB   @RP3TRY
7305 041424 012746 041310          MOV     @RPSW, -(SP)
7306 041430 000002                    MOV     @RP33, -(SP)
7307 041432 032737 000040 172516 1S:  RTI
7308 041440 001404                    BIT     @BIT5, @MMR3
7309 041442 005046                    BEQ     2S
7310 041444 004737 052320          CLR     -(SP)
7311 041450 005726                    JSR     PC, @GIVEMAP
7312 041452 112737 000200 001676 2S:  TST    (SP)+
7313 041460 000002                    MOVB   @200, @RP3HSTA
7314
7315
7316
7317

```

```

*****
:SBTTL  RK11/RK05 SERVICE ROUTINE
:*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
*****

```



```

7319 021464 005337 002062          DEC      @RKFUN          ;RESTORE FUNCTION
7320 021470 022737 000001 002062  CMP      @1,@RKFUN      ;WHAT IS IT?
7321 021476 001475          BEQ      RK1            ;BRANCH IF WC
7322 021500 002402          BLT      IS            ;BRANCH IF WRITE
7323 021502 000137 037724          JMP      @RKWTRY       ;IT WAS A WRITE
7324 021506 000137 042144          JMP      @RK3
7325 021512 062737 000001 002062  RKSRV:  ADD      @1,@RKFUN      ;FIND OUT WHAT FUNCTION
7326 021520 022737 000002 002062  CMP      @2,@RKFUN      ;WAS EXECUTED
7327 021526 001507          BEQ      RKWRCK        ;WAS IT A WRITE CHECK?
7328 021530 100002          BPL      +6            ;BRANCH IF YES
7329 021532 000137 042176          JMP      @RKREAD       ;BRANCH IF IT WAS A WRITE
7330
7331
7332          ;FUNCTION JUST EXECUTED WAS A WRITE. ANY ERRORS?
7333 021536 032737 000400 001700  BIT      @BIT@,@RKHSTAT ;REPEAT FLAG SET?
7334 021544 001040          BNE      RKLOOP        ;BRANCH IF YES
7335 021546 005777 140354          TST      @RKCS         ;ANY ERRORS?
7336 021552 100047          BPL      RK1            ;BRANCH IF NO
7337 021554 105737 002073          TSTB    @RKTRY         ;TRYED 3 TIMES?
7338 021560 001417          BEQ      RKERR         ;BRANCH IF YES
7339 0215 012777 000001 140336  MOV      @1,@RKCS      ;CLEAR THE ERROR
7340 021570 004737 042322          JSR      PC,@TIMER     ;WAIT A LITTLE
7341 021574 105777 140326          TSTB    @RKCS         ;WAIT FOR CONT CLR TO FINISH
7342 021600 100375          BPL      -4
7343 021602 105237 002073          INCB    @RKTRY         ;INCREMENT TRY COUNT
7344 021606 013746 177776          MOV      @PSW,-(SP)
7345 021612 012746 037724          MOV      @RKWTRY,-(SP)
7346 021616 000002          RTI
7347 021620 012737 100200 001700  RKERR:  MOV      @100200,@RKHSTAT ;SET ERROR & DONE FLAG
7348 021626 010046          MOV      RO,-(SP)      ;SAVE RO
7349 021630 013700 001572          MOV      @RK11,RO     ;GET SAVED RUN TABLE INDEX
7350 021634 052760 100000 001642  BIS      @BIT15,RUNTRAK(RO) ;SET ERROR BIT IN RUN TABLE
7351 021642 012600          MOV      (SP)+,RO     ;RESTORE RO
7352 021644 000002          RTI                    ;RETURN
7353
7354 021646 012737 100200 001700  RKLOOP:MOV @100200,@RKHSTAT ;SET DONE AND ERROR BITS
7355 021654 005777 140246          TST      @RKCS        ;ANY ERRORS?
7356 021660 100403          BMI      IS            ;BRANCH IF YES
7357 021662 042737 100000 001700  BIC      @BIT15,@RKHSTAT ;CLEAR ERROR BIT
7358 021670 000002          RTI                    ;RETURN
7359
7360          ;WRITE WAS OK, NOW DO A WRITE CHECK
7361 021700 112737 177775 002073  RK1:   MOVB    @-3,@RKTRY    ;RESTORE TRY COUNT
7362 021706 053767 001740 137662  MOV      @507,RK10     ;SET FUNCTION TO WRITE
7363 021714 013777 002046 140212  BIS      @RKOLD+2,RK10 ;SET BA EXT BITS
7364 021722 013777 001720 140200  RK2:   MOV      @RKHDA,@RKDA  ;LOAD DISK ADDRESS
7365 021730 013777 001736 140174  MOV      @RKHWC,@RKWC   ;LOAD WORD COUNT
7366 021736 016777 137626 140162  MOV      @RKOLD,@RKBA  ;LOAD BUS ADDRESS
7367 021744 000002          RTI                    ;START FUNCTION
7368
7369          ;FUNCTION JUST EXECUTED WAS A WRITE CHECK. ANY ERRORS?
7370 041746 032737 000400 001700  RKWRCK:BIT @BIT@,@RKHSTAT ;REPEAT FLAG SET?
7371 041754 001334          BNE      RKLOOP        ;BRANCH IF YES
7372 041756 005777 140144          TST      @RKCS        ;ANY ERRORS?
7373 041762 100033          BPL      IS            ;BRANCH IF NO

```

7375	041770	001424				BEQ	45			: BRANCH IF YES
7376	041772	105737	002073		55:	TSTB	2#RKTRY			: TRYED 3 TIMES?
7377	041776	001710				BEQ	RKERR			: BRANCH IF YES
7378	042000	005337	002062			DEC	2#RKFUN			: SET FUNCTION BACK TO WC
7379	042004	012777	000001	140114		MOV	#1, 2#RKCS			: CLEAR THE ERROR
7380	042012	004737	042322			JSR	PC, 2#TIMER			: WAIT A LITTLE
7381	042016	105777	140104			TSTB	2#RKCS			: WAIT FOR CLR TO FINISH
7382	042022	100375				BPL	-4			
7383	042024	105237	002073			INCB	2#RKTRY			: INCREMENT TRY COUNT
7384	042030	013746	177776			MOV	2#PSW, -(SP)			
7385	042034	012746	041714			MOV	2#RK2, -(SP)			
7386	042040	000002				RTI				
7387	042042	032777	040000	140056	45:	BIT	2#BIT14, 2#RKCS			: HARD ERROR?
7388	042050	001350				BNE	55			: BRANCH IF YES
7389										
7390										
7391	042052	112737	177775	002073		WRITE CHECK WAS OK, NOW DO A READ.				
7392	042060	032737	000040	172516	15:	MOV	2#-3, 2#RKTRY			: RESTORE TRY COUNT
7393	042066	001410				BIT	2#BITS, 2#MMR3			: MAP ON?
7394	042070	012746	000001			BEQ	25			: BRANCH IF NO
7395	042074	004767	010220			MOV	#1, -(SP)			: PUT DEVICE ID ON STACK
7396	042100	012746	001766			JSR	PC, GIVEMAP			: RELINQUISH MAP REG
7397	042104	004737	052054			MOV	2#RKNEWL, -(SP)			: PUT ADDR OF BADR ON STACK
7398	042110	010046			25:	JSR	PC, 2#GETMAP			: GET MAPREGISTER
7399	042112	013700	001770			MOV	RO, -(SP)			: SAVE RO
7400	042116	072027	000004			MOV	2#RKNEWH, RO			: GET BA EXT
7401	042122	010037	001770			ASH	#4, RO			: ADJUST
7402	042126	012600				MOV	RO, 2#RKNEWH			: SAVE
7403	042130	012767	000105	137432		MOV	(SP)+, RO			: RESTORE RO
7404	042136	053767	001770	137424		MOV	#105, RK10			: SET FUNCTION
7405	042144	013777	002046	137762	RK3:	BIS	2#RKNEWH, RK10			: SET BA EXT BITS IN FUNCTION
7406	042152	013777	001720	137750		MOV	2#RKHDA, 2#RKDA			: LOAD DISK ADDRESS
7407	042160	013777	001766	137744		MOV	2#RKHWC, 2#RKWC			: LOAD WORD COUNT
7408	042166	016777	137376	137732		MOV	2#RKNEWL, 2#RKBA			: LOAD BUS ADDRESS
7409	042174	000002				MOV	RK10, 2#RKCS			: LOAD FUNCTION AND GO
7410						RTI				: RETURN
7411										
7412	042176	032737	000400	001700		: FUNCTION JUST EXECUTED WAS A READ. ANY ERRORS?				
7413	042204	001220				AKREAD: BIT	2#BITS, 2#RKHSTAT			: REPEAT FLAG SET?
7414	042206	005777	137714			BNE	RKLOOP			: BRANCH IF YES
7415	042212	100026				TST	2#RKCS			: ANY ERRORS?
7416	042214	105737	002073			BPL	15			: BRANCH IF NO
7417	042220	001002				TSTB	2#RK TRY			: TRYED 3 TIMES?
7418	042222	000167	177372			BNE	35			: BRANCH IF NO
7419	042226	005337	002062			JMP	RKERR			
7420	042232	012777	000001	137666	35:	DEC	2#RKFUN			: SET FUNCTION BACK TO READ
7421	042240	004737	042322			MOV	#1, 2#RKCS			: CLEAR THE ERROR
7422	042244	105777	137656			JSR	PC, 2#TIMER			: WAIT A LITTLE
7423	042250	100375				TSTB	2#RKCS			: WAIT FOR CLR TO FINISH
7424	042252	105237	002073			BPL	-4			
7425	042256	013746	177776			INCB	2#RKTRY			: INCREMENT TRY COUNT
7426	042262	012746	042144			MOV	2#PSW, -(SP)			
7427	042266	000002				MOV	2#RK3, -(SP)			
7428	042270	032737	000040	172516	15:	RTI				
7429	042276	001405				BIT	2#BITS, 2#MMR3			: MAP ON?
						BEQ	25			: BRANCH IF NO

```

7431 042304 004737 052320          JSR      PC, @GIVEMAP      ;RELINQUISH MAP REGISTER
7432 042310 005726                   TST      (SP)+             ;POP THE STACK
7433 042312 112737 000200 001700 2$:  MOVB     @200, @ARKHSTAT  ;SET DONE FLAG
7434 042320 000002                   RTI                          ;RETURN
7435 042322 005067 000010          TIMER: CLR      1$
7436 042326 105267 000004          2$:     INCB     1$
7437 042332 001375                   BNE     2$
7438 042334 000207                   RTS     PC
7439 042336 000000          1$:     .WORD
7440
7441 ;*****
7442 ;SBTTL  RH70/RP04 SERVICE ROUTINE
7443 ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
7444 ;*****
7445 042340 000005          RP4RPT: RESET
7446 042342 005337 002066          DEC     @RP4FUN          ;RESTORE FUNCTION
7447 042346 022737 000001 002066  CMP     #1, @RP4FUN      ;WHAT IS IT?
7448 042354 001501                   BEQ     RP41             ;BRANCH IF WC
7449 042356 002560                   BLT     RP43             ;BRANCH IF READ
7450 042360 000137 040246          JMP     @RP4WTRY         ;GO TO WRITE
7451 042364 005237 002066          RP4SRV: INC    @RP4FUN   ;FIND OUT WHAT FUNCTION
7452 042370 022737 000002 002066  CMP     #2, @RP4FUN      ;WAS JUST EXECUTED
7453 042376 001504                   BEQ     RP4WCK
7454 042400 100566                   BMI     RP4READ
7455
7456 ;WRITE FUNCTION WAS JUST EXECUTED.
7457 042402 032737 000400 001706  BIT     @BIT8, @RP4HSTAT ;REPEAT FLAG SET?
7458 042410 001050                   BNE     RP4LOOP         ;BRANCH IF YES
7459 042412 032777 040000 137540  BIT     @BIT14, @RP4DS   ;ANY ERRORS
7460 042420 001457                   BEQ     RP41             ;BRANCH IF NO
7461 042422 105737 002075          TSTB   @RP4TRY          ;TRIED 3 TIMES?
7462 042426 001426                   BEQ     RP4ERR          ;BRANCH IF YES
7463 042430 052777 000040 137516  BIS    @BIT5, @RP4CS2    ;CLEAR ALL ERRORS
7464 042436 004737 040300          PC, @DRP4              ;RELOAD THE UNIT NO
7465 042442 105237 002075          @RP4TRY                ;INCREMENT TRY COUNT
7466 042446 013746 177776          @PSW, -(SP)            ;SETUP THE STACK TO
7467 042452 012746 040246          @RP4WTRY, -(SP)        ;TRY WRITE AGAIN
7468 042456 032737 000400 001706  BIT     @BIT8, @RP4HSTAT ;REPEAT FLAG SET?
7469 042464 001006                   BNE     2$              ;BRANCH IF YES
7470 042466 012777 000007 137446  MOV     #7, @RP4CS1     ;RECALIBRATE
7471 042474 105777 137460          1$:    TSTB   @RP4DS     ;DRIVE READY?
7472 042500 100375                   BPL     1$              ;BRANCH IF NO
7473 042502 000002          2$:    RTI
7474 042504 012737 100200 001706  RP4ERR: MOV     #100200, @RP4HSTA ;SET ERROR & DONE BIT
7475 042512 010046                   MOV     RO, -(SP)       ;SAVE RO
7476 042514 013700 001574          MOV     @RP4I1, RO     ;GET RUN TABLE INDEX
7477 042520 052760 100000 001642  BIS    @BIT15, @UNTRAK(RO) ;SET ERROR BIT
7478 042526 012600                   MOV     (SP)+, RO      ;RESTORE RO
7479 042530 000002                   RTI                      ;RETURN
7480
7481 042532 012737 100200 001706  RP4LOOP: MOV    #100200, @RP4HSTAT ;SET DONE AND ERROR BITS
7482 042540 032777 040000 137412  BIT    @BIT14, @RP4DS   ;ANY ERRORS?
7483 042546 001003                   BNE     1$              ;BRANCH IF YES
7484 042550 042737 100000 001706  BIC    @BIT15, @RP4HSTAT ;CLEAR ERROR BIT
7485 042556 000002          1$:    RTI                      ;RETURN

```

```

7487 042560 112737 177775 002075 RP41:  MOVB  #3,2#RP4TRY      ; INITIALIZE TRY COUNT
7488 042566 105777 137366          RP42:  TSTB  #RP4DS      ; IS DRIVE READY?
7489 042572 001775                    BEQ   RP42      ; BRANCH IF NO
7490 042574 004737 040300          JSR   PC,2#LDRP4
7491 042600 112777 000151 137334  MOVB  #151,2#RP4CS1  ; LOAD FUNCTION AND GO
7492 042606 000002
7493
7494
7495 042610 032737 000400 001706  RP4WCK: BIT  #BIT8,2#RP4HSTAT  ; REPEAT FLAG SET?
7496 042616 001345                    BNE  RP4LOOP    ; BRANCH IF YES
7497 042620 032777 040000 137332  BIT  #BIT14,2#RP4DS  ; ANY ERRORS?
7498 042626 001421                    BEQ   1$        ; BRANCH IF NO
7499 042630 105737 002075          3$:  TSTB  2#RP4TRY    ; TRIED 3 TIMES?
7500 042634 001723                    BEQ   RP4ERR    ; BRANCH IF YES
7501 042636 005337 002062          DEC  2#RP4FUN    ; SET FUNCTION TO WC
7502 042642 052777 000040 137304  BIS  #BITS,2#RP4CS2 ; CLEAR ALL ERRORS
7503 042650 004737 040300          JSR   PC,2#LDRP4  ; RELOAD THE UNIT NO
7504 042654 105237 002075          INCB 2#RP4TRY    ; INCREMENT TRY COUNT
7505 042660 013746 177776          MOV  2#PSW,-(SP)
7506 042664 012746 042566          MOV  #RP42,-(SP)
7507 042670 000002                    RTI
7508 042672 032777 040000 137254  1$:  BIT  #BIT14,2#RP4CS2  ; TRY AGAIN
7509 042700 001404                    BEQ   2$        ; WRITE CHECK ERROR?
7510 042702 005737 001574          TST  2#RP411    ; BRANCH IF NO
7511 042706 001401                    BEQ   2$        ; FIRST 2K?
7512 042710 000747                    BR   3$
7513
7514
7515 042712 112737 177775 002075  2$:  MOVB  #3,2#RP4TRY    ; INITIALIZE TRY COUNT
7516 042720 105777 137234          RP43:  TSTB  #RP4DS      ; IS DRIVE READY?
7517 042724 001775                    BEQ   RP43      ; BRANCH IF NO
7518 042726 004737 040300          JSR   PC,2#LDRP4  ; LOAD REGISTERS
7519 042732 013777 002004 137210  MOV  2#RP4NWL,2#RP4BAE ; LOAD EXTENDED ADR BITS
7520 042740 013777 002002 137200  MOV  2#RP4NWL,2#RP4BA ; LOAD BUS ADR
7521 042746 112777 000171 137166  MOVB  #171,2#RP4CS1 ; LOAD FUNCTION AND GO
7522 042754 000002                    RTI              ; RETURN
7523
7524
7525 042756 032737 000400 001706  RP4READ: BIT  #BIT8,2#RP4HSTAT  ; REPEAT FLAG SET?
7526 042764 001262                    BNE  RP4LOOP    ; BRANCH IF YES
7527 042766 032777 040000 137164  BIT  #BIT14,2#RP4DS  ; ANY ERRORS?
7528 042774 001421                    BEQ   1$        ; BRANCH IF NO
7529 042776 105737 002075          1$:  TSTB  2#RP4TRY    ; TRIED 3 TIMES?
7530 043002 001640                    BEQ   RP4ERR    ; BRANCH IF YES
7531 043004 005337 002066          DEC  2#RP4FUN    ; SET FUNCTION TO A READ
7532 043010 052777 000040 137136  BIS  #BITS,2#RP4CS2 ; CLEAR ALL ERRORS
7533 043016 004737 040300          JSR   PC,2#LDRP4  ; RELOAD THE UNIT NO
7534 043022 105237 002075          INCB 2#RP4TRY    ; INCREMENT TRY COUNT
7535 043026 013746 177776          MOV  2#PSW,-(SP)
7536 043032 012746 042720          MOV  #RP43,-(SP)
7537 043036 000002                    RTI
7538 043040 112737 000200 001706  1$:  MOVB  #200,2#RP4HSTA  ; TRY AGAIN
7539 043046 000002                    RTI              ; SET DONE FLAG
7540
7541
;*****
;SBTTL RH70/R504 SERVICE ROUTINE

```

```

7543
7544 043050 000005
7545 043052 005337 002070
7546 043058 022737 000001 002070
7547 043064 001465
7548 043066 002542
7549 043070 000137 040544
7550 043074 005237 002070
7551 043100 022737 000002 002070
7552 043106 001470
7553 043110 100550
7554
7555
7556 043112 032737 000400 001710
7557 043120 001034
7558 043122 032777 040000 137070
7559 043130 001443
7560 043132 105737 002076
7561 043136 001412
7562 043140 052777 000040 137046
7563 043146 105237 002076
7564 043152 013746 177776
7565 043156 012746 040544
7566 043162 000002
7567 043164 012737 100200 001710
7568 043172 010046
7569 043174 013700 001576
7570 043200 052760 100000 001642
7571 043206 012600
7572 043210 000002
7573
7574 043212 012737 100200 001710
7575 043220 032777 040000 136772
7576 043226 001003
7577 043230 042737 100000 001710
7578 043236 000002
7579
7580 043240 112737 177775 002076
7581 043246 105777 136746
7582 043252 001775
7583 043254 004737 040604
7584 043260 112777 000151 136714
7585 043266 000002
7586
7587
7588 043270 032737 000400 001710
7589 043276 001345
7590 043300 032777 040000 136712
7591 043306 001417
7592 043310 105737 002076
7593 043314 001723
7594 043316 005337 002070
7595 043322 052777 000040 136664
7596 043330 105237 002076
7597 043334 013746 177776

:*****
RSRPT: RESET
DEC @RSFUN ; RESTORE FUNCTION
CMP #1,@RSFUN ; WHAT IS IT?
BEQ RS41 ; BRANCH IF WC
BLT RS43 ; BRANCH IF WRITE
JMP @RSWTRY
RSSRV: INC @RSFUN ; FIND OUT WHAT FUNCTION
CMP #2,@RSFUN ; WAS JUST EXECUTED
BEQ RSWCK
BMI RSREAD

;WRITE FUNCTION WAS JUST EXECUTED
BIT #BIT8,@RSHSTAT ; REPEAT FLAG SET?
BNE RSLOOP ; BRANCH IF YES
BIT #BIT14,@RSDS ; ANY ERRORS?
BEQ RS41 ; BRANCH IF NO
TSTB @RSTRY ; TRIED 3 TIMES?
BEQ RSERR ; BRANCH IF YES
BIS #BITS,@RSCS2 ; CLEAR ALL ERRORS
INCB @RSTRY ; INCREMENT TRY COUNT
MOV @PSW,-(SP) ; SETUP THE STACK TO
MOV @RSWTRY,-(SP) ; TRY THE WRITE AGAIN
RSERR: MOV #100200,@RSHSTAT ; SET ERROR AND DONE BIT
MOV RO,-(SP) ; SAVE RO
MOV @RS11,RO ; GET RUN TBL INDEX
BIS #BIT15,RUNTRAK(RO) ; SET ERROR BIT
MOV (SP)+,RO ; RESTORE RO

RSLOOP: MOV #100200,@RSHSTAT ; SET DONE AND ERROR BITS
BIT #BIT14,@RSDS ; ANY ERRORS?
BNE 1$ ; BRANCH IF YES
BIC #BIT15,@RSHSTAT ; CLEAR ERROR BIT
RTI ; RETURN
1$:
;WRITE OK...NOW DO A WRITE CHECK
RS41: MOVB #-3,@RSTRY ; INIT TRY COUNT
RS42: TSTB @RSDS ; IS DRIVE READY?
BEQ RS42 ; BRANCH IF NO
JSR @LDRS ; LOAD RS REGISTERS
MOVB #151,@RSCS1 ; LOAD FUNCTION AND GO
RTI ; RETURN

;FUNCTION JUST EXECUTED WAS A WRITE CHECK
RSWCK: BIT #BIT8,@RSHSTAT ; REPEAT FLAG SET?
BNE RSLOOP ; BRANCH IF YES
BIT #BIT14,@RSDS ; ANY ERRORS?
BEQ 1$ ; BRANCH IF NO
TSTB @RSTRY ; TRIED 3 TIMES?
BEQ RSERR ; BRANCH IF YES
DEC @RSFUN ; SET FUNCTION BACK TO WC
BIS #BITS,@RSCS2 ; CLEAR THE ERROR
INCB @RSTRY ; INCREMENT THE TRY COUNT
MOV @PSW,-(SP)

```

```

7599 043344 J00002 RTI ;TRY AGAIN
7600
7601 043346 032777 040000 136640 1S: BIT #BIT14, @RSCS2 ;WRITE CHECK ERROR?
7602 043354 001404 BEQ 25 ;BRANCH IF NO
7603 043356 005737 001576 TST @#RS11 ;FIRST 2K?
7604 043362 001401 BEQ 25 ;BRANCH IF YES
7605 043364 000751 BR 35
7606
7607 ;WRITE CHECK WAS OK...NOW DO A READ.
7608 043366 112737 177775 002076 25: MOV# #-3, @#RSTRY ;INIT TRY COUNT
7609 043374 105777 136620 RS43: TSTB @RSOS ;IS DRIVE READY?
7610 043400 001775 BEQ RS43 ;BRANCH IF NO
7611 043402 004737 040604 JSR PC, @#LDRS ;LOAD RS REGISTERS
7612 043406 013777 002010 136574 MOV @#ASNEWH, @#RSBAE ;LOAD BAE
7613 043414 013777 002006 136564 MOV @#ASNEWL, @#RSBA ;LOAD BUS ADR
7614 043422 112777 000171 136552 MOV# @#171, @#RSCS1 ;LOAD FUNCTION AND GO
7615 043430 000002 RTI ;RETURN
7616
7617 ;FUNCTION JUST EXECUTED WAS A READ.
7618 043432 032737 000400 001710 RSREAD: BIT #BIT8, @#RSHSTAT ;REPEAT FLAG SET?
7619 043440 001264 BNE RSL0OP ;BRANCH IF YES
7620 043442 032777 040000 136550 BIT #BIT14, @#RSDS ;ANY ERRORS?
7621 043450 001417 BEQ 15 ;BRANCH IF NO
7622 043452 105737 002076 TSTB @#RSTRY ;TRIED 3 TIMES?
7623 043456 001642 BEQ RSERR ;BRANCH IF YES
7624 043460 005337 002070 DEC @#RSFUN ;RESTORE FUN TO READ
7625 043464 052777 000040 136522 BIS #BITS, @#RSCS2 ;CLEAR ALL ERRORS
7626 043472 105237 002076 INCB @#RSTRY ;INCREMENT TRY COUNT
7627 043476 013746 177776 MOV @#PSW, -(SP)
7628 043502 012746 043374 MOV @#RS43, -(SP)
7629 043506 000002 RTI ;TRY AGAIN
7630 043510 112737 000200 001710 1S: MOV# @#200, @#RSHSTAT ;SET DONE FLAG
7631 043516 000002 RTI ;RETURN
7632
7633 ;*****
7634 ;SBTTL UNIBUS EXERCISER SERVICE ROUTINE
7635 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
7636 ;*****
7637 UBESRV: SAVREG
7638 JSR PC, @#LDKT ;GO TO LOW CORE
7639 MOV #UBETBL+6, R4 ;GET ADDRESS OF UBECR1
7640 TST @#(R4) ;WAS THERE AN ERROR?
7641 BMI UBE2 ;BRANCH IF YES
7642 MOV #3, -(SP) ;PUT DEVICE ID IN STACK
7643 JSR PC, @#GIVEMAP ;GIVE UP MAP REG
7644 ADD #776, @#UBESAV ;INCREMENT UBE BUS ADR
7645 ADC @#UBESAV+2
7646 MOV @#UBESAV, @#UBEADR
7647 MOV @#UBESAV+2, @#UBEADR+2
7648 MOV @#UBEADR, -(SP)
7649 JSR PC, @#GETMAP
7650 MOV @#UBEADR+2, @#-(R4) ;LOAD UBECR2
7651 MOV @#UBEADR, @#-(R4) ;LOAD UBEBA
7652 MOV @#170000, @#-(R4) ;LOAD UBEC
7653 JSR PC, @#RESKT ;GO BACK TO ORIGINAL CORE
RESREG

```

```

7655 043636 000002 RTI ;RETURN
7656
7657 ;UBE ERROR-IS IT LAST MEMORY?
7658 UBE2: CLR 2#STMP0
7659 SUB 4,R4 ;ADJUST R4
7660 MOV 2(R4),R3 ;GET BECR2
7661 BIC 3,R3 ;GET RID OF ADDRESS BITS
7662 CMP 400,R3 ;WAS ERROR A TIMEOUT?
7663 BNE UBERRA ;BRANCH IF NO
7664 MOV 2(R4),2#ERRBA ;SAVE BUS ADR OF ERROR
7665 MOV 2(R4),2#ERRBA+2
7666 BIC 177774,2#ERRBA+2
7667 JSR PC,2#PHYMAP ;GET PHYSICAL ADDRESS THAT TIMED OUT
7668 SUB 4,2#PA1500 ;ADJUST PHYSICAL ADR THAT FAILED
7669 SBC 2#PA2116 ;UBE STOPS AT ADR+4
7670 CMP 2#PA1500,2#MXMML0 ;AT MAXIMUM MEMORY L0?
7671 BNE MHOLE ;BRANCH IF NO
7672 CMP 2#PA2116,2#MXMMHI ;AT MAX MEMORY HI?
7673 BNE MHOLE ;BRANCH IF NO
7674 MOV 3,-(SP) ;PUT DEVICE ID ON STACK
7675 JSR PC,2#GIVEMAP
7676 TST (SP)+
7677 JSR PC,2#UBEINIT
7678 JSR PC,2#RESKT
7679 RESREG
7680 MOV 64545,2#UBETBL+6
7681 RTI
7682
7683 MHOLE: MOV SP,2#STMP0
7684 UBEERR: MOV 2#SLPERR,2#STMP2 ;SAVE LOOP ERROR ADR
7685 MOV 2#UBERR,2#SLPERR ;SET LOOP ADR
7686 MOV 22,R3
7687 TST 2#STMP0
7688 BNE 15
7689 ERROR 7
7690 BR UBE3
7691 15: MOV 2#PA1500,2#SGDDAT
7692 MOV 2#PA2116,2#SBDDAT
7693 ERROR 12
7694
7695 ;RESTART UBE IN SAME MEMORY
7696 UBE3: MOV 2#STMP2,2#SLPERR ;RESTORE ERROR LOOP ADR
7697 MOV R4,-(SP) ;SAVE R4
7698 MOV 2#UBETBL,R4 ;GET ADDRESS OF UBE TABLE
7699 MOV 170000,2(R4)+ ;SET UBEC
7700 MOV 2#UBEADR,2(R4)+ ;SET UBEBA <15:00>
7701 CLR 2(R4) ;CLEAR ALL ERRORS
7702 MOV 2#UBEADR+2,2(R4)+ ;SET EXT ADR BITS
7703 MOV 64545,2(R4) ;START UBE
7704 MOV (SP)+,R4 ;RESTORE R4
7705 JSR PC,2#RESKT
7706 RESREG
7707 RTI ;RETURN
7708
7709 ;*****

```

```

7711 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
7712 ;*****
7713 MBTSRV: SAVREG
7714 JSR PC, @LDKT ;GO TO LOW CORE
7715 CLR @STMPD
7716 MOV @MBTTBL, R4 ;GET ADDRESS OF ADDRESS OF CS1 REG
7717 BIT @BIT14, @R4+ ;ANY ERRORS?
7718 BNE 15 ;BRANCH IF YES
7719 JSR PC, @RESKT ;GO BACK TO ORIGINAL CORE
7720 RESREG
7721 MOV @161, @MBTTBL
7722 RTI ;RESTART MBT AND RETURN
7723 ADD @10, R4 ;ADJUST R4
7724 BIT @BIT11, @R4 ;NON-EXISTANT MEMORY ERROR?
7725 BEQ MBTERR ;BRANCH IF NO
7726 SUB @6, R4 ;ADJUST R4
7727 MOV @R4+, @PA1500 ;GET BUS ADR
7728 MOV @R4+, @PA2116 ;GET BUS ADR EXT
7729 SUB @4, @PA1500 ;ADJUST BUS ADR
7730 SBC @PA2116
7731 CMP @PA1500, @MXMML0 ;IS IT LAST MEMORY?
7732 BNE MEMHOLE ;BRANCH IF NO
7733 CMP @PA2116, @MXMMHI ;CHECK EXT ADR BITS
7734 BNE MEMHOLE
7735 TST (R4)+ ;INCREMENT R4
7736 BIS @47, @R4 ;CLEAR THE ERROR
7737 MOV @7, @R4+ ;SELECT UNIT 7
7738 CLR @-12(R4) ;CLEAR WORD COUNT
7739 BR 25 ;CONTINUE
7740
7741 MEMHOLE: MOV SP, @STMPD
7742 MBTERR: MOV @SLPERR, @STMP2 ;SAVE LOOP ADDRESS
7743 MOV @15, @SLPERR ;SET NEW LOOP ADR
7744 MOV @20, R3 ;PUT DEVICE ID IN R3
7745 TST @STMPD
7746 BNE 25
7747 ERROR 7
7748 BR 15
7749 MOV @PA1500, @SGDDAT 25:
7750 MOV @PA2116, @SBDDAT
7751 ERROR 13
7752 MOV @STMP2, @SLPERR 15:
7753 MOV @MBTTBL+10, R4 ;RESTORE LOOP ADR
7754 MOV @-(R4), R0 ;GET ADR OF MBTTBL+10
7755 MOV @-(R4), R1 ;GET BUS ADR EXTENDED
7756 MOV @-(R4), R2 ;GET BUS ADR
7757 ASL R2 ;GET WORD COUNT
7758 SUB R2, R1 ;ADJUST WORD COUNT
7759 SBC R0 ;FORM START ADR OF THIS XFER
7760 BIS @47, @10(R4) ;CLEAR THE WORLD
7761 MOV @7, @10(R4) ;SELECT UNIT 7
7762 TST (R4)+ ;ADJUST R4
7763 MOV R1, @R4+ ;RESTORE BUS ADR
7764 MOV R0, @R4
7765 JSR PC, @RESKT ;GO BACK TO ORIGINAL CORE

```



```

7767 044434 112777 000161 135604      MOVB    #161,2MBTTBL      ;START MBT AGAIN
7768 044442 000002                    RTI                    ;RETURN
7769                                     ;*****
7770 .SBTTL  LINE CLOCK SERVICE ROUTINE
7771 *      THIS ROUTINE FIRST REMAPS PROGRAM EXECUTION TO LOW
7772 *      MEMORY. IT THEN INCREMENTS AND KEEPS TRACK OF THE
7773 *      SECOND AND MINUTE COUNTS KEPT IN LOCATIONS "LTICKS"
7774 *      AND "MTICKS" RESPECTIVELY.
7775                                     ;*****
7776 044444 104412      LKSRV: SAVREG
7777 044446 004737 053130      JSR     PC,2#LDKT        ;GO TO LOW CORE
7778 044452 105237 001602      INCB   2#LTICKS        ;INCREMENT TICK COUNT
7779 044456 122737 000074 001602      CMPB   #60.,2#LTICKS   ;ONE SECOND YET?
7780 044464 001014                    BNE    1$              ;BRANCH IF NO
7781 044466 105237 001603      INCB   2#LTICKS+1      ;INCREMENT SECOND COUNT
7782 044472 105037 001602      CLRB   2#LTICKS        ;CLEAR SECOND COUNT
7783 044476 122737 000074 001603      CMPB   #60.,2#LTICKS+1 ;ONE MINUTE YET?
7784 044504 001004                    BNE    1$              ;BRANCH IF NO
7785 044506 105037 001603      CLRB   2#LTICKS+1      ;INCREMENT MINUTE COUNT
7786 044512 005237 001600      INC    2#MTICKS        ;INCREMENT MINUTE COUNT
7787 044516 004737 053216      1$:   JSR     PC,2#RESKT   ;RESTORE THE KT
7788 044522 104414      RESREG
7789 044524 012737 000100 177546      MOV    #BIT6,2#LKS     ;CLEAR READY BIT IN CLOCK
7790 044532 000002                    RTI                    ;RETURN
7791                                     ;*****
7792 .SBTTL  SCOPE HANDLER ROUTINE
7793
7794 *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
7795 *AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG. (DISPLAY<7:0>)
7796 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
7797 *$SW14=1      LOOP ON TEST
7798 *$SW11=1      INHIBIT ITERATIONS
7799 *$SW09=1      LOOP ON ERROR
7800 *CALL        SCOPE          ;;SCOPE=IOT
7801
7802 $SCOPE:
7803 BIT     #SW14,2#SWR      ;;LOOP ON PRESENT TEST?
7804 BNE     $OVER           ;;YES IF SW14=1
7805 *****START OF CODE FOR THE XOR TESTER*****
7806 $XTSTR: BR     6$
7807                                     ;IF RUNNING ON THE "XOR" TESTER CHANGE
7808                                     ;THIS INSTRUCTION TO A "NOP" (NOP=240)
7809 MOV     2#ERRVEC, -(SP)  ;SAVE THE CONTENTS OF THE ERROR VECTOR
7810 MOV     #55,2#ERRVEC    ;SET FOR TIMEOUT
7811 TST    2#177060         ;TIME OUT ON XOR?
7812 MOV     (SP)+,2#ERRVEC  ;RESTORE THE ERROR VECTOR
7813 BR     $SVLAD          ;GO TO THE NEXT TEST
7814 CMP    (SP)+,(SP)+     ;CLEAR THE STACK AFTER A TIME OUT
7815 MOV    (SP)+,2#ERRVEC  ;RESTORE THE ERROR VECTOR
7816 BR     7$             ;LOOP ON THE PRESENT TEST
7817 5$:   BR     7$
7818 6$:   *****END OF CODE FOR THE XOR TESTER*****
7819 2$:   TSTB   $ERFLG     ;HAS AN ERROR OCCURRED?
7820 BEQ    3$             ;BR IF NO
7821

```

```

7823 044616 101015          BHI      3$          ;; BR IF NO
7824 044620 032737 001000 177570  BIT      #BIT09,2#SWR  ;; LOOP ON ERROR?
7825 044626 001404          BEQ      4$          ;; BR IF NO
7826 044630 016767 134356 134352 7$:  MOV     $LPERR,$LPADR  ;; SET LOOP ADDRESS TO LAST SCOPE
7827 044636 000441          BR       $OVER        ;;
7828 044640 105067 134340          4$:  CLRB   $ERFLG        ;; ZERO THE ERROR FLAG
7829 044644 005067 134446          CLR     $TIMES        ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
7830 044650 000415          BR       1$          ;; ESCAPE TO THE NEXT TEST
7831 044652 032737 004000 177570 3$:  BIT      #BIT11,2#SWR  ;; INHIBIT ITERATIONS?
7832 044660 001011          BNE     1$          ;; BR IF YES
7833 044662 105757 134312          TSTB   $PASS         ;; IF FIRST PASS OF PROGRAM
7834 044666 001406          BEQ     1$          ;; INHIBIT ITERATIONS
7835 044670 005267 134312          INC     $ICNT        ;; INCREMENT ITERATION COUNT
7836 044674 026767 134416 134304  CMP     $TIMES,$ICNT  ;; CHECK THE NUMBER OF ITERATIONS MADE
7837 044702 002017          BGE     $OVER        ;; BR IF MORE ITERATION REQUIRED
7838 044704 012767 000001 134274 1$:  MOV     #1,$ICNT     ;; REINITIALIZE THE ITERATION COUNTER
7839 044712 016767 000054 134376          MOV     $MXCNT,$TIMES  ;; SET NUMBER OF ITERATIONS TO DO
7840 044720 011667 134264          $SVLAD: MOV    (SP),$LPADR  ;; SAVE SCOPE LOOP ADDRESS
7841 044724 011667 134262          MOV    (SP),$LPERR    ;; SAVE ERROR LOOP ADDRESS
7842 044730 005067 134364          CLR     $ESCAPE     ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
7843 044734 112767 000001 134255          MOV    #1,$ERMAX    ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
7844 044742 105767 134236          $OVER: TSTB   $ERFLG  ;; ANY ERRORS?
7845 044746 001403          BEQ     1$          ;; BRANCH IF NO
7846 044750 116737 134230 001203          MOV    $ERFLG,2#$STSTNM+1
7847 044756 016737 134220 177570 1$:  MOV     $STSTNM,2#$DISPLAY  ;; DISPLAY TEST NUMBER
7848 044764 016716 134220          MOV     $LPADR,(SP)  ;; FUDGE RETURN ADDRESS
7849 044770 000002          RTI                    ;; FIXES PS
7850 044772 000010          $MXCNT: 10          ;; MAX. NUMBER OF ITERATIONS
7851          ;;*****
7852          .SBTTL  ERROR HANDLER ROUTINE
7853
7854
7855          ;;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
7856          ;;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
7857          ;;AND GO TO SERRTYP ON ERROR
7858          ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
7859          ;;SW15=1          HALT ON ERROR
7860          ;;                HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
7861          ;;SW13=1          INHIBIT ERROR TYPEOUTS
7862          ;;SW10=1          BELL ON ERROR
7863          ;;SW09=1         LOOP ON ERROR
7864          ;;CALL
7865          ;;          ERROR      N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
7866
7867          $ERROR:
7868 044774 116737 134204 001203          MOV    $ERFLG,2#$STSTNM+1
7869 045002 105267 134176          7$:  INCB   $ERFLG        ;; SET THE ERROR FLAG
7870 045006 001775          BEQ     7$          ;; DON'T LET THE FLAG GO TO ZERO
7871 045010 016737 134166 177570          MOV    $STSTNM,2#$DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
7872 045016 005737 177570          TST    2#SWR        ;; HALT ON ERROR = 1?
7873 045022 100001          BPL     8$          ;; BRANCH IF NO
7874 045024 000000          HALT                    ;; YES--HALT
7875 045026 032737 002000 177570 8$:  BIT      #BIT10,2#SWR  ;; BELL ON ERROR?
7876 045034 001402          BEQ     1$          ;; NO - SKIP
7877 045036 104400 001322          TYPE   ,SBELL        ;; RING BELL
    
```

```

7879 045046 011667 134146      MOV      (SP),SERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
7880 045052 162767 000002 134140  SUB      #2,SERRPC
7881 045060 117767 134134 134130  MOVB    JSERRPC,SITEMB    ;;STRIP AND SAVE THE ERROR ITEM CODE
7882 045066 032737 020000 177570  BIT     #BIT13,#SWR      ;;SKIP TYPEOUT IF SET
7883 045074 001004      BNE     2$              ;;SKIP TYPEOUTS
7884 045076 004767 000056  JSR     PC,SERRTYP      ;;GO TO USER ERROR ROUTINE
7885 045102 104400 001327      TYPE   $CRLF
7886 045106 005737 177570 2$:    TST     #SWR          ;;HALT ON ERROR
7887 045112 100001      BPL     9$              ;;SKIP IF CONTINUE
7888 045114 000000      HALT                    ;;HALT ON ERROR!
7889 045116 022767 036752 132716 9$:    CMP     #SENDAD,42     ACT-11?
7890 045124 001001      BNE     3$              ;;BRANCH IF NO
7891 045126 000000      HALT                    YES
7892 045130 032737 001000 177570 3$:    BIT     #BIT09,#SWR    ;;LOOP ON ERROR SWITCH SET?
7893 045136 001402      BEQ     4$              BR IF NO
7894 045140 016716 134046  MOV     $LPERR,(SP)     FUDGE RETURN FOR LOOPING
7895 045144 005767 134150 4$:    TST     $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
7896 045150 001402      BEQ     5$              BR IF NONE
7897 045152 016716 134142  MOV     $ESCAPE,(SP)   ;;FUDGE RETURN ADDRESS FOR ESCAPE
7898 045156      5$:
7899 045156 000002      RTI                    ;;RETURN

```

```

7900 :*****
7901 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
7902
7903 :*THIS ROUTINE FIRST TYPES A STANDARD MESSAGE CONSISTING OF THE
7904 :*VIRTUAL PC, THE PHYSICAL PC, THE PSW AT THE TIME OF THE ERROR CALL,
7905 :*AND THE SUB-PASS COUNT. THE SUB-PASS COUNT CONSISTS OF THE SUB PASS COUNT IN THE
7906 :*HIGH BYTE AND THE PASS COUNT IN THE LOW BYTE.
7907 :*
7908 :*IT THEN USES THE "CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
7909 :*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE"
7910 :*THE ERROR MESSAGE POINTER AND TYPES THE ERROR MESSAGE. THE DATA
7911 :*HEADER POINTER IS THEN OBTAINED AND A DATA HEADER IS TYPED.
7912 :*THE DATA POINTER AND DATA FORMAT ARE THEN OBTAINED. THERE ARE
7913 :*FOUR TYPES OF DATA FORMAT, AS FOLLOWS:
7914 :*
7915 :* 0 TYPE THE CONTENTS OF THE DATA TABLE WORD IN
7916 :* 5 DIGIT OCTAL FORMAT
7917 :* 1 CONVERT THE CONTENTS OF THE DATA TABLE WORD TO
7918 :* 22 BITS AND TYPE AN 8 DIGIT OCTAL NUMBER
7919 :* 2 TYPE THE CONTENTS OF THE DATA TABLE WORD AND
7920 :* THE WORD+2 IN 8 DIGIT OCTAL FORMAT
7921 :* 3 USE THE CONTENTS OF THE DATA TABLE WORD AS A
7922 :* DEVICE ID AND TYPE THE DEVICES NAME
7923 :* 4 CONVERT THE TWO WORDS POINTED TO BY THE DATA
7924 :* TABLE TO FLOATING POINT FORMAT AND TYPE.
7925 :* 5 CONVERT THE FOUR WORDS POINTED TO BY THE DATA
7926 :* TABLE TO FLOATING DOUBLE FORMAT AND TYPE
7927 :*****

```

```

7928
7929 045160 104412      SERRTYP: SAVREG
7930 045162 104400 001327      TYPE   $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
7931 045166 004737 046660  JSR     PC,#TYPTIME    ;;GO TYPE THE TIME
7932 045172 104400 053662      TYPE   ,MSG3
7933 045176 104400 001327      TYPE   ,CRLF

```

```

7935                                     ;; TYPE THE VIRTUAL PC
7936 045206 104402 TYP0C                                     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
7937 045210 104400 046432 TYPE 08$
7938 045214 013700 001474 MOV 0#VAOR,RO ;SAVE VAOR
7939 045220 013737 001220 MOV 0#ERRPC,0#VAOR ;SAVE THE VIR PC FOR CONVERSION
7940 045226 122737 000014 CMPB 014,0#ITEMB
7941 045230 003403 BLE 51$
7942 045234 105737 001216 TSTB 0#ITEMB ;ERROR ZERO?
7943 045238 001005 BNE 42$ ;BRANCH IF NO
7944 045244 004737 051644 JSR PC,0#CNVAOR ;CONVERT TO 22 BITS
7945 045250 010037 001474 MOV RO,0#VAOR
7946 045254 000407 BR 41$
7947 045258 013737 001474 001476 42$: MOV 0#VAOR,0#PA1500
7948 045264 005037 001500 CLR 0#PA2116
7949 045270 010037 001474 MOV RO,0#VAOR
7950 045274 012746 001476 41$: MOV 0#PA1500,-(SP) ;PUT ADDRESS OFPC ON STACK
7951 045300 004737 047712 JSR PC,0#SOB20 ;CONVERT TO ASCII
7952 045304 062716 000003 ADD 03,(SP) ;GET RID OF 3 MS DIGITS
7953 045310 012667 000002 MOV (SP)+,30$ ;SAVE POINTER TO ASCII
7954 045314 104400 TYPE ;TYPE IT
7955 045316 000000 30$: .WORD
7956 045320 104400 046432 TYPE 08$
7957 045324 016646 000030 MOV 30(SP),-(SP) ;GET PSW AT TIME OF ERROR
7958 045330 104402 TYP0C ;TYPE IT
7959 045332 104400 046432 TYPE 08$
7960 045336 016746 134242 MOV $MAINT,-(SP) ;SAVE $MAINT FOR TYPEOUT
7961 045342 104402 TYP0C ;TYPE THE MAINTENANCE REG
7962 045344 104400 046432 ;GO TYPE--OCTAL ASCII(ALL DIGITS)
7963 045348 116746 133626 MOVB $TSTNM,-(SP)
7964 045350 105066 000001 CLRB 1(SP)
7965 045354 104402 TYP0C ;TYPE THE TEST NUMBER
7966 045358 104400 046432 TYPE 08$
7967 045362 013746 001532 MOV 0#SUBPASS,-(SP)
7968 045366 162716 000060 SUB 060,(SP)
7969 045372 104402 TYP0C
7970 045376 104400 046432 TYPE 08$
7971 045400 133570 MOV $PASS,-(SP) ;SAVE $PASS FOR TYPEOUT
7972 045404 104402 TYP0C ;TYPE THE PASS COUNT
7973 045410 104400 001327 ;GO TYPE--OCTAL ASCII(ALL DIGITS)
7974 045412 005000 CLR RO
7975 045416 153700 001216 BISB 0#ITEMB,RO ;PICK UP THE INDEX
7976 045420 001431 BEQ 6$ ;EXIT IF ZERO
7977 045424 022700 000007 1$: CMP 07,RO ;IS THIS ERROR 7?
7978 045426 001551 BEQ 15$ ;BRANCH IF YES
7979 045432 005300 DEC RO ;ADJUST THE INDEX SO THAT IT WILL
7980 045434 006300 ASL RO ; WORK FOR THE ERROR TABLE
7981 045436 006300 ASL RO
7982 045440 006300 ASL RO
7983 045442 002304 ADD 0#ERRTB,RO ;FORM TABLE POINTER
7984 045444 012067 000004 MOV (RO)+,2$ ;PICKUP "ERROR MESSAGE" POINTER
7985 045450 001404 BEQ 3$ ;SKIP TYPEOUT IF NO POINTER
7986 045454 104400 TYPE ;TYPE THE "ERROR MESSAGE"
7987 045456 000000 2$: .WORD 0 ;"ERROR MESSAGE" POINTER GOES HERE
7988
7989

```

```

7991 045466 012067 000004      3$:  MOV      (R0)+,4$      ;; PICKUP "DATA HEADER" POINTER
7992 045472 001404              BEQ      5$              ;; SKIP TYPEOUT IF 0
7993 045474 104400              TYPE                    ;; TYPE THE "DATA HEADER"
7994 045476 000000      4$:  .WORD      0              ;; "DATA HEADER" POINTER GOES HERE
7995 045500 104400 001327      TYPE      $SCRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
7996 045504 012001      5$:  MOV      (R0)+,R1        ;; PICKUP "DATA TABLE" POINTER
7997 045506 001004              BNE      7$              ;; GO TYPE THE DATA
7998 045510 104414      6$:  RESREG                    ;; "CARRIAGE RETURN" & "LINE FEED"
7999 045512 104400 001327      TYPE      $SCRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
8000 045516 000207              RTS      PC              ;; RETURN
8001 045520 011002      7$:  MOV      (R0),R2        ;; GET "DATA FORMAT" POINTER
8002 045522 122712 000001      10$: CMPB     #1,(R2)        ;; DATA FORMAT 1?
8003 045526 001424              BEQ      9$              ;; BRANCH IF YES
8004 045530 122712 000002      CMPB     #2,(R2)        ;; DATA FORMAT 2?
8005 045534 001441              BEQ      11$             ;; BRANCH IF YES
8006 045536 122712 000003      CMPB     #3,(R2)        ;; DATA FORMAT 3?
8007 045542 001445              BEQ      24$             ;; BRANCH IF YES
8008 045544 122712 000004      CMPB     #4,(R2)        ;; DATA FORMAT 4?
8009 045550 001456              BEQ      40$             ;; BRANCH IF YES
8010 045552 122712 000005      CMPB     #5,(R2)        ;; DATA FORMAT 5?
8011 045556 001465              BEQ      60$             ;; BRANCH IF YES
8012
8013 ;:*****
8014 ;:DATA FORMAT 0
8015 045560 005202              INC      R2              ;; INCREMENT FORMAT POINTER
8016 045562 013146              MOV      @R1+,-(SP)      ;; PUSH DATA TO BE TYPED
8017 045564 104402              TYPOC                    ;;
8018 045566 005711      13$: TST      (R1)          ;; ANY MORE DATA?
8019 045570 001747              BEQ      6$              ;; BRANCH IF NO
8020 045572 104400 046432      TYPE      8$              ;; TYPE TWO SPACES
8021 045576 000751              BR      10$
8022 ;:*****
8023 ;:DATA FORMAT 1
8024 045600 005202      9$:  INC      R2              ;; INCREMENT FORMAT POINTER
8025 045602 004737 051644      JSR      PC,@CNVADR      ;; GET 22 BIT ADR
8026 045606 012746 001476      MOV      #PA1500,-(SP)   ;; PUSH ADR OF 22 BIT ADR
8027 045612 004737 047712      JSR      PC,@SDB20      ;; CONVERT TO ASCII
8028 045616 062716 000003      ADD      #3,(SP)        ;; DELETE LEADING ZEROS
8029 045622 012667 000002      MOV      (SP)+,12$      ;; GET ADR OF ASCII STRING
8030 045626 104400              TYPE                    ;;
8031 045630 000000      12$: .WORD      0              ;;
8032 045632 062701 000002      ADD      #2,R1          ;; INCREMENT R1
8033 045636 000753              BR      13$
8034 ;:*****
8035 ;:DATA FORMAT 2
8036 045640 005202      11$: INC      R2              ;; INCREMENT FORMAT POINTER
8037 045642 011100              MOV      (R1),R0
8038 045644 012037 001476      MOV      (R0)+,@PA1500
8039 045650 011037 001500      MOV      (R0)+,@PA2116
8040 045654 000754              BR      14$
8041 ;:*****
8042 ;:DATA FORMAT 3
8043 045656 005202      24$: INC      R2              ;; INCREMENT FORMAT POINTER
8044 045660 013167 000016      MOV      @R1+,25$      ;; GET DEVICE ID
8045 045664 062767 053612 000010      ADD      #MSGINX,25$   ;; FORM ADR OF ASCII ADR
8046 045672 017767 000004 000002      MOV      @25$,25$      ;; GET ADR OF ASCII

```

```

8047 045702 000000
8048 045704 000730
8049
8050
8051 045706 005202
8052 045710 012167 000002
8053 045714 104416
8054 045716 000000
8055 045720 012667 000002
8056 045724 104400
8057 045726 000000
8058 045730 000716
8059
8060
8061 045732 005202
8062 045734 012167 000002
8063 045740 104420
8064 045742 000000
8065 045744 012667 000002
8066 045750 104400
8067 045752 000000
8068 045754 000704
8069
8070
8071 045756 010300
8072 045760 062700 053612
8073 045764 011067 000002
8074 045770 104400
8075 045772 000000
8076 045774 104400 046002
8077 046000 000404
8078
8079 046012
8080 046012 010300
8081 046014 022700 000010
8082 046020 003403
8083 046022 104400 054116
8084 046026 000411
8085
8086
8087 046030 022703 000020
8088 046034 001426
8089 046036 002435
8090 046040 104400 054231
8091 046044 022700 000012
8092 046050 001140
8093
8094
8095 046052 062700 053566
8096 046056 011000
8097 046060 022703 000002
8098 046064 001404
8099 046066 100406
8100 046070 012704 000007
8101 046074 000423

```

```

25$: WORD
BR 13$ ;CONTINUE
:*****
:DATA FORMAT 4
40$: INC R2
MOV (R1)+,44$ ;GET ADDRESS OF DATA
FLD20 ;CONVERT TO FLOATING FORMAT
44$: .WORD
MOV (SP)+,45$ ;GET ADDRESS OF ASCIZ STRING
TYPE ;TYPE THE DATA
45$: .WORD
BR 13$
:*****
:DATA FORMAT 5
60$: INC R2 ;INCREMENT FORMAT POINTER
MOV (R1)+,61$ ;GET ADDRESS OF DATA
FLD20 ;CONVERT TO FLOATING ASCIZ
61$: .WORD
MOV (SP)+,62$ ;GET ADDRESS OF ASCIZ STRING
TYPE ;TYPE THE DATA
62$: .WORD
BR 13$
:*****
:ERROR 7 DECODE
15$: MOV R3,RO ;SAVE R3
ADD #MSGINX,RO ;GEN ADRS OF ASCIZ
MOV (RO),16$
TYPE
16$: .WORD
TYPE 65$ ;:TYPE ASCIZ STRING
BR 64$ ;:GET OVER THE ASCIZ
:65$: .ASCIZ /FAILED/<CRLF>
64$: MOV R3,RO ;SAVE DEVICE ID
CMP #10,RO ;:MASS BUS DEVICE?
BLE 17$ ;:BRANCH IF YES
TYPE MSG12
BR 18$
:*****
:MASS BUS ERR
17$: CMP #20,R3 ;:MBT ERROR?
BEQ 26$ ;:BRANCH IF MBT ERROR
BLT 27$ ;:BRANCH IF UBE ERROR
TYPE MSG13
CMP #12,RO ;:WAS IT RS?
BNE 29$ ;:BRANCH IF NO
:*****
:UNIBUS ERROR OR RS04 ERROR
18$: ADD #REGINX,RO ;:FORM ADR OF REG TABLE
MOV (RO),RO ;:GET ADR OF REG TABLE
CMP #2,R3 ;:RP3 OR RK?
BEQ 20$ ;:BRANCH IF RK
BMI 21$ ;:BRANCH IF NOT RPO3
MOV #7,R4 ;:SET RPO3 SOB COUNT
BR 22$

```

```

8103 046102 000420
8104 046104 012704 000011
8105 046110 000415
8106
8107
8108 046112 104400 054420
8109 046116 012704 000011
8110 046122 062700 053566
8111 046126 011000
8112 046130 000405
8113
8114 046132 104400 054527
8115 046136 012704 000004
8116 046142 000767
8117 046144 013046
8118 046146 104402
8119 046150 104400 046432
8120 046154 077405
8121
8122
8123 046156 022703 000022
8124 046162 001454
8125 046164 022703 000002
8126 046170 002445
8127 046172 001005
8128
8129 046174 104400 054723
8130 046200 012700 002126
8131 046204 000404
8132
8133 046206 012700 002104
8134 046212 104400 054733
8135
8136 046216 013001
8137 046220 005720
8138 046222 013037 001672
8139 046226 072127 177774
8140 046232 042701 177774
8141 046236 010137 001674
8142 046242 162737 000002 001672 74$:
8143 046250 005637 001674
8144 046254 004737 051536
8145 046260 012746 001476
8146 046264 004737 047712
8147 046270 062716 000003
8148 046274 012667 000002
8149 046300 104400
8150 046302 000000
8151 046304 104400 001327

21$: BR 22$
MOV #11,R4 ;SET RS04 SOB COUNT
BR 22$
;*****
;MBT ERROR
26$: TYPE MSG16
MOV #11,R4 ;SET MBT SOB COUNT
28$: ADD #REGINX,RO
MOV (RO),RO ;GET ADR OF MBT TABLE
BR 22$ ;GO TYPE REGISTERS
;UNIBUS EXERCISER ERROR
27$: TYPE MSG17
MOV #4,R4 ;SET UBE SOB COUNT
BR 28$ ;GO TYPE UBE REGISTERS
22$: MOV @ (RO)+,-(SP) ;GET DATA IN REG
TYPOC ;TYPE IT
TYPE #8 ;TYPE TWO SPACES
SOB #4,22$ ;CONTINUE
;*****
;THIS CODE TYPES A PHYSICAL BUS ADDRESS IF THE ERROR WAS AN RPO3, RK05, OR UBE
CMP #22,R3 ;UBE ERROR?
BEQ 73$ ;BRANCH IF YES
CMP #2,R3 ;RK05?
BLT 32$ ;BRANCH IF NOT RK OR RPO3
BNE 70$ ;BRANCH IF RPO3
;RK05 ERROR
TYPE MSG22
MOV #RKCS,RO ;GET ADR OF ADR OF RKCS REG
BR 71$
;RPO3 ERROR
70$: MOV #RP3CS,RO ;GET ADR OF ADR OF RP3CS REG
TYPE MSG23
;GET, CALCULATE, & TYPE PHYSICAL BUS ADDRESS
71$: MOV @ (RO)+,R1 ;GET BUS ADR EXTENDED BITS
TST (RO)+ ;ADJUST RO
MOV @ (RO)+,@ERRBA ;GET BUS ADDRESS THAT FAILED
ASH #-4,R1 ;GET BITS 4&5 INTO BITS 0&1
BIC #177774,R1 ;GET RID OF UNUSED BITS
MOV R1,@ERRBA+2 ;SAVE EXTENDED BITS
SUB #2,@ERRBA ;DECREMENT BUS ADR
SBC @ERRBA+2
JSR PC,@PHYMAP ;GO CONVERT TO 22 BIT PHYSICAL
MOV #PA1500,-(SP)
JSR PC,@SOB20 ;CONVERT TO ASCIZ STRING
ADD #3,(SP) ;GET RID OF LEADING ZEROS
MOV (SP)+,72$
TYPE
72$: .WORD
32$: TYPE ,$CRLF

```

```

8153
8154 046314 012700 002232
8155 046320 013037 001672
8156 046324 013037 001674
8157 046330 042737 177774
8158 046336 162737 000002
8159 046344 005637 001674
8160 046350 000734
8161
8162
8163 046352 062700 053566
8164 046356 011000
8165 046360 012704 000011
8166 046364 013046
8167 046366 104402
8168 046370 104400 046432
8169 046374 077405
8170 046376 104400 001327
8171 046402 104400 001327
8172 046406 012704 000004
8173 046412 104400 054341
8174 046416 013046
8175 046420 104402
8176 046422 104400 046432
8177 046426 077405
8178 046430 000725
8179 046432 020040 000
8180
8181 046436
8182
8183
8184
8185
8186
8187
8188
8189
8190
8191
8192
8193
8194
8195
8196
8197
8198
8199
8200
8201
8202
8203 046436 105767 132611
8204 046442 100002
8205 046444 000000
8206 046446 000407
8207 046450 010046

```

```

;GET UBE VIRTUAL ADDRESS
73$: MOV #UBETBL+2,RO ;GET ADR OF UBE TABLE +2
MOV 2(RO)+,2#ERRBA ;GET BUS ADR THAT FAILED
MOV 2(RO)+,2#ERRBA+2;GET BAE BITS
BIC #177774,2#ERRBA+2;MASK OFF ADR BITS
SUB #2,2#ERRBA
SBC 2#ERRBA+2
BR 74$ ;GO CONVERT & TYPE PHYSICAL ADR
;*****
;RPO4 ERROR
29$: ADD #REGINX,RO
MOV (RO) RO ;FORM ADR OF RPO4 TABLE
MOV #11,R4 ;SET SOB COUNT
31$: MOV 2(RO)+,-(SP) ;GET DATA TO BE TYPED
TYPOC ;TYPE DATA
TYPE BS
SOB R4,31$ ;CONTINUE
TYPE ,SCLF
TYPE ,SCLF
MOV #4,R4 ;SET SOB COUNT
TYPE MSG14
50$: MOV 2(RO)+,-(SP) ;GET DTA TO BE TYPED
TYPOC ;TYPE IT
TYPE BS
SOB R4,50$ ;CONTINUE
BR 32$
8$: .ASCIZ / / ;;TWO(2) SPACES
.EVEN
;*****
.SBTTL TYPE ROUTINE
;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;*
;*CALL:
;*1) USING A TRAP INSTRUCTION
;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;* TYPE
;* MESADR
;*
;*2) USING A JSR INSTRUCTION
;* MOV PS,-(SP) ;;PUSH PROCESSOR STATUS WORD ON THE STACK
;* JSR PC,$TYPE ;;CALL TYPE ROUTINE
;* MESADDR ;;FIRST ADDRESS OF MESSAGE
$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
BPL IS ;;BR IF YES
HALT ;;HALT HERE IF NO TERMINAL
BR 3$ ;;LEAVE
1$: MOV RO,-(SP) ;;SAVE RO

```



TYPE ROUTINE

```

8209 046456 112046      25:  MOVB  (RO)+, -(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
8210 046460 001005      BNE  45                      ;; BR IF IT ISN'T THE TERMINATOR
8211 046462 005726      TST  (SP)+                    ;; IF TERMINATOR POP IT OFF THE STACK
8212 046464 012600      MOV  (SP)+, RO                ;; RESTORE RO
8213 046466 062716 000002    35:  ADD  #2, (SP)                ;; ADJUST RETURN PC
8214 046472 000002      RTI                          ;; RETURN
8215 046474 122716 000011    45:  CMPB #HT, (SP)              ;; BRANCH IF <HT>
8216 046500 001426      BEQ  85                      ;;
8217 046502 122716 000200    85:  CMPB #CRLF, (SP)            ;; BRANCH IF NOT
8218 046506 001004      BNE  55                      ;;
8219 046510 005726      TST  (SP)+                    ;; POP <CR><LF> EQUIV
8220 046512 104400 001327    55:  TYPE $CRLF                  ;;
8221 046516 000757      BR  25                      ;; GET NEXT CHARACTER
8222 046520 004767 000056    25:  JSR  PC, $TYPEC              ;; GO TYPE THIS CHARACTER
8223 046524 126726 132522    65:  CMPB #FILLC, (SP)+          ;; IS IT TIME FOR FILLER CHARS.?
8224 046530 001352      BNE  25                      ;; IF NO GO GET NEXT CHAR.
8225 046532 016746 132512    25:  MOV  $NULL, -(SP)           ;; GET # OF FILLER CHARS. NEEDED
8226                                     AND THE NULL CHAR.
8227 046536 105366 000001    75:  DECB 1(SP)                  ;; DOES A NULL NEED TO BE TYPED?
8228 046542 002770      BLT  65                      ;; BR IF NO--GO POP THE NULL OFF OF STACK
8229 046544 004767 000032    65:  JSR  PC, $TYPEC              ;; GO TYPE A NULL
8230 046550 105367 000100    DECB $CHARCNT                ;; DON'T COUNT THE NULL AS A CHARACTER
8231 046554 000770      BR  75                      ;; LOOP
8232
8233                                     ;; HORIZONTAL TAB PROCESSOR
8234
8235 046556 112716 000040    85:  MOVB #' (SP)                ;; REPLACE TAB WITH SPACE
8236 046562 004767 000014    95:  JSR  PC, $TYPEC              ;; TYPE A SPACE
8237 046566 132767 000007 000060    BITB #7, $CHARCNT            ;; BRANCH IF NOT AT
8238 046574 001372      BNE  95                      ;; TAB STOP
8239 046576 005726      TST  (SP)+                    ;; POP SPACE OFF STACK
8240 046600 000726      BR  25                      ;; GET NEXT CHARACTER
8241 046602 005737 001466    $TYPEC: TST  $NOTYPE              ;; INHIBIT TYPING?
8242 046606 100423      BMI  $TYPEX                  ;; BRANCH IF YES
8243 046610 105777 132430    TSTB $STPB                    ;; WAIT UNTIL PRINTER IS READY
8244 046614 100372      BPL  $TYPEC                  ;;
8245 046616 116677 000002 132422    MOVB 2(SP), $STPB            ;; LOAD CHAR TO BE TYPED INTO DATA REG.
8246 046624 122766 000015 000002    CMPB #CR, 2(SP)              ;; BRANCH IF
8247 046632 001003      BNE  15                      ;; NOT <CR>
8248 046634 105067 000014    CLRB $CHARCNT                ;;
8249 046640 000406      BR  $TYPEX                  ;; EXIT
8250 046642 122766 000012 000002    15:  CMPB #LF, 2(SP)             ;; BRANCH IF
8251 046650 001402      BEQ  $TYPEX                  ;; <LF>
8252 046652 105227      INCB (PC)+                    ;; INC SPACE
8253 046654 000000      $CHARCNT: .WORD 0            ;; COUNT
8254 046656 000207      $TYPEX: RTS PC
8255
8256                                     ;*****
8257                                     ;SBTTL ROUTINE TO TYPE THE ELAPSED RUN TIME OF THE PROGRAM
8258                                     ;* THIS ROUTINE CONVERTS THE CONTENTS OF LOCATIONS "LTICKS"
8259                                     ;* AND "MTICKS" TO SECONDS AND MINUTES/HOURS RESPECTIVELY
8260                                     ;* AND TYPES THEM IN THE FOLLOWING FORMAT:
8261                                     ;* HHH:MM:SS
8262                                     ;*****
8263 046660 104412      $TIME: SAVREG

```

```

8265 046666 113701 001603
8266 046672 005000
8267 046674 071027 000012
8268 046700 062701 000060
8269 046704 110137 047116
8270 046710 010001
8271 046712 005000
8272 046714 071027 000006
8273 046720 062701 000060
8274 046724 110137 047115
8275 046730 013701 001600
8276 046734 005000
8277 046736 071027 000012
8278 046742 062701 000060
8279 046746 110167 000141
8280 046752 010001
8281 046754 005000
8282 046756 071027 000006
8283 046762 062701 000060
8284 046766 110167 000120
8285 046772 005700
8286 046774 001434
8287 046776 010001
8288 047000 005000
8289 047002 071027 000012
8290 047006 062701 000060
8291 047012 110167 000072
8292 047016 005700
8293 047020 001422
8294 047022 010001
8295 047024 005000
8296 047026 071027 000010
8297 047032 062701 000060
8298 047036 110167 000045
8299 047042 005700
8300 047044 001410
8301 047046 010001
8302 047050 005000
8303 047052 071027 000012
8304 047056 062701 000060
8305 047062 110167 000020
8306 047066 104400 047106
8307 047072 104400 001327
8308 047076 004737 053216
8309 047102 104414
8310 047104 000207
8311 047106 001 001 001
8312 047111 072 001 001
8313 047114 072 060 060
8314 047117 000
8315
8316
8317
8318
8319

```

```

MOV B J#LTICKS+1,R1 ;GET SECOND COUNT
CLR RO
DIV #10,R0
ADD #60,R1
MOV B R1,J#TIMEBUF+10
MOV RO,R1
CLR RO
DIV #6,R0
ADD #60,R1
MOV B R1,J#TIMEBUF+7
MOV J#MTICKS,R1 ;GET MINUTE COUNT
CLR RO
DIV #10,R0 ;GET HOURS AND MINUTES
ADD #60,R1 ;MAKE REMAINDER ASCII
MOV B R1,↑TIMEBUF+5 ;PUT IN BUFFER
MOV RO,R1
CLR RO
DIV #6,R0
ADD #60,R1
MOV B R1,↑TIMEBUF+4
RO
TST RO
BEQ 2$
MOV RO,R1
CLR RO
DIV #10,R0
ADD #60,R1
MOV B R1,↑TIMEBUF+2
RO
TST RO
BEQ 2$
MOV RO,R1
CLR RO
DIV #10,R0
ADD #60,R1
MOV B R1,↑TIMEBUF+1
RO
TST RO
BEQ 2$
MOV RO,R1
CLR RO
DIV #10,R0
ADD #60,R1
MOV B R1,↑TIMEBUF
2$:
TYPE ↑TIMEBUF
TYPE ↑CRLF
JSR PC,J#RESKT ;GO BACK TO ORIGINAL MEMORY
RESREG
RTS PC
TIMEBUF: .BYTE 1,1,1,72,1,1,72,60,60,0

```

```

.EVEN
:*****
:SBTTL ROUTINE TO TYPE THE AVAILABLE DEVICES AND UNIT NUMBERS
:* THIS ROUTINE SEARCHES THE SYSTEM SIZE TABLE FOR NON-
:* ZERO ENTRIES.WHEN IT FINDS ONE, IT TYPES THE NAME OF THE

```

8321  
8322  
8323 017120 104400 056372  
8324 017121 104400 053751  
8325 017130 012700 000010  
8326 017131 005001  
8327 017136 105761 001606  
8328 017142 001004  
8329 017144 062701 000002  
8330 017150 077006  
8331 017152 000207  
8332 017154 010102  
8333 017156 062702 053612  
8334 017162 011267 000002  
8335 017166 104400  
8336 017170 000000  
8337 017172 112767 000060 000034  
8338 017200 116102 001606  
8339 017204 012703 000010  
8340 017210 006002  
8341 017212 103002  
8342 017214 104400 047234  
8343 017220 005267 000010  
8344 017224 077307  
8345 017226 104400 001327  
8346 017232 000744  
8347 017234 000 054 040  
8348 017237 000  
8349  
8350  
8351  
8352  
8353  
8354  
8355  
8356  
8357  
8358  
8359  
8360  
8361  
8362  
8363  
8364  
8365  
8366  
8367  
8368  
8369  
8370  
8371  
8372  
8373  
8374  
8375 047240 017646 000000

```

;* AVAILABLE FOR THAT DEVICE.
;*****
TYPsiz: TYPE ,SWITCH
        TYPE ,MSG4
        MOV #10,R0 ;SET SOB COUNT
        CLR R1
1$: TSTB SYSSIZE(R1) ;DEVICE AVAILABLE?
   BNE 2$ ;BRANCH IF YES
7$: ADD #2,R1 ;INCREMENT INDEX
   SOB RO,1$ ;CONTINUE
   RTS PC ;RETURN
2$: MOV R1,R2 ;GET INDEX
   ADD #MSGINX,R2 ;GET ADR OF MESSAGE ADR
   MOV (R2),3$ ;GET ADDRESS OF MESSAGE
3$: .WORD
   MOVB #60,4$ ;INIT UNIT NO. BUFFER (ASCII)
   MOVB SYSSIZE(R1),R2 ;GET WORD WITH AVAILABLE UNITS
   MOV #10,R3 ;SET SOB COUNT
6$: ROR R2 ;GET UNITS
   BCC 5$ ;BRANCH IF NOT A UNIT
5$: INC 4$
   SOB R3,6$ ;CONTINUE
   TYPE $CRLF
4$: .BYTE 0,54,40,0 ;NUMBER, COMMA, SPACE, TERMINATOR
;*****
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;*OCTAL (ASCII) NUMBER AND TYPE IT.
;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;*CALL:
;* MOV NUM,-(SP) ;:NUMBER TO BE TYPED
;* TYPOS ;:CALL FOR TYPEOUT
;* .BYTE N ;:N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;* .BYTE M ;:M=1 OR 0
; ;:1=TYPE LEADING ZEROS
; ;:0=SUPPRESS LEADING ZEROS
;*
;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;*$TYPOS OR $TYPOC
;*CALL:
;* MOV NUM,-(SP) ;:NUMBER TO BE TYPED
;* TYPON ;:CALL FOR TYPEOUT
;*
;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;*CALL:
;* MOV NUM,-(SP) ;:NUMBER TO BE TYPED
;* TYPOC ;:CALL FOR TYPEOUT
$TYPOS: MOV 2(SP),-(SP) ;:PICKUP THE MODE

```

```

8377 047252 112667 000207          MOVB      (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
8378 047256 062716 000002          ADD       #2,(SP)        ;;ADJUST RETURN ADDRESS
8379 047262 000406                   BR        $TYPON
8380 047264 112767 000001 000171 $TYPON: MOVB      #1,$OFILL          ;;SET THE ZERO FILL SWITCH
8381 047272 112767 000006 000165 $TYPON: MOVB      #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
8382 047300 112767 000005 000154 $TYPON: MOVB      #5,$OCNT          ;;SET THE ITERATION COUNT
8383 047306 010346                   MOV       R3,-(SP)       ;;SAVE R3
8384 047310 010446                   MOV       R4,-(SP)       ;;SAVE R4
8385 047312 010546                   MOV       R5,-(SP)       ;;SAVE R5
8386 047314 116704 000145          MOVB      $OMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
8387 047320 005404                   NEG       R4
8388 047322 062704 000006          ADD       #6,R4         ;;SUBTRACT IT FOR MAX. ALLOWED
8389 047326 110467 000132          MOVB      R4,$OMODE     ;;SAVE IT FOR USE
8390 047332 116704 000125          MOVB      $OFILL,R4     ;;GET THE ZERO FILL SWITCH
8391 047336 016605 000012          MOV       12(SP),R5     ;;PICKUP THE INPUT NUMBER
8392 047342 005003                   CLR       R3            ;;CLEAR THE OUTPUT WORD
8393 047344 006105                   1$:      ROL       R5            ;;ROTATE MSB INTO "C"
8394 047346 000404                   BR        3$           ;;GO DO MSB
8395 047350 006105                   2$:      ROL       R5            ;;FORM THIS DIGIT
8396 047352 006105                   ROL       R5
8397 047354 006105                   ROL       R5
8398 047356 010503                   MOV       R5,R3
8399 047360 006103                   3$:      ROL       R3            ;;GET LSB OF THIS DIGIT
8400 047362 105367 000076          DECB     $OMODE         ;;TYPE THIS DIGIT?
8401 047366 100016                   BPL      7$           ;;BR IF NO
8402 047370 042703 177770          BIC      #177770,R3     ;;GET RID OF JUNK
8403 047374 001002                   BNE      4$           ;;TEST FOR 0
8404 047376 005704                   TST      R4            ;;SUPPRESS THIS 0?
8405 047400 001403                   BEQ      5$           ;;BR IF YES
8406 047402 005204                   4$:      INC       R4            ;;DON'T SUPPRESS ANYMORE 0'S
8407 047404 052703 000060          BIS      #'0,R3        ;;MAKE THIS DIGIT ASCII
8408 047410 052703 000040          5$:      BIS      #' ,R3        ;;MAKE ASCII IF NOT ALREADY
8409 047414 110367 000040          MOVB     R3,$S         ;;SAVE FOR TYPING
8410 047420 104400 047460          TYPE     $S           ;;GO TYPE THIS DIGIT
8411 047424 105367 000032          7$:      DECB     $OCNT        ;;COUNT BY 1
8412 047430 003347                   BGT      2$           ;;BR IF MORE TO DO
8413 047432 002402                   BLT      6$           ;;BR IF DONE
8414 047434 005204                   INC      R4            ;;INSURE LAST DIGIT ISN'T A BLANK
8415 047436 000744                   BR        2$           ;;GO DO THE LAST DIGIT
8416 047440 012605                   6$:      MOV       (SP)+,R5     ;;RESTORE R5
8417 047442 012604                   MOV       (SP)+,R4     ;;RESTORE R4
8418 047444 012603                   MOV       (SP)+,R3     ;;RESTORE R3
8419 047446 016666 000002 000004  MOV       2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
8420 047454 012616                   MOV       (SP)+,(SP)
8421 047456 000002                   RTI
8422 047460 000                   8$:      .BYTE     0           ;;RETURN
8423 047461 000                   .BYTE     0           ;;STORAGE FOR ASCII DIGIT
8424 047462 000                   $OCNT:    .BYTE     0           ;;TERMINATOR FOR TYPE ROUTINE
8425 047463 000                   $OFILL:   .BYTE     0           ;;OCTAL DIGIT COUNTER
8426 047464 000000                   $OMODE:   .WORD     0           ;;ZERO FILL SWITCH
8427                                     ;;NUMBER OF DIGITS TO TYPE
8428                                     ;;*****
8429 .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
8430
8431 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT

```

```

0433      ; *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
0434      ; *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
0435      ; *REPLACED WITH SPACES.
0436      ; *CALL:
0437      ; *      MOV      NUM,-(SP)      ;; PUT THE BINARY NUMBER ON THE STACK
0438      ; *      TYPDS      ;; GO TO THE ROUTINE
0439
0440      $TYPDS:
0441      MOV      R0,-(SP)      ;; PUSH R0 ON STACK
0442      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
0443      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
0444      MOV      R3,-(SP)      ;; PUSH R3 ON STACK
0445      MOV      R5,-(SP)      ;; PUSH R5 ON STACK
0446      MOV      #20200,-(SP)      ;; SET BLANK SWITCH AND SIGN
0447      MOV      20(SP),R5      ;; GET THE INPUT NUMBER
0448      BPL      1$           ;; BR IF INPUT IS POS.
0449      NEG      R5           ;; MAKE THE BINARY NUMBER POS.
0450      MOVVB   #'-,1(SP)     ;; MAKE THE ASCII NUMBER NEG.
0451      CLR      R0           ;; ZERO THE CONSTANTS INDEX
0452      MOV      #SDBLK,R3    ;; SETUP THE OUTPUT POINTER
0453      MOVVB   #' ,(R3)+     ;; SET THE FIRST CHARACTER TO A BLANK
0454      CLR      R2           ;; CLEAR THE BCD NUMBER
0455      MOV      $DTBL(R0),R1  ;; GET THE CONSTANT
0456      SUB     R1,R5         ;; FORM THIS BCD DIGIT
0457      BLT     4$           ;; BR IF DONE
0458      INC     R2           ;; INCREASE THE BCD DIGIT BY 1
0459      BR      3$
0460      ADD     R1,R5         ;; ADD BACK THE CONSTANT
0461      TST     R2           ;; CHECK IF BCD DIGIT=0
0462      BNE     5$           ;; FALL THROUGH IF 0
0463      TSTB   (SP)          ;; STILL DOING LEADING 0'S?
0464      BMI     7$           ;; BR IF YES
0465      ASLB   (SP)          ;; MSD?
0466      BCC     6$           ;; BR IF NO
0467      MOVB   1(SP),-1(R3)   ;; YES--SET THE SIGN
0468      BIS    #'0,R2        ;; MAKE THE BCD DIGIT ASCII
0469      BIS    #' ,R2        ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
0470      MOVB   R2,(R3)+     ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
0471      TST   (R0)+         ;; JUST INCREMENTING
0472      CMP   R0,#10        ;; CHECK THE TABLE INDEX
0473      BLT   2$           ;; GO DO THE NEXT DIGIT
0474      BGT   8$           ;; GO TO EXIT
0475      MOV   R5,R2         ;; GET THE LSD
0476      BR   6$           ;; GO CHANGE TO ASCII
0477      TSTB (SP)+         ;; WAS THE LSD THE FIRST NON-ZERO?
0478      BPL   9$           ;; BR IF NO
0479      MOVB -1(SP),-2(R3)   ;; YES--SET THE SIGN FOR TYPING
0480      CLRB (R3)          ;; SET THE TERMINATOR
0481      MOV   (SP)+,R5      ;; POP STACK INTO R5
0482      MOV   (SP)+,R3      ;; POP STACK INTO R3
0483      MOV   (SP)+,R2      ;; POP STACK INTO R2
0484      MOV   (SP)+,R1      ;; POP STACK INTO R1
0485      MOV   (SP)+,R0      ;; POP STACK INTO R0
0486      TYPE $SDBLK        ;; NOW TYPE THE NUMBER
0487      MOV   2(SP),4(SP)   ;; ADJUST THE STACK

```

8489 047670 000002  
8490 047672 023420  
8491 047674 001750  
8492 047676 000144  
8493 047700 000012  
8494 047702 000004  
8495  
8496  
8497  
8498  
8499  
8500  
8501  
8502  
8503  
8504  
8505  
8506  
8507 047712 104412  
8508 047714 016601 000002  
8509 047720 012705 050047  
8510 047724 012704 000014  
8511 047730 012703 177770  
8512 047734 012100  
8513 047736 012101  
8514 047740 005002  
8515 047742 110245  
8516 047744 010002  
8517 047746 005304  
8518 047750 003016  
8519 047752 001414  
8520 047754 005205  
8521 047756 010566 000002  
8522 047762 122765 000061 000003  
8523 047770 002003  
8524 047772 112765 000060 000003  
8525 050000 104414  
8526 050002 000207  
8527 050004 006203  
8528 050006 006001  
8529 050010 006000  
8530 050012 006001  
8531 050014 006000  
8532 050016 006001  
8533 050020 006000  
8534 050022 040302  
8535 050024 062702 000060  
8536 050030 000744  
8537 050032 000016  
8538  
8539  
8540  
8541  
8542  
8543

```

RTI ;:RETURN TO USER
SDTBL: 10000.
      1000.
      100.
      10.
$DBLK: .BLKW 4
;:*****
.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
;:THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
;:UNSIGNED OCTAL ASCII NUMBER.
;:CALL
;:   MOV      #PNTR, -(SP) ;: POINTER TO LOW WORD OF BINARY NUMBER
;:   JSR      PC, 2#$DB20 ;: CALL THE ROUTINE
;:   RETURN   ;: THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
$DB20: SAVREG ;: SAVE ALL REGISTERS
      MOV     2(SP), R1 ;: PICKUP THE POINTER TO LOW WORD
      MOV     #S$OCTVL+13., R5 ;: POINTER TO DATA TABLE
      MOV     #12., R4 ;: DO ELEVEN CHARACTERS
      MOV     #1C7, R3 ;: MASK
      MOV     (R1)+, R0 ;: LOWER WORD
      MOV     (R1)+, R1 ;: HIGH WORD
      CLR     R2 ;: TERMINATOR
1$:     MOVB   R2, -(R5) ;: PUT CHARACTER IN DATA TABLE
      MOV     R0, R2 ;: GET THIS DIGIT
      DEC     R4 ;: COUNT THIS CHARACTER
      BGT     3$, ;: BR IF NOT THE LAST DIGIT
      BEQ     2$, ;: BR IF IT IS THE LAST DIGIT
      INC     R5 ;: ALL DIGITS DONE-ADJUST POINTER FOR FIRST
      MOV     R5, 2(SP) ;: ASCII CHAR. & PUT IT ON THE STACK
      CMPB   #61, 3(R5) ;: LAST NUMBER LEGAL?
      BGE     4$, ;: BRANCH IF YES
      MOVB   #60, 3(R5) ;: MAKE IT ZERO
4$:     RESREG ;: RESTORE ALL REGISTERS
      RTS    PC ;: RETURN TO USER
2$:     ASR   R3 ;: POSITION THE MASK FOR THE LAST DIGIT
3$:     ROR   R1 ;: POSITION THE BINARY NUMBER FOR
      ROR   R0 ;: THE NEXT OCTAL DIGIT
      ROR   R1
      ROR   R0
      ROR   R1
      ROR   R0
      BIC   R3, R2 ;: MASK OUT ALL JUNK
      ADD   #'0, R2 ;: MAKE THIS CHAR. ASCII
      BR    1$ ;: GO PUT IT IN THE DATA TABLE
$OCTVL: .BLKB 14. ;: RESERVE DATA TABLE
;:*****
.SBTTL SAVE AND RESTORE R0-R5 ROUTINES
;:SAVE R0-R5
;:CALL:

```

8545  
8546  
8547  
8548  
8549  
8550  
8551  
8552  
8553  
8554  
8555  
8556  
8557  
8558  
8559  
8560  
8561  
8562  
8563  
8564  
8565  
8566  
8567  
8568  
8569  
8570  
8571  
8572  
8573  
8574  
8575  
8576  
8577  
8578  
8579  
8580  
8581  
8582  
8583  
8584  
8585  
8586  
8587  
8588  
8589  
8590  
8591  
8592  
8593  
8594  
8595  
8596  
8597  
8598  
8599

050050  
050050 010046  
050052 010146  
050054 010246  
050056 010346  
050060 010446  
050062 010546  
050064 016646 000022  
050070 016646 000022  
050074 016646 000022  
050100 016646 000022  
050104 000002  
  
050106  
050106 012666 000022  
050112 012666 000022  
050116 012666 000022  
050122 012666 000022  
050126 012605  
050130 012604  
050132 012603  
050134 012602  
050136 012601  
050140 012600  
050142 000002

UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:

\*  
\*TOP---(+16)  
\* +2---(+18)  
\* +4---R5  
\* +6---R4  
\* +8---R3  
\*+10---R2  
\*+12---R1  
\*+14---R0

\$SAVREG:  
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  
MOV R4, -(SP) : PUSH R4 ON STACK  
MOV R5, -(SP) : PUSH R5 ON STACK  
MOV 22(SP), -(SP) : SAVE PS OF MAIN FLOW  
MOV 22(SP), -(SP) : SAVE PC OF MAIN FLOW  
MOV 22(SP), -(SP) : SAVE PS OF CALL  
MOV 22(SP), -(SP) : SAVE PC OF CALL  
RTI

\*RESTORE RO-R5  
\*CALL:

RESREG  
\$RESREG:  
MOV (SP)+, 22(SP) : RESTORE PC OF CALL  
MOV (SP)+, 22(SP) : RESTORE PS OF CALL  
MOV (SP)+, 22(SP) : RESTORE PC OF MAIN FLOW  
MOV (SP)+, 22(SP) : RESTORE PS OF MAIN FLOW  
MOV (SP)+, R5 : POP STACK INTO R5  
MOV (SP)+, R4 : POP STACK INTO R4  
MOV (SP)+, R3 : POP STACK INTO R3  
MOV (SP)+, R2 : POP STACK INTO R2  
MOV (SP)+, R1 : POP STACK INTO R1  
MOV (SP)+, R0 : POP STACK INTO R0  
RTI

\*\*\*\*\*

\$BITL CONVERT FLOATING BINARY TO OCTAL ASCIZ  
\*  
\*THIS ROUTINE CONVERTS A 32 BIT FLOATING NUMBER TO AN OCTAL  
\*ASCIZ STRING IN THE FOLLOWING FORMAT:  
\*  
\* W XXX YYY ZZZZZZ  
\*  
\* WHERE W = SIGN BIT  
\* X = 8-BIT EXPONENT (RIGHT JUSTIFIED)  
\* Y = FRACTION BITS (57:51) (RIGHT JUSTIFIED)  
\* Z = FRACTION BITS (50:35)  
\*  
\*IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING  
\*NUMBER IN THE WORD FOLLOWING THE CALL.  
\*IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.

8601	050144	104412		\$FL20:	SAVREG	
8602	050146	017600	000000		MOV	2(SP),R0 ; GET ADDRESS OF DATA
8603	050152	062716	000002		ADD	#2,(SP) ; ADJUST RETURN PC
8604	050156	016001	00C002		MOV	2(R0),R1 ; PUT SECOND DATA WORD IN R1
8605	050162	011000			MOV	(R0),R0 ; PUT FIRST DATA WORD IN R0
8606	050164	012704	001357		MOV	#\$FLBUFF+23,R4 ; GET ADDRESS OF BUFFER END IN R4
8607	050170	112744	000000		MOV	#0,-(R4) ; PUT TERMINATOR IN BUFFER
8608	050174	012705	000005		MOV	#5,R5 ; SET SOB COUNT FOR FRACTION DIGITS
8609	050200	010103		15:	MOV	R1,R3 ; GET LSB'S OF FRACTION
8610	050202	042703	177770		BIC	#1C7,R3 ; SAVE LS 3 BITS
8611	050206	062703	000060		ADD	#60,R3 ; MAKE THEM ASCII
8612	050212	110344			MOV	R3,-(R4) ; STORE IN BUFFER
8613	050214	073027	177775		ASHC	#-3,R0 ; SHIFT NUMBER TO NEXT 3 BITS
8614	050220	077511			SOB	R5,1\$ ; CONTINUE FOR 7 DIGITS
8615	050222	010103			MOV	R1,R3 ; GET NEXT DIGITS
8616	050224	042703	177776		BIC	#1C1,R3 ; ONLY WANT 1 BIT
8617	050230	062703	000060		ADD	#60,R3 ; MAKE THEM ASCII
8618	050234	110344			MOV	R3,-(R4) ; STORE IN BUFFER
8619	050236	112744	000040		MOV	#40,-(R4) ; PUT SPACE IN BUFFER
8620	050242	073027	177777		ASHC	#-1,R0
8621	050246	012705	000002		MOV	#2,R5 ; SET SOB COUNT
8622	050252	010103		35:	MOV	R1,R3 ; GET LOW WORD
8623	050254	042703	177770		BIC	#1C7,R3 ; MASK 3 BITS
8624	050260	062703	000060		ADD	#60,R3 ; MAKE THEM ASCII
8625	050264	110344			MOV	R3,-(R4) ; PUT IN BUFFER
8626	050266	073027	177775		ASHC	#-3,R0 ; GET NEXT 3 BITS
8627	050272	077511			SOB	R5,3\$ ; CONVERT THEM
8628	050274	010103			MOV	R1,R3
8629	050276	042703	177776		BIC	#1C1,R3 ; ONLY WANT 1 BIT
8630	050302	062703	000060		ADD	#60,R3 ; MAKE IT ASCII
8631	050306	110344			MOV	R3,-(R4) ; PUT IN BUFFER
8632	050310	112744	000040		MOV	#40,-(R4) ; PUT SPACE IN BUFFER
8633	050314	112744	000040		MOV	#40,-(R4)
8634	050320	072127	177777		ASHC	#-1,R1
8635	050324	012705	000002		MOV	#2,R5 ; GET FIRST 3 BITS OF EXPONENT
8636	050330	010103		25:	MOV	R1,R3 ; SET SOB COUNT FOR 2 DIGITS
8637	050332	042703	177770		BIC	#1C7,R3 ; GET LSB'S OF EXPONENT
8638	050336	062703	000060		ADD	#60,R3 ; SAVE 3 BITS
8639	050342	110344			MOV	R3,-(R4) ; MAKE THEM ASCII
8640	050344	072127	177775		ASHC	#-3,R1 ; STORE IN BUFFER
8641	050350	077511			SOB	R5,2\$ ; GET NEXT 3 BITS
8642	050352	010103			MOV	R1,R3 ; CONTINUE
8643	050354	042703	177774		BIC	#1C3,R3 ; GET LAST 2 BITS OF EXPONENT
8644	050360	062703	000060		ADD	#60,R3 ; MAKE SURE ONLY 2 BITS
8645	050364	110344			MOV	R3,-(R4) ; MAKE THEM ASCII
8646	050366	112744	000040		MOV	#40,-(R4) ; STORE IN BUFFER
8647	050372	112744	000040		MOV	#40,-(R4) ; PUT SPACE IN BUFFER
8648	050376	042700	177776		BIC	#1C1,R0 ; GET SIGN BIT (IT WAS EXTENDED)
8649	050402	062700	000060		ADD	#60,R0 ; MAKE IT ASCII
8650	050406	110044			MOV	R0,-(R4) ; PUT IT IN THE BUFFER
8651	050410	104414			RESREG	
8652	050412	011646			MOV	(SP),-(SP) ; SAVE RETURN PC
8653	050414	016666	000004 000002		MOV	4(SP),2(SP) ; AND RETURN PSW
8654	050422	012766	001334 000004		MOV	#\$FLBUFF,4(SP) ; PUT BUFFER ADDRESS ON STACK
8655	050430	000006			RTT	; RETURN



B15

```

8657
8658
8659
8660
8661
8662
8663
8664
8665
8666
8667
8668
8669
8670
8671
8672
8673
8674
8675
8676 050432 104412
8677 050434 017667 000000 000006
8678 050442 062716 000002
8679 050446 104416
8680 050450 000000
8681 050452 012600
8682 050454 010067 130720
8683 050460 062700 000041
8684 050464 105040
8685 050466 016701 177756
8686 050472 062701 000004
8687 050476 012102
8688 050500 012103
8689 050502 012701 000002
8690 050506 012704 000005
8691 050512 010305
8692 050514 042705 177770
8693 050520 062705 000060
8694 050524 110540
8695 050526 073227 177775
8696 050532 077411
8697 050534 010305
8698 050536 042705 177776
8699 050542 062705 000060
9700 050546 110540
8701 050550 112740 000040
8702 050554 073227 177777
8703 050560 077126
8704 050562 104414
8705 050564 011646
8706 050566 016666 000004 000002
8707 050574 016766 130600 000004
8708 050602 000006
8709
8710
8711

```

```

;*****
SBTTL CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCIZ
;
;THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
;ASCIZ STRING IN THE FOLLOWING FORMAT:
;
;      U VVV WWW XXXXXX YYYYYY ZZZZZZ
;
;      WHERE  U = SIGN BIT
;             V = 8-BIT EXPONENT (RIGHT JUSTIFIED)
;             W = FRACTION BITS<57:51> (RIGHT JUSTIFIED)
;             X = FRACTION BITS <50:35>
;             Y = FRACTION BITS <34:19>
;             Z = FRACTION BITS <18:03>
;
;IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
;NUMBER IN THE WORD FOLLOWING THE CALL.
;IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
;*****
$FLD20: SAVRFG
MOV      2(SP), 1$      ;GET ADDRESS OF DATA TO CONVERT
ADD      #2, (SP)      ;ADJUST RETURN PC
FL20     ;CONVERT MS 32 BITS
1$:      .WORD
MOV      (SP)+, R0      ;GET ADDRESS OF CONVERTED DATA
MOV      R0, $BUFF     ;SAVE IT
ADD      #4, R0        ;ADJUST TO END OF BUFFER
CLAB     -(R0)         ;PUT TERMINATOR IN BUFFER
MOV      1$, R1        ;GET ADDRESS OF DATA TO CONVERT
ADD      #4, R1        ;ADJUST TO LOWER 32 BITS
MOV      (R1)+, R2     ;SAVE THE DATA
MOV      (R1)+, R3
3$:      MOV      #2, R1 ;SET LOOP COUNT
4$:      MOV      #5, R4 ;SET LOOP COUNT
MOV      R3, R5        ;GET LS 32 BITS OF DATA
BIC      #1<7, R5     ;MASK 3 BITS
ADD      #60, R5      ;MAKE THEM ASCII
MOVB     R5, -(R0)    ;PUT IN BUFFER
ASHC     #-3, R2      ;GET NEXT 3 BITS
SOB     R4, 4$       ;CONTINUE
MOV      R3, R5        ;GET LS 32 BITS
BIC      #1<1, R5     ;ONLY WANT 1 BIT
ADD      #60, R5      ;MAKE IT ASCII
MOVB     R5, -(R0)    ;PUT IN TABLE
MOVB     #40, -(R0)   ;PUT SPACE IN TABLE
ASHC     #-1, R2
SOB     R1, 3$       ;CONVERT NEXT 16 BITS
RESREG
MOV      (SP), -(SP)  ;ADJUST STACK
MOV      4(SP), 2(SP) ;TO RETURN WITH ADDRESS
MOV      $BUFF, 4(SP) ;OF BUFFER ON STACK
RTT      ;RETURN
;*****

```

```

8713
8714
8715
8716
8717
8718
8719
8720
8721
8722 050604
8723 050604 010046
8724 050606 010146
8725 050610 010246
8726 050612 016700 130704
8727 050616 016701 130702
8728 050622 012702 177771
8729 050626 006300
8730 050630 006101
8731 050632 005202
8732 050634 001374
8733 050636 066700 130660
8734 050642 005501
8735 050644 066701 130654
8736 050650 062700 001057
8737 050654 005501
8738 050656 062701 047401
8739 050662 010067 130634
8740 050666 010167 130632
8741 050672 012602
8742 050674 012601
8743 050676 012600
8744 050700 000207
8745
8746
8747
8748
8749
8750
8751
8752
8753
8754 050702 012767 000002 000130
8755 050710 016700 000124
8756 050714 012702 001276
8757 050720 012701 000002
8758 050724 004767 177654
8759 050730 022701 000002
8760 050734 001404
8761 050736 022767 000002 000074
8762 050744 001407
8763 050746 016703 130552
8764 050752 042703 000177
8765 050756 022703 000000
8766 050762 001760
8767 050764 016722 130534

```

```

; *THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
; *WITH A RANGE OF 0 TO 2(+33)-1.
; *CALL:
; *      JSR      PC, $RAND      ; CALL THE ROUTINE
; *      RETURN     ; RETURN HERE THE RANDOM
; *                      ; NUMBER WILL BE IN
; *                      ; $HINUM, $LONUM
; *
$RAND:
      MOV      RO, -(SP)      ; PUSH RO ON STACK
      MOV      R1, -(SP)      ; PUSH R1 ON STACK
      MOV      R2, -(SP)      ; PUSH R2 ON STACK
      MOV      $LONUM, RO      ; SET RO WITH LOW
      MOV      $HINUM, R1      ; SET R1 WITH HIGH
      MOV      #-7, R2        ; SET SHIFT COUNT
1$:   ASL      RO              ; SHIFT RO LEFT AND
      ROL      R1              ; ROTATE CARRY INTO R1 AND
      INC      R2              ; CHECK FOR DONE
      BNE     1$              ; CONTINUE SHIFT LOOP
      ADD     $LONUM, RO      ; ADD NUMBER TO MAKE X 129
      ADC     R1              ; PROPAGATE CARRY
      ADD     $HINUM, R1      ; ADD NUMBER TO MAKE X 129
      ADD     #1057, RO       ; ADD LOW CONSTANT
      ADC     R1              ; PROPAGATE CARRY
      ADD     #47401, R1      ; ADD HIGH CONSTANT
      MOV     RO, $LONUM      ; SAVE RO
      MOV     R1, $HINUM      ; SAVE R1
      MOV     (SP)+, R2       ; POP STACK INTO R2
      MOV     (SP)+, R1       ; POP STACK INTO R1
      MOV     (SP)+, RO       ; POP STACK INTO RO
      RTS     PC              ; RETURN
; *****
; SBTTL  FLOATING POINT NUMBER GENERATOR
; *      THIS ROUTINE GENERATES TWO RANDOM FLOATING POINT NUMBERS
; *      IN EITHER SINGLE OR DOUBLE PRECISION. FOR SINGLE PRECISION
; *      THE NUMBERS ARE STORED IN $TMP0 AND $TMP2. DOUBLE PRECISION
; *      NUMBERS ARE STORED IN $TMP0 AND $TMP4.
; *      IN EITHER SINGLE OR DOUBLE THE EXTENDED EXPONENT IS STORED
; *      IN $REG0 AND $REG1.
; *****
FLTDBL: MOV     #2, $OB0BL      ; SET LOOP FOR 2, FOUR WORD NUMBERS
FLTSGL: MOV     $OB0BL, RO     ; SET WORD LENGTH LOOP
      MOV     $TMP0, R2       ; GET ADDRESS TO STORE WORDS IN
2$:   MOV     #2, R1          ; SET NUMBER OF WORDS TO 2
1$:   JSR     PC, $RAND        ; GET RANDOM NUMBER
      CMP     #2, R1          ; FIRST TIME?
      BEQ     3$              ; BRANCH IF YES
      CMP     #2, $OB0BL      ; DOUBLE PRECISION?
      BEQ     4$              ; BRANCH IF YES
3$:   MOV     $HINUM, R3      ; GET EXPONENT PART
      BIC     #177, R3        ; CHECK FOR MINUS ZERO
      CMP     #BIT15, R3      ;
      BEQ     1$              ; BRANCH IF MINUS ZERO
4$:   MOV     $HINUM, (R2)+    ; SAVE HINUM

```

```

8769 050774 077125          SOB      R1,1$          ;CONTINUE
8770 050776 077030          SOB      R0,2$          ;CONTINUE FOR DOUBLE PREC
8771 051000 012746 001276    MOV      #1$TMP0,-(SP)  ;PUT ADDRESS OF NUMBER ON STACK
8772 051004 012746 001002    MOV      #1002,-(SP)   ;PUT CONTROL WORD ON STACK
8773 051010 022767 000002 000022    CMP      #2,S0B0BL     ;DOUBLE PREC?
8774 051016 001002          BNE      5$           ;BRANCH IF NO
8775 051020 012716 001004    MOV      #1004,(SP)    ;CHANGE CONTROL WORD
8776 051024 004767 000012 000002 5$:      JSR      PC,EXPEXT     ;CALCULATE EXT EXPONENTS
8777 051030 012767 000001    MOV      #1,S0B0BL     ;INIT S0B0BL FOR SINGLE PREC
8778 051036 000207          RTS      PC           ;RETURN
8779 051040 000001          S0BDBL: .WORD 1
8780 *****
8781 :SBTTL  FLOATING POINT EXPONENT EXTENSION
8782 :      THIS ROUTINE CONVERTS THE ACTUAL EXPONENT OF A FLOATING POINT
8783 :      NUMBER INTO AN ACTUAL EXPONENT OF 200 AND AN EXTENDED
8784 :      EXPONENT EQUAL TO THE DIFFERENCE BETWEEN THE ORIGINAL
8785 :      ACTUAL EXPONENT AND 100.
8786 :
8787 :      THE ROUTINE IS ENTERED WITH A CONTROL WORD ON THE STACK.
8788 :      BIT 15 OF THE CONTROL WORD INDICATES WHETHER THE NUMBER
8789 :      IS IN MEMORY (<15>=0) OR IN AN ACCUMULATOR (<15>=1).
8790 :      IF THE NUMBER IS IN AN ACCUMULATOR, BITS <9:8> INDICATE
8791 :      THE ACCUMULATOR NUMBER. IF THE NUMBER(S) IS IN MEMORY,
8792 :      BITS <9:8> INDICATE THE NUMBER OF NUMBERS TO CONVERT AND
8793 :      BITS <2:0> INDICATE THE WORD LENGTH OF THE NUMBER(S).
8794 :      IN THE CASE OF A MEMORY CONVERSION, THE ADDRESS OF THE
8795 :      FIRST WORD TO CONVERT IS ALSO ON THE STACK (PRECEDING
8796 :      THE CONTROL WORD).
8797 :*****
8798 051042 012605          EXPEXT: MOV      (SP)+,R5      ;SAVE RETURN PC
8799 051044 012600          MOV      (SP)+,R0      ;GET CONTROL WORD
8800 051046 100437          BMI      1$           ;BRANCH IF ACC CONVERSION
8801 051050 012601          MOV      (SP)+,R1      ;GET START ADDRESS
8802 051052 162700 000400    SUB      #400,R0
8803 051056 012702 001276    MOV      #1$TMP0,R2    ;GET OFFSET FROM $TMP0
8804 051062 160102          SUB      R1,R2
8805 051064 005402          NEG      R2
8806 051066 006202          ASR      R2
8807 051070 062702 001256    ADD      #1$REG0,R2    ;GEN ADDRESS OF EXT WORD
8808 051074 011103 3$:      MOV      (R1),R3      ;GET DATA
8809 051076 042703 100177    BIC      #100177,R3    ;GET EXPONENT
8810 051102 072327 177771    ASH      #-7,R3      ;RIGHT JUSTIFY EXPONENT
8811 051106 162703 000200    SUB      #200,R3      ;CONVERT TO 2'S COMPLIMENT
8812 051112 010312          MOV      R3,(R2)      ;ADD TO EXTENDED EXPONENT
8813 051114 042711 077E00    BIC      #77600,(R1)  ;MAKE ACTUAL
8814 051120 052711 040000    BIS      #BIT14,(R1)  ;EXPONENT 200
8815 051124 162700 000400    SUB      #400,R0      ;ANY MORE WORDS?
8816 051130 100435          BMI      2$           ;BRANCH IF NO
8817 051132 110003          MOV      R0,R3      ;GET WORD LENGTH
8818 051134 006303          ASL      R3
8819 051136 060301          ADD      R3,R1      ;SELECT NEXT NUMBER ADDRESS
8820 051140 062702 000002    ADD      #2,R2      ;SELECT NEXT EXTENDED ADDRESS
8821 051144 000753          BR      3$           ;CONTINUE
8822 :ACCUMULATOR CONVERSION
8823 051146 072027 177776 1$:      ASH      #-2,R0      ;GET ACCUMULATOR NUMBER

```

```

8825 051156 010002          MOV      R0,R2          ;GENERATE
8826 051160 072227 177773  ASH      #-5,R2        ;ADDRESS OF
8827 051164 062702 001402  ADD      #SAC0,R2      ;EXTENDED EXPONENT
8828 051170 042767 000300 000004  BIC      #300,$$      ;GENERATE INSTRUCTION
8829 051176 050067 000000  BIS      R0,$$        ;TO GET EXPONENT
8830 051202 175003          STEXP   AC0,R3        ;GET EXPONENT
8831 051204 060312          ADD      R3,(R2)     ;ADD TO EXTENDED EXPONENT
8832 051206 005003          CLR      R3
8833 051210 042767 000300 000004  BIC      #300,$$      ;GENERATE INSTRUCTION
8834 051216 050067 000000  BIS      R0,$$        ;TO LOAD EXPONENT BACK TO ACC
8835 051222 176403          LDEXP   R3,AC0       ;LOAD EXPONENT OF 200
8836 051224 010546          MOV      R5,-(SP)    ;RESTORE RETURN PC
8837 051226 000207          RTS      PC          ;RETURN
8838
8839
8840          ;:*****
8841          .SBTTL POWER DOWN AND UP ROUTINES
8842
8843          :POWER DOWN ROUTINE
8843 051230 012737 051356 000024 $PWRDN: MOV      $SILLUP,2#PWRVEC ;:SET FOR FAST UP
8844 051236 012737 000340 000026  MOV      #340,2#PWRVEC+2 ;:PRIO:7
8845 051244 010046          MOV      R0,-(SP)    ;:PUSH R0 ON STACK
8846 051246 010146          MOV      R1,-(SP)    ;:PUSH R1 ON STACK
8847 051250 010246          MOV      R2,-(SP)    ;:PUSH R2 ON STACK
8848 051252 010346          MOV      R3,-(SP)    ;:PUSH R3 ON STACK
8849 051254 010446          MOV      R4,-(SP)    ;:PUSH R4 ON STACK
8850 051256 010546          MOV      R5,-(SP)    ;:PUSH R5 ON STACK
8851 051260 010667 000076          MOV      SP,$SAVR6   ;:SAVE SP
8852 051264 012737 051276 000024  MOV      $PWRUP,2#PWRVEC ;:SET UP VECTOR
8853 051272 000000          HALT
8854 051274 000776          BR      .-2          ;:HANG UP
8855
8856          :POWER UP ROUTINE
8857 051276 016706 000060 $PWRUP: MOV      $SAVR6,SP ;:GET SP
8858 051302 005067 000054          CLR      $SAVR6     ;:WAIT LOOP FOR THE TTY
8859 051306 005267 000050 1$: INC      $SAVR6     ;:WAIT FOR THE INC
8860 051312 001375          BNE     1$          ;:OF WORD
8861 051314 012605          MOV      (SP)+,R5    ;:POP STACK INTO R5
8862 051316 012604          MOV      (SP)+,R4    ;:POP STACK INTO R4
8863 051320 012603          MOV      (SP)+,R3    ;:POP STACK INTO R3
8864 051322 012602          MOV      (SP)+,R2    ;:POP STACK INTO R2
8865 051324 012601          MOV      (SP)+,R1    ;:POP STACK INTO R1
8866 051326 012600          MOV      (SP)+,R0    ;:POP STACK INTO R0
8867 051330 012737 051230 000024  MOV      $PWRDN,2#PWRVEC ;:SET UP THE POWER DOWN VECTOR
8868 051336 012737 000340 000026  MOV      #340,2#PWRVEC+2 ;:PRIO:7
8869 051344 104400          TYPE     $POWER      ;:REPORT THE POWER FAILURE
8870 051346 051364          SPWRMG: .WORD $POWER ;:POWER FAIL MESSAGE POINTER
8871 051350 012716          MOV      (PC)+,(SP) ;:RESTART AT START
8872 051352 003212          SPWRAD: .WORD START ;:RESTART ADDRESS
8873 051354 000002          RTI
8874 051356 000000          $SILLUP: HALT
8875 051360 000776          BR      .-2          ;:THE POWER UP SEQUENCE WAS STARTED
8876 051362 000000          $SAVR6: 0            ;:BEFORE THE POWER DOWN WAS COMPLETE
8877 051364 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER" ;:PUT THE SP HERE
8878 051372 000122
8879          .EVEN

```

8881  
8882  
8883  
8884  
8885  
8886  
8887  
8888  
8889 051374 010046  
8890 051376 016600 000002  
8891 051402 005740  
8892 051404 111000  
8893 051406 016000 051414  
8894 051412 000200  
8895  
8896  
8897  
8898  
8899  
8900  
8901  
8902  
8903  
8904 051414  
8905 051414 046436  
8906 051416 047264  
8907 051420 047240  
8908 051422 047300  
8909 051424 047466  
8910 051426 050050  
8911 051430 050106  
8912 051432 050144  
8913 051434 050432  
8914  
8915  
8916  
8917  
8918  
8919 051436 012703 001662  
8920 051442 005023  
8921 051444 005023  
8922 051446 005023  
8923 051450 005013  
8924  
8925  
8926  
8927 051452 012702 002230  
8928 051456 005072 000010  
8929 051462 012772 043520 000012  
8930 051470 012772 000340 000014  
8931 051476 012732 170000  
8932  
8933 051502 012746 000003  
8934 051506 012746 001666  
8935 051512 004737 052054

.SBTTL TRAP DECODER

;\*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION  
;\*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS  
;\*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL  
;\*GO TO THAT ROUTINE.

```

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

```

.SBTTL TRAP TABLE

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

ROUTINE  
-----

```

$TRPAD: $TYPE    ;; CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
        $TYPOC   ;; CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS   ;; CALL=TYPOS    TRAP+4(104404)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON   ;; CALL=TYPON    TRAP+6(104406)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        $TYPDS   ;; CALL=TYPOS    TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
        $SAVREG  ;; CALL=SAVREG   TRAP+12(104412) SAVE RO-R5 ROUTINE
        $RESREG  ;; CALL=RESREG   TRAP+14(104414) RESTORE RO-R5 ROUTINE
        $FL20    ;; CALL=FL20     TRAP+16(104416)
        $FLD20   ;; CALL=FLD20    TRAP+20(104420)

```

\*\*\*\*\*  
.SBTTL UNIBUS EXERCISER INITIALIZATION ROUTINE  
;\*THIS ROUTINE INITIALIZES THE BASE ADDRESS FOR THE  
;\*UNIBUS EXERCISER AND LOADS UP THE EXERCISER REGISTERS.  
\*\*\*\*\*

```

UBEINIT: MOV    #UBESAV, R3      ;; GET ADDRESS OF NEXT UBE ADRES
        CLR    (R3)+           ;; INITIALIZE
        CLR    (R3)+
        CLR    (R3)+
        CLR    (R3)

```

;; SET UP THE UBE AND START IT

```

MOV    #UBETBL, R2          ;; GET ADDRESS OF UBE TABLE
CLR    @10(R2)             ;; CLEAR ALL ERRORS
MOV    #UBESRV, @12(R2)    ;; SET UP UBE VECTOR
MOV    #PR7, @14(R2)       ;; SET UP UBE VECTOR PSW
MOV    #170000, @1(R2)+    ;; SET CC FOR 2K WORD TRANSFER
;; UBE IS DOING BYTE TRANSFERS
MOV    #3, -(SP)          ;; PUT DEVICE ID IN STACK
MOV    #UBEADR, -(SP)     ;; PUT ADDRESS OF PHYSICAL BA ON STACK
JSR    PC, @#GETMAP       ;; GO GET MAP REGISTER

```

8937 051522 013732 001670  
8938 051526 052737 000040 172516  
8939 051534 000207  
8940  
8941  
8942  
8943  
8944  
8945  
8946  
8947  
8948 051536 104412  
8949 051540 013703 001672  
8950 051544 013702 001674  
8951 051550 042702 177774  
8952 051554 032737 000040 172516  
8953 051562 001005  
8954 051564 010337 001476  
8955 051570 010237 001500  
8956 051574 000421  
8957 051576 010305  
8958 051600 073227 000005  
8959 051604 042702 000003  
8960 051610 062702 170200  
8961 051614 012237 001476  
8962 051620 011237 001500  
8963 051624 042705 160000  
8964 051630 060537 001476  
8965 051634 005537 001500  
8966 051640 104414  
8967 051642 000207  
8968  
8969  
8970  
8971  
8972  
8973  
8974  
8975  
8976  
8977  
8978  
8979  
8980  
8981  
8982  
8983  
8984  
8985  
8986 051644 104412  
8987 051646 013703 001474  
8988 051652 105737 001503  
8989 051656 001426  
8990 051660 005002  
8991 051662 073227 000003

```

MOV    2#UBEADR+2,2(R2)+.LOAD ADR BITS 16 & 17
BIS    #40,2#SR3      ;ENABLE MAP
RTS    PC              ;RETURN
*****
SBTTL  CONVERT UNIBUS VIRTUAL ADDRESS TO PHYSICAL ADDRESS
      * THIS ROUTINE CONVERTS THE CONTENTS OF LOCATIONS
      * "ERRBA" AND "ERRBA+2" FROM A VIRTUAL 18-BIT ADDRESS
      * TO A PHYSICAL 22-BIT ADDRESS AS MAPPED BY THE APPROPRIATE
      * MAP REGISTER. THE 22-BIT ADDRESS IS STORED IN LOCATIONS
      * "PA2116" AND "PA1500".
*****
PHYMAP: SAVREG
MOV    2#ERRBA,R3      ;GET BUS ADDRESS <15:00>
MOV    2#ERRBA+2,R2   ;GET BUS ADDRESS <17:16>
BIC    #177774,R2
BIT    #BITS,2#MMR3   ;MAP ON?
BNE    1$             ;BRANCH IF YES
MOV    R3,2#PA1500    ;PHY ADR=BUS ADR
MOV    R2,2#PA2116
BR     MAPEND
1$:   MOV    R3,R5      ;SAVE ADR BITS <15:00>
      ASHC  #5,R2      ;GET MAP REG SELECT BITS
      BIC  #3,R2
      ADD  #MAPLO,R2   ;FORM ADDRESS OF MAP REG
      MOV  (R2)+,2#PA1500 ;GET CONTENTS OF MAP REG LO
      MOV  (R2),2#PA2116 ;GET CONTENTS OF MAP REG HI
      BIC  #160000,R5  ;FORM PHYSICAL ADDRESS
      ADD  R5,2#PA1500 ;THAT TIMED OUT
      ADC  2#PA2116
MAPEND: RESREG
RTS    PC
*****
SBTTL  CONVERT A VIRTUAL ADDRESS TO A PHYSICAL ADDRESS
      * THIS ROUTINE CONVERTS A 16-BIT VIRTUAL ADDRESS TO A
      * 22-BIT PHYSICAL ADDRESS. THE VIRTUAL ADDRESS IS
      * ASSUMED TO BE IN LOCATION "VADR" AND THE PHYSICAL
      * ADDRESS IS PLACED IN LOCATIONS "PA2115" AND "PA1500".
      *
      * IF MEMORY MANAGEMENT IS OFF THE PHYSICAL ADDRESS IS
      * GENERATED BY SUBTRACTING THE CONTENTS OF LOCATION
      * "FACTOR" FROM THE VIRTUAL ADDRESS. THIS LOCATION
      * CONTAINS THE BYTE OFFSET BETWEEN THE RELOCATED CODE
      * AND THE NON-RELOCATED CODE.
      *
      * IF MEMORY MANAGEMENT IS ON, THE CONTENTS OF THE
      * APPROPRIATE PAR REGISTER IS ADDED(AFTER ADJUSTMENT)
      * TO THE LEAST SIGNIFICANT 13 BITS OF THE VIRTUAL ADDRESS.
*****
CNVADR: SAVREG
MOV    2#VADR,R3      ;GET VIRTUAL ADDRESS TO CONVERT
TSTB  2#MMON         ;IS MEMORY MGMT ON?
BEQ   1$             ;BRANCH IF NO
CLR   R2
      ASHC  #3,R2      ;GET PAR SELECT BITS

```

8993 051672 042703 160000  
 8994 051676 006102  
 8995 051700 062702 172340  
 8996 051704 011205  
 8997 051706 005004  
 8998 051710 073427 000006  
 8999 051714 060305  
 9000 051716 005504  
 9001 051720 010437 001500  
 9002 051724 010537 001476  
 9003 051730 104414  
 9004 051732 000207  
 9005 051734 163703 001506  
 9006 051740 005004  
 9007 051742 010305  
 9008 051744 000765  
 9009  
 9010  
 9011  
 9012  
 9013  
 9014  
 9015  
 9016  
 9017  
 9018  
 9019  
 9020  
 9021

```

BIC #160000,R3 ;MAKE SURE SIGN DIDN'T EXTEND
ROL R2 ;MAKE R2 EVEN FOR WORD ADDRESSING
ADD #KIPARO,R2 ;GET ADDRESS OF PAR
MOV (R2),R5 ;GET PAR DATA
CLR R4 ;SETUP R4
ASHC #6,R4 ;SHIFT PAR DATA
ADD R3,R5 ;FORM PHYSICAL ADDRESS
ADC R4
2$: MOV R4,2#PA2116 ;SAVE PHYSICAL
MOV R5,2#PA1500 ;ADDRESS
RESREG
RTS PC ;RETURN
1$: SUB 2#FACTOR,R3 ;FORM PHYSICAL ADDRESS
CLR R4
MOV R3,R5
BR 2$ ;RETURN

```

9022 051746 012737 052010 000114  
 9023 051754 024042  
 9024 051756 001013  
 9025 051760 005112  
 9026 051762 005112  
 9027 051764 021210  
 9028 051766 001006  
 9029 051770 020005  
 9030 051772 001365  
 9031 051774 012737 052642 000114  
 9032 052002 000207  
 9033 052004 000262  
 9034 052006 000207  
 9035 052010 013737 177744 001302  
 9036 052016 013737 177740 001304  
 9037 052024 013737 177742 001306  
 9038 052032 010237 001474  
 9039 052036 010037 001276  
 9040 052042 104005  
 9041 052044 012737 177777 177744  
 9042 052052 000754  
 9043  
 9044  
 9045  
 9046  
 9047

```

*****
:SBTTL ROUTINE TO CHECK RELOCATED DATA
:ROUTINE TO CHECK DATA RELOCATED
:CALL: R0= HIGHEST ADDRESS +2 OF SOURCE DATA
: R2= HIGHEST ADDRESS +2 OF DEST DATA
: R5= LOWEST ADDRESS OF THE SOURCE DATA
:
: THIS ROUTINE USES A COMPARE INSTRUCTION TO CHECK
: THE DATA THAT WAS RELOCATED. IF A PARITY ERROR OCCURS
: DURING THIS CHECK A SPECIAL ERROR MESSAGE IS TYPED
: INSTEAD OF THE UNEXPECTED TRAP MESSAGE.
*****
CHKDAT: MOV #2$,2#CACHVEC ;SETUP PARITY VECTOR
CMP -(R0),-(R2) ;CHECK DATA
BNE 99$+2
COM (R2) ;COMPLEMENT DEST DATA
COM (R2) ;TWICE
CMP (R2),(R0) ;CHECK DATA
BNE 99$
1$: CMP R0,R5 ;BRANCH IF ALL DATA NOT CHECKED
BNE CHKDAT
MOV #.PARSRV,2#CACHVEC ;RESTORE CACHVEC
RTS PC ;RETURN
99$: SEV
RTS PC
2$: MOV 2#MEMERR,2#STMP2 ;SAVE ERROR REG
MOV 2#LOADRS,2#STMP3 ;SAVE ERROR ADR
MOV 2#HIADRS,2#STMP4
MOV R2,2#VADR
MOV R0,2#STMP0
ERROR 5
MOV #-1,2#MEMERR ;CLEAR ERROR REG
BR 99$ ;RETURN

```

```

*****
:SBTTL ROUTINE TO GET A MAP REGISTER
:THIS ROUTINE TAKES AN 18 BIT RANDOM NUMBER, FINDS TWO
:CONSECUTIVE MAP REGISTERS THAT ARE NOT IN USE, LOADS THE

```

```

9049
9050
9051
9052
9053
9054
9055
9056
9057
9058
9059
9060
9061
9062
9063
9064
9065
9066
9067 052054 016600 000004
9068 052060 016601 000002
9069 052064 013746 177776
9070 052070 005116
9071 052072 042716 177437
9072 052076 000237
9073 052100 104412
9074 052102 012137 001226
9075 052106 012137 001230
9076 052112 004737 050604
9077 052116 013702 001524
9078 052122 013703 001522
9079 052126 073227 177764
9080 052132 042703 177760
9081 052136 022702 000016
9082 052142 100001
9083 052144 000762
9084 052146 005737 001504
9085 052152 001403
9086 052154 122702 000000
9087 052160 001754
9088
9089 052162 010204
9090 052164 042703 100000
9091 052170 073227 177776
9092 052174 042703 000001
9093 052200 010241
9094 052202 010341
9095 052204 012705 000004
9096 052210 120465 001655
9097 052214 001435
9098 052216 077504
9099 052220 110460 001656
9100 052224 072427 000003
9101 052230 062704 170200
9102 052234 042703 160000
9103 052240 013701 001226

```

```

; *AND THE NUMBER + 4K, AND RETURNS A NEW BUS ADDRESS, BASED
; *ON THE RANDOM NUMBER.
; *
; * MAP REGISTERS 0 AND 1 ARE NOT USED IF THE PROGRAM IS
; * RUNNING ON ACT11. THIS ALLOWS "MOTHER" TO ACCESS THE
; * END OF PASS HOOKS.
; *
; * THE MAP TABLE (MAPTBL) CONTAINS 4 BYTES, ONE FOR EACH
; * UNIBUS DEVICE. IF THE UBE IS PRESENT IT USES THE
; * 4TH BYTE. WHEN A REGISTER IS ASSIGNED TO A DEVICE,
; * THE LOWER 4 ADDRESS BITS OF THAT REGISTER ARE PLACED
; * IN THE TABLE. WHEN A DEVICE REQUESTS A REGISTER PAIR
; * THIS TABLE IS THEN SEARCHED TO SEE IF THE REGISTER
; * PAIR IS IN USE.
; * ENTER WITH:
; * 4(SP)=DEVICE ID
; * 2(SP)=ADDRESS OF THE PHYSICAL ADDRESS
; * *****
GETMAP: MOV 4(SP),R0 ;GET DIVICE ID
MOV 2(SP),R1 ;GET ADR OF PHY ADR
MOV 2(PSW),-(SP) ;SAVE CURRENT PRIORITY
COM (SP) ;MAKE IT READY FOR RESTORE
BIC #1CPR7,(SP)
SPL 7 ;IF THIS IS RK CALL, LOCK OUT UBE
SAVREG
MOV (R1)+,2(PSGDAT) ;SAVE PHYSICAL
MOV (R1)+,2(SBDAT) ;ADDRESS
2$: JSR PC,2(SRAND) ;GET RANDOM NUMBER
MOV 2(SHINUM),R2 ;GET HIGH RANDOM NUMBER
MOV 2(SLONUM),R3 ;GET LOW RANDOM NUMBER
ASHC #14,R2 ;CONVERT TO 20 BIT NUMBER
BIC #177760,R2 ;GET RID OF 11 BITS OF SIGN EXT
CMP #16,R2 ;LEGAL MAP REG SELECT?
BPL 3$ ;BRANCH IF YES
BR 2$ ;TRY AGAIN
3$: TST 2(SQV) ;ACT11 (QV OR AUTO)?
BEQ 4$ ;BRANCH IF NO
CMPB #0,R2 ;MAP SELECT 0?
BEQ 2$ ;BRANCH IF YES.(ACT MUST
;USE THIS MAP REG)
4$: MOV R2,R4 ;SAVE MAP SELECT BITS
BIC #BIT15,R3 ;CLEAR SELECT BIT 0
ASHC #2,R2 ;FORM 18 BIT ADDRESS
BIC #BIT0,R3 ;MAKE SURE ITS EVEN
MOV R2,-(R1) ;RETURN NEW BUS ADDRESS
MOV R3,-(R1) ;TO THE APPROPRIATE HANDLER
MOV #4,R5 ;SET SOB COUNT
1$: CMPB R4,MAPTBL-1(R5) ;IS THIS MAP IN USE?
BEQ 5$ ;BRANCH IF YES
SOB R5,1$ ;CONTINUE
MOVSB R4,MAPTBL(R0) ;PUT MAP SELECT BITS IN TABLE
ASH #3,R4 ;FORM INDEX TO GET MAP REG ADDR
ADD #MAPLC,R4 ;GENERATE MAP ADDRESS
BIC #160000,R3 ;GET LS 13 BITS OF RAND NO.
MOV 2(SGDAT),R1 ;GET PHYSICAL

```



```

9105 052250 160301          SUB      R3,R1          ;GENERATE MAP
9106 052252 005602          SBC      R2          ;REGISTER DATA
9107 052254 010124          MOV      R1,(R4)+     ;LOAD THE
9108 052256 010224          MOV      R2,(R4)+     ;FIRST MAP REGISTER
9109 052260 062701 020000  ADD      #20000,R1    ;ADD 4K
9110 052264 005502          ADC      R2          ;TO MAP DATA
9111 052266 010124          MOV      R1,(R4)+     ;LOAD THE
9112 052270 010224          MOV      R2,(R4)+     ;SECOND MAP REGISTER
9113 052272 104414          RESREG
9114 052274 042637 177776  BIC      (SP)+,2#PSW  ;RETURN PRIORITY TO ORIGINAL VALUE
9115 052300 011666 000004  MOV      (SP),4(SP)  ;SETUP RETURN PC
9116 052304 022626          CMP      (SP)+,(SP)+ ;CLEAN UP THE STACK
9117 052306 000207          RTS      PC          ;RETURN
9118
9119 052310 062701 000004  ;REGISTER PAIR IS IN USE, TRY ANOTHER RANDOM NUMBER
9120 052314 000137 052112  SS:     ADD      #4,R1          ;RESTORE R1
          JMP      #2#S          ;GET ANOTHER RANDOM NUMBER
9121
9122
9123
9124
9125
9126 052320 010046          GIVEMAP:MOV RO,-(SP)  ;SAVE RO
9127 052322 016600 000004  MOV      4(SP),RO    ;GET DEVICE ID
9128 052326 112760 000377 001656  MOVB    #377,MAPTBL(RO) ;TAKE IT OUT OF THE TABLE
9129 052334 012600          MOV      (SP)+,RO    ;RESTORE RO
9130 052336 000207          RTS      PC          ;RETURN
9131
9132
9133
9134
9135 052340 013746 177776  ;*****
9136 052344 011627          ;SBTTL ROUTINE TO CLEAR 'T' BIT
9137 052346 000000          ;*****
9138 052350 042716 000020  CLRBIT:MOV 2#PSW,-(SP) ;PUSH PSW ONTO STACK
          MOV      (SP),(PC)+ ;SAVE IN RETPSW BELOW
9139
9140
9141
9142 052354 012746 052362  RETPSW: .WORD 0
          BIC      #20,(SP) ;CLEAR T BIT IN PSW ON STACK
9143 052360 000002          ;*****
9144 052362 000207          ;SBTTL ROUTINE TO RESTORE THE T BIT
          RESPSW:MOV #1$,-(SP) ;SET RETURN PC FOR RTI
          RTI          ;CLEAR 'T' BIT IN PSW
9145
9146 052364 042737 177400 177776  1$:     RTS      PC          ;RETURN
9147 052372 016746 177750  RESTPS: BIC      #177400,2#PSW ;SET KERNEL MODE
          MOV      RETPSW,-(SP) ;PUSH ORIG PSW ONTO STACK
9148 052376 000766          BR       RESPSW
9149
9150
9151
9152
9153
9154
9155
9156
9157
9158
9159
;*****
;SBTTL KEYBOARD INT SERV ROUTINE
;THIS ROUTINE HANDLES INTERRUPTS FROM THE KEYBOARD
;
;TYPING A CONTROL 'C' WILL CAUSE THE PROCESSOR TO HALT
;
;TYPING A CARRAGE RETURN WILL CAUSE A CARRIAGE RETURN-LINE FEED
;TO BE TYPED.
;
;TYPING A CONTROL 'O' WILL INHIBIT ANY FURTHER TYPEOUT. THE SECOND CONTROL 'O'

```

```

9161
9162
9163
9164
9165      000003
9166      000017
9167
9168      052400 017746 126636
9169      052404 042716 177600
9170      052410 022716 000003
9171      052414 001010
9172      052416 012737 001326 001270
9173      052424 106277 126614
9174      052430 005726
9175      052432 000000
9176      052434 000002
9177
9178      052436 122716 000015
9179      052442 001007
9180      052444 012737 001326 001270
9181      052452 106277 126566
9182      052456 005726
9183      052460 000002
9184
9185      052462 122716 000017
9186      052466 001012
9187      052470 005726
9188      052472 005167 126770
9189      052476 100405
9190      052500 012737 001326 001270
9191      052506 106277 126532
9192      052512 000002
9193
9194      052514 104412
9195      052516 011605
9196      052520 004737 053130
9197      052524 013700 001442
9198      052530 110520
9199      052532 105010
9200      052534 022700 001464
9201      052540 001002
9202      052542 012700 001444
9203      052546 010037 001442
9204      052552 004737 053216
9205      052556 104414
9206      052560 005737 001466
9207      052564 100004
9208      052566 005726
9209      052570 105077 126450
9210      052574 000002
9211      052576 105777 126442
9212      052602 100375
9213      052604 112677 126436
9214      052610 000002
9215

```

```

: *
: *ANY OTHER CHARACTER WILL JUST BE ECHOED.
: *****

```

```

          CNTRLC=3
          CNTRLO=17
TKISR:  MOV    @STKB, -(SP)          ;GET CHARACTER
        BIC    @177600, (SP)        ;STRIP UNUSED BITS
        CMP    @CNTRLC, (SP)        ;BRANCH IF NOT CONTROL C (↑C)
        BNE    1$
        MOV    @SCRLF-1, @SREGS     ;ECHO CR LF
        ASRB   @STPS
        TST    (SP)+                ;POP CHARACTER OFF THE STACK
        HALT
        RTI                          ;RETURN
1$:     CMPB   @15, (SP)             ;BRANCH IF NOT <CR>
        BNE    2$
        MOV    @SCRLF-1, @SREGS     ;ECHO CR LF
        ASRB   @STPS
        TST    (SP)+                ;POP CHARACTER OFF STACK
        RTI                          ;RETURN
2$:     CMPB   @CNTRLO, (SP)        ;BRANCH IF NOT CONTROL 0 (↑0)
        BNE    3$
        TST    (SP)+
        COM    NOTYPE
        BMI    7$
        MOV    @SCRLF-1, @SREGS     ;ECHO CR LF
        ASRB   @STPS
        RTI
7$:
3$:     SAVREG
        MOV    (SP), R5              ;RETRIEVE CHARACTER
        JSR   PC, @LDKT              ;GO TO LOW CORE
        MOV    @TKBFRP, R0           ;GET BUFFER PTR
        MOVB  R5, (R0)+              ;LOAD CHAR INTO BFR
        CLRB  (R0)                  ;CLEAR NEXT LOC
        CMP   @TKBFR+20, R0          ;BRANCH IF NOT END OF BFR
        BNE   6$
        MOV   @TKBFR, R0             ;RESET BUFFER PTR
        MOV   R0, @TKBFRP            ;RESTORE BFR PTR
        JSR   PC, @RESKT            ;GO BACK TO ORIGINAL MEMORY
ECHO:   TST   @NOTYPE                ;TYPEOUT DISABLED?
        BPL   1$                    ;BRANCH IF NO
        TST   (SP)+                 ;FIX UP STACK
        CLRB  @STPS                 ;CLEAR IE BIT
        RTI                          ;RETURN
1$:     TSTB  @STPS                  ;PRINTER READY?
        BPL   -4                    ;BRANCH IF NO
        MOVB (SP)+, @STPB           ;MOVE CHAR TO PRINTER
        RTI                          ;RETURN

```

```

9217      ;*****
9218      ;SBTTL TELETYPE INTERRUPT SERVICE ROUTINE
9219      ;*THIS ROUTINE TYPES A MESSAGE POINTED TO BY THE ADR STORED
9220      ;*IN LOCATION $REGS. THIS ROUTINE IS INTERRUPT DRIVEN.
9221      ;*****
9222      052612 005237 001270  TPISR: INC      J#$REGS      ;STEP MESSAGE ADDRESS PTR
9223      052616 117746 126446  MOVB     J#$REGS, -(SP)  ;GET CHAR TO BE TYPED
9224      052622 001356          BNE      ECHO           ;GO TYPE CHAR IF NOT '0'
9225      052624 005726          TST      (SP)+         ;POP STACK
9226      052626 005077 126412  CLR      J$STPS        ;CLEAR IE BIT
9227      052632 012737 001526 001270  MOV      #NULLS, J#$REGS
9228      052640 000002          RTI              ;RETURN
9229
9230      ;*****
9231      ;SBTTL PARITY ERROR SERVICE
9232      ;THIS ROUTINE FIELDS UNEYPECTED TRAPS TO 114. IT IS ASSUMED
9233      ;* THAT THE ERROR WAS IN CACHE AND WAS CAUSED BY THE "OTHER
9234      ;* WORD" RATHER THAN THE "WANTED WORD" WHICH MEANS THAT THE
9235      ;* BAD DATA IS STILL IN THE CACHE. SO, TO CLEAR THE BAD DATA
9236      ;* THE ERROR ADDRESS IS REFERENCED CAUSING THE CACHE TO GO
9237      ;* TO MAIN MEMORY TO GET THE DATA. THIS PREVENTS AN
9238      ;* ARBITRARY REFERENCE TO THE BAD WORD FROM TRAPPING.
9239
9240      ;*
9241      ;* AFTER THE ERROR IS REPORTED, BITS 2 AND 3 OF THE MEMORY
9242      ;* ERROR REGISTER ARE TESTED TO SEE IF THE BAD DATA IS IN
9243      ;* MAIN MEMORY. IF IT IS, THE PROGRAM RESTARTS SINCE THE
9244      ;* GOOD DATA IS NOW LOST FOREVER. OTHERWISE THE PROGRAM
9245      ;* RETURNS TO THE ADDRESS POINTED TO BY "$LPERR".
9246      ;*****
9246      052642 012737 053122 000114  PARSRV: MOV      #RT1, J#CACHVEC ;PUT NEW ADDRESS IN PARITY VECTOR
9247      052650 016637 000002 001276  MOV      2(SP), J$STMP0 ;SAVER ERROR PSW
9248      052656 011637 001474          MOV      (SP), J$VAOR    ;SAVE PC
9249      052662 162737 000002 001474  SUB      #2, J$VAOR     ;ADJUST ERROR PC
9250      052670 013702 177744          MOV      J#MEMERR, R2   ;GET ERROR REGISTER
9251      052674 013703 177740          MOV      J#LOADRS, R3  ;GET LO ADDRESS ERROR REG
9252      052700 010337 001304          MOV      R3, J$STMP3   ;PUT LOW ADR IN MEMORY
9253      052704 013737 177742 001306  MOV      J#HIADRS, J$STMP4 ;GET HI ADDRESS ERROR REG
9254      052712 042703 176000          BIC      #176000, R3    ;MASK OFF LOWER TEN BITS
9255      052716 013704 172354          MOV      J#KIPAR6, R4  ;SAVE PAR6
9256      052722 105737 001503          TSTB    J#MMON        ;IS MEMORY MGMT ON?
9257      052726 001407          BEQ     IS           ;BRANCH IF NO
9258      052730 005037 172354          CLR     J#KIPAR6     ;CLEAR PAR6
9259      052734 012737 077406 172314  MOV      #77406, J#KIPDR6 ;ENSURE PDR 6 RESIDENT
9260      052742 052703 140000          BIS      #140000, R3   ;SETUP R3 TO REFERENCE THRU PAR6
9261      052746 105713          1$:    TSTB    (R3)     ;REFERENCE ADDRESS THAT TRAPPED
9262
9263      052750 005102          2$:    COM     R2      ;SHOULD CAUSE ABORT
9264      052752 010237 177744          MOV     R2, J#MEMERR  ;GET ORIGINAL MEMORY
9265      052756 013737 177744 001302  PERET: MOV     J#MEMERR, J$STMP2 ;ERROR REG DATA
9266      052764 013737 001212 001266  MOV     J#SLPERR, J#$REG4 ;SAVE ERROR REG FOR TYPEOUT
9267      052772 012737 053002 001212  MOV     #2$, J$SLPERR  ;SAVE LOOP ADDRESS
9268      053000 104004          MOV     #2$, J$SLPERR  ;SET RETURN ADDRESS IF LOOPING
9269      053002 013737 001266 001212  2$:    MOV     J#$REG4, J$SLPERR ;RESTORE LOOP ADDRESS
9270      053010 010437 172354          MOV     R4, J#KIPAR6  ;RESTORE PAR6
9271      053014 013704 177744          MOV     J#MEMERR, R4  ;GET MEM ERR REG

```

9273 053026 012737 052642 000114  
 9274 053034 042704 177763  
 9275 053040 001426  
 9276 053042 104400 053050  
 9277 053046 000420  
 9278  
 9279 053110  
 9280 053110 000005  
 9281 053112 000137 003212  
 9282 053116 012716 053124  
 9283  
 9284 053122 000002  
 9285 053124 000177 126062  
 9286  
 9287  
 9288  
 9289  
 9290  
 9291  
 9292 053130 105737 001503  
 9293 053134 001427  
 9294 053136 012604  
 9295 053140 013737 177776 053272  
 9296 053146 042737 140000 177776  
 9297 053154 012700 172340  
 9298 053160 012001  
 9299 053162 012002  
 9300 053164 012003  
 9301 053166 012740 000400  
 9302 053172 012740 000200  
 9303 053176 005040  
 9304 053200 012700 053264  
 9305 053204 010120  
 9306 053206 010220  
 9307 053210 010320  
 9308 053212 010446  
 9309 053214 000207  
 9310  
 9311  
 9312  
 9313  
 9314  
 9315 053216 105737 001503  
 9316 053222 001417  
 9317 053224 012604  
 9318 053226 012700 053264  
 9319 053232 012001  
 9320 053234 012002  
 9321 053236 012003  
 9322 053240 012700 172340  
 9323 053244 010120  
 9324 053246 010220  
 9325 053250 010310  
 9326 053252 013737 053272 177776  
 9327 053260 010446

```

MOV      #.PARSRV, @#CACHVEC      ;RESTORE PARITY VECTOR
BIC      #177763, R4              ;CLEAR ALL BUT BITS 2 & 3
BEQ      1$                       ;BRANCH IF NOT MAIN MEMORY ERROR
TYPE     65$                      ;TYPE ASCIZ STRING
BR       64$                      ;GET OVER THE ASCIZ
;:65$: .ASCIZ /FATAL PARITY ERROR-RESTARTING/<CALF>
64$:
RESET
JMP      @#START                  ;CLEAR THE WORLD
1$:     MOV      @X, (SP)          ;PUT ADDRESS ON STACK TO GET ORIGINAL
;PSW BACK
RT1:    RTI
X:      JMP      @#SLPERR         ;JUMP TO START OF TEST THAT HAD THE PE
;*****
;SBTTL CONTEXT SWITCH DOWN SUBROUTINE
;*
;* SUBROUTINE TO SAVE & LOAD KIPAR'S 0, 1, AND 2 (IF MEM MGMT ENABLED)
;* THIS ROUTINE IS CALLED BY THE KEYBOARD INTERRUPT, LINE CLOCK
;* INTERRUPT, USE SERVICE ROUTINE, MBT SERVICE ROUTINE, AND TYPE TIME ROUTINE.
;*****
LOKT:   TSTB   @#MMON             ;BRANCH IF MEM MGMT DISABLED
BEQ     1$
MOV     (SP)+, R4                ;SAVE RETURN PC
MOV     @#PSW, @#SSAVPSW        ;SAVE THE CURRENT PSW
BIC     #140000, @#PSW          ;GO TO KERNEL MODE
MOV     @#KIPAR0, R0            ;GET ADDRESS OF PAR0
MOV     (R0)+, R1               ;GET PAR0
MOV     (R0)+, R2               ;GET PAR1
MOV     (R0)+, R3               ;GET PAR2
MOV     #400, -(R0)             ;RELOC BACK TO LOW CORE
MOV     #200, -(R0)
CLR     -(R0)
MOV     @#SSAVPAR, R0           ;GET ADDRESS OF SAVE BUFFER
MOV     R1, (R0)+               ;PUT PAR DATA IN MEMORY
MOV     R2, (R0)+
MOV     R3, (R0)+
MOV     R4, -(SP)              ;PUT RETURN PC ON STACK
1$:     RTS      PC

;*****
;SBTTL CONTEXT SWITCH UP SUBROUTINE
;SUBROUTINE TO RESTORE KIPAR0, 1, AND 2 (IF MGMT ENABLED)
;*****
RESKT:  TSTB   @#MMON             ;BRANCH IF MEM MGMT DISABLED
BEQ     1$
MOV     (SP)+, R4                ;GET RETURN PC
MOV     @#SSAVPAR, R0           ;GET ADDRESS OF SAVE BUFF
MOV     (R0)+, R1               ;GET OLD PAR DATA
MOV     (R0)+, R2
MOV     (R0)+, R3
MOV     @#KIPAR0, R0           ;GET ADDRESS OF PAR0
MOV     R1, (R0)+               ;RELOCATE BACK
MOV     R2, (R0)+
MOV     R3, (R0)
MOV     @#SSAVPSW, @#PSW
MOV     R4, -(SP)
  
```

```

9329 053264 000003          $$SAVPAR:.BLKW 3
9330 053272 000000          $$SAVPSW:.WORD
9331                                     :*****
9332                                     :SBTTL  KT ABORT SUBROUTINE
9333                                     :*****
9334 053274 016637 000002 001276  KTABRT: MOV      2(SP),J$STMP0          ;SAVE ERROR PSW
9335 053302 011637 001474          MOV      (SP),J$VAOR          ;SAVE ERROR PC
9336 053306 162737 000002 001474          SUB      #2,J$VAOR
9337 053314 013737 177572 001302          MOV      J$MMR0,J$STMP2          ;SAVE MMR0
9338 053322 013737 177576 001304          MOV      J$MMR2,J$STMP3          ;SAVE MMR2
9339 053330 013737 001212 001266          MOV      J$SLPERR,J$SREG4        ;SAVE LOOP ADDRESS
9340 053336 012737 053346 001212          MOV      #15,J$SLPERR          ;SET RETURN ADR IF LOOPING
9341 053344 104003          ERROR      3
9342 053346 013737 001266 001212 15:      MOV      J$SREG4,J$SLPERR        ;RESTORE LOOP ADR
9343 053354 042737 170000 177572          BIC      #170000,J$MMR0          ;CLEAR ERRORS
9344 053362 013716 001212          MOV      J$SLPERR,(SP)          ;GET LOOP ADDRESS
9345 053366 000002          RTI
9346
9347                                     :*****
9348                                     :SBTTL  RESERVED INSTRUCTION ROUTINE
9349                                     :*****
9350 053370 016637 000002 001276  RESERR: MOV      2(SP),J$STMP0          ;SAVE PSW
9351 053376 011637 001474          MOV      (SP),J$VAOR          ;SAVE ERROR PC
9352 053402 162737 000002 001474          SUB      #2,J$VAOR
9353 053410 013737 001212 001266          MOV      J$SLPERR,J$SREG4        ;SAVE LOOP ADR
9354 053416 012737 053426 001212          MOV      #15,J$SLPERR          ;SET RETURN ADR IF LOOPING
9355 053424 104002          ERROR      2
9356 053426 013737 001266 001212 15:      MOV      J$SREG4,J$SLPERR        ;RESTORE LOOP ADR
9357 053434 013716 001212          MOV      J$SLPERR,(SP)          ;GET LOOP ADDRESS
9358 053440 000002          RTI
9359
9360                                     :*****
9361                                     :SBTTL  TRAP TO 4 SERVICE ROUTINE
9362                                     :*****
9363 053442 016637 000002 001276  ERPRT: MOV      2(SP),J$STMP0          ;SAVE ERROR PSW
9364 053450 011637 001474          MOV      (SP),J$VAOR          ;SAVE ERROR PC
9365 053454 162737 000002 001474          SUB      #2,J$VAOR
9366 053462 012706 000700          MOV      #5UPSTK,SP          ;RESTORE SP
9367 053466 013737 177766 001302          MOV      J$CPUERR,J$STMP2        ;GET ERROR REG
9368 053474 013737 001212 001266          MOV      J$SLPERR,J$SREG4        ;SAVE LOOP ADR
9369 053502 012737 053512 001212          MOV      #15,J$SLPERR          ;SET RETURN ADR IF LOOPING
9370 053510 104001          ERROR      1
9371 053512 013737 001266 001212 15:      MOV      J$SREG4,J$SLPERR        ;SET LOOP ADR
9372 053520 005037 177766          CLR      J$CPUERR
9373 053524 013746 001276          MOV      J$STMP0,-(SP)          ;SETUP STACK TO RETURN
9374 053530 013746 001212          MOV      J$SLPERR,-(SP)
9375 053534 000002          RTI

```

```

9377
9378
9379
9380
9381
9382
9383 053536 000000
9384 053540 000020
9385 053542 140000
9386 053544 144020
9387 053546 040000
9388 053550 044020
9389
9390
9391 053552 000000
9392 053554 000004
9393 053556 000006
9394 053560 000010
9395 053562 000000
9396 053564 000012
9397
9398
9399 053566 002100
9400 053570 002122
9401 053572 000000
9402 053574 000000
9403 053576 002142
9404 053600 002202
9405 053602 000000
9406 053604 000000
9407 053606 002246
9408 053610 002230
9409 053612 054066
9410 053614 054074
9411 053616 054703
9412 053620 054703
9413 053622 054102
9414 053624 054110
9415 053626 054703
9416 053630 054703
9417 053632 054377
9418 053634 054742
9419 053636 046200 053517 046040
9420 053644 046511 000077
9421 053650 044510 044107 046040
9422 053656 046511 000077
9423 053662 051105 047522 050122
9424 053670 020103 044120 051531
9425 053676 020103 041520 020040
9426 053704 020040 051520 020127
9427 053712 020040 040515 047111
9428 053720 020124 020040 042524
9429 053726 052123 047040 020117
9430 053734 052523 026502 040520
9431 053742 051523 041440 052116

```

```

: THE BELOW TABLE REPRESENTS THE 'NEW' PSW SET BY THE PROGRAM ON
: SUCCESSIVE SUB-PASSES.
: NOTE THE BELOW TABLE MAY BE MODIFIED TO CAUSE THE PROGRAM TO RUN
: UNDER USER DEFINED PARAMETERS BY PATCHING IN THE DESIRED PASS PARAMETER
: FOR EXAMPLE TO CAUSE THE PROGRAM TO RUN WITHOUT SETTING THE 'T' BIT
: IN ALL PASSES PATCH OUT THE 'T' BIT IN THE TABLE.
PSWTAB: 000000
          000020
          140000
          144020
          040000
          044020
          : T-BIT TRAPPING
          : USER MODE
          : USER MODE, REG SET #1, T-BIT TRAPPING
          : SUPERVISOR MODE
          : SUPERVISOR MODE, REG SET #1, T-BIT TRAPPING

```

```

: THE BELOW TABLE IS USED TO SET MEMORY MARGINS
MRGTAB: .WORD 0 : NO MARGINS
         .WORD 4 : EARLY STROBE
         .WORD 6 : LATE STROBE
         .WORD 10 : LOW DRIVE CURRENT
         .WORD 8 : NO MARGINS
         .WORD 12 : HIGH DRIVE CURRENT

```

```

; MESSAGES
REGINX: .EVEN
         RP30S
         RKDS
         .WORD
         .WORD
         RP4CS!
         RSCSI
         .WORD
         .WORD
         MBTTBL
         UBETBL
MSGINX: .WORD MSG5
         .WORD MSG6
         .WORD MSG21
         .WORD MSG21
         .WORD MSG10
         .WORD MSG11
         .WORD MSG21
         .WORD MSG21
         .WORD MSG15
         .WORD MSG24
MSG1:   .ASCIZ <CRLF>'LOW LIM?'
MSG2:   .ASCIZ 'HIGH LIM?'
MSG3:   .ASCIZ /ERRORPC PHYSC PC PSW MAINT TEST NO SUB-PASS CNT.

```

9433 053751 124 0427510 043040  
 9434 053756 117 0427514 044527  
 9435 053764 116 0427516 053105  
 9436 053772 111 0427511 040440  
 9437 054008 116 0427516 044527  
 9438 054014 114 0427514 020105  
 9439 054020 114 0427514 043040  
 9440 054026 114 0427514 046105  
 9441 054032 114 0427514 047511  
 9442 054038 116 0427516 041511  
 9443 054044 104 0427510 053111  
 9444 054050 104 0427510 041511  
 9445 054056 104 0427510 053111  
 9446 054062 104 0427510 041511  
 9447 054068 104 0427510 053111  
 9448 054074 104 0427510 041511  
 9449 054080 104 0427510 053111  
 9450 054110 104 0427510 041511  
 9451 054116 104 0427510 053111  
 9452 054122 104 0427510 041511  
 9453 054128 104 0427510 053111  
 9454 054134 104 0427510 041511  
 9455 054140 104 0427510 053111  
 9456 054146 104 0427510 041511  
 9457 054152 104 0427510 053111  
 9458 054158 104 0427510 041511  
 9459 054164 104 0427510 053111  
 9460 054170 104 0427510 041511  
 9461 054176 104 0427510 053111  
 9462 054182 104 0427510 041511  
 9463 054188 104 0427510 053111  
 9464 054194 104 0427510 041511  
 9465 054200 104 0427510 053111  
 9466 054206 104 0427510 041511  
 9467 054212 104 0427510 053111  
 9468 054218 104 0427510 041511  
 9469 054224 104 0427510 053111  
 9470 054230 104 0427510 041511  
 9471 054236 104 0427510 053111  
 9472 054242 104 0427510 041511  
 9473 054248 104 0427510 053111  
 9474 054254 104 0427510 041511  
 9475 054260 104 0427510 053111  
 9476 054266 104 0427510 041511  
 9477 054272 104 0427510 053111  
 9478 054278 104 0427510 041511  
 9479 054284 104 0427510 053111  
 9480 054290 104 0427510 041511  
 9481 054296 104 0427510 053111  
 9482 054302 104 0427510 041511  
 9483 054308 104 0427510 053111  
 9484 054314 104 0427510 041511  
 9485 054320 104 0427510 053111  
 9486 054326 104 0427510 041511  
 9487 054332 104 0427510 053111

MSG4: .ASCII /THE FOLLOWING DEVICES AND DRIVES WILL BE USED FOR RELOCATION://<CRLF>

.ASCIZ /DEVICE DRIVES/<CRLF>

MSG5: .ASCIZ ?RPO3 ?  
 MSG6: .ASCIZ ?RK05 ?  
 MSG10: .ASCIZ ?RPO4 ?  
 MSG11: .ASCIZ ?RS04 ?  
 MSG12: .ASCIZ /DRVSTA ERRREG CSREG WRDCNT BUSADR DSKADR CYLADR(RPO3) PHYS BUSA

MSG13: .ASCIZ / CS1 WRDCNT BUSADR BADREX DSKADR CS2 CS3 DRVSTA ERRREG/

MSG14: .ASCIZ /DESCYL ER2 ER3 RPCC/<CRLF>

MSG15: .ASCIZ /MASS BUS TESTER /

MSG16: .ASCIZ / CS1 WRDCNT BUSADR BADREX MR2 CS2 ST ER CS3 \

```

9549 055066 001474 001474 001276
9550 055063 000000 000000 000000
9551 055063 000000 000000 000000
9552 055066 001474 001474 001276
9553 055074 001302 000000 000000
9554 055066 001474 001474 001276
9555 055063 000000 000000 000000
9556 055063 000000 000000 000000
9557 055066 001474 001474 001276
9558 055063 000000 000000 000000
9559 055063 000000 000000 000000
9560 055066 001474 001474 001276
9561 055063 000000 000000 000000
9562 055063 000000 000000 000000
9563 055066 001474 001474 001276
9564 055063 000000 000000 000000
9565 055063 000000 000000 000000
9566 055066 001474 001474 001276
9567 055063 000000 000000 000000
9568 055063 000000 000000 000000
9569 055066 001474 001474 001276
9570 055063 000000 000000 000000
9571 055063 000000 000000 000000
9572 055066 001474 001474 001276
9573 055063 000000 000000 000000
9574 055063 000000 000000 000000
9575 055066 001474 001474 001276
9576 055063 000000 000000 000000
9577 055063 000000 000000 000000
9578 055066 001474 001474 001276
9579 055063 000000 000000 000000
9580 055063 000000 000000 000000
9581 055066 001474 001474 001276
9582 055063 000000 000000 000000
9583 055063 000000 000000 000000
9584 055066 001474 001474 001276
9585 055063 000000 000000 000000
9586 055063 000000 000000 000000
9587 055066 001474 001474 001276
9588 055063 000000 000000 000000
9589 055063 000000 000000 000000
9590 055066 001474 001474 001276
9591 055063 000000 000000 000000
9592 055063 000000 000000 000000
9593 055066 001474 001474 001276
9594 055063 000000 000000 000000
9595 055063 000000 000000 000000
9596 055066 001474 001474 001276
9597 055063 000000 000000 000000
9598 055063 000000 000000 000000
9599 055066 001474 001474 001276
9600 055063 000000 000000 000000

```

MSG17: .ASCIZ / CC BUSADR CR2 CR1 PHYS BUSADR/<CRLF>

MSG20: .ASCIZ /THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789/<15><12>

MSG21: .ASCIZ /ILLEGAL DEVICE/<CRLF>

MSG22: .ASCII / /

MSG23: .ASCIZ / /

MSG24: .ASCIZ /UNIBUS EXERCISER /

MSG25: .ASCIZ /OPT.CP=/

EM1: .ASCIZ /UNEXPECTED TRAP TO 4/

DH1: .ASCIZ /PCOFTP PHYSPC PSW CPUERR/

DF1: .BYTE 0,1,0,0,0

DT1: .EVEN  
.WORD VADR,VADR,\$TMP0,\$TMP2,0





9601	055534	001302	000000						
9602	055540	051105	047522	020122	EM6:	.ASCIZ	/ERROR DURING DATA CHECK-RELOC WAS BY CP/		
9603	055546	052504	044522	043516					
9604	055554	042040	052101	020101					
9605	055562	044103	041505	026513					
9606	055570	042522	047514	020103					
9607	055576	040527	020123	054502					
9608	055604	051144	000120						
9609	055610	051123	040503	051104	DH6:	.ASCIZ	/SRCADR DSTADR/		
9610	055616	020040	051504	040524					
9611	055624	051104	000						
9612		055630							
9613	055630	001276	001474	000000	DT6:	.EVEN			
9614	055636	051105	047522	020122	EM10:	.WORD	\$TMP0,VADR,0		
9615	055644	052504	044522	043516		.ASCIZ	?ERROR DURING DATA CHECK-RELOC WAS BY I/O?		
9616	055652	042040	052101	020101					
9617	055660	044103	041505	026513					
9618	055666	042522	047514	020103					
9619	055674	040527	020123	054502					
9620	055702	044440	047457	000					
9621	055707	123	041522	042101	DH10:	.ASCIZ	/SRCADR DSTADR DEVICE THAT DID XFER/		
9622	055714	020122	020040	051504					
9623	055722	040524	051104	020040					
9624	055730	042040	053105	041511					
9625	055736	020105	044124	052101					
9626	055744	042040	042111	054040					
9627	055752	042506	000122						
9628	055756	000	001	003	DF10:	.BYTE	0,1,3,0		
9629	055761	000							
9630									
9631	055762	001276	001474	001302	DT10:	.EVEN			
9632	055770	001304	000000			.WORD	\$TMP0,VADR,\$TMP2,\$TMP3,0		
9633	055774	044502	024124	024523	EM11:	.ASCIZ	/BIT(S) STUCK IN MICRO-BREAK REGISTER/		
9634	056002	051440	052524	045503					
9635	056010	044440	020116	044515					
9636	056016	051103	026517	051102					
9637	056024	040505	020113	042522					
9638	056032	044507	052123	051105					
9639	056040	000							
9640	056041	107	047517	042104	DH11:	.ASCIZ	/GOODDAT BAD DATA/		
9641	056046	052101	041040	042101					
9642	056054	042040	052101	000101					
9643	056062	000	000		DF11:	.BYTE	0,0		
9644						.EVEN			
9645	056064	001276	001300	000000	DT11:	.WORD	\$TMP0,\$TMP1,0		
9646	056072	041125	020105	047516	EM12:	.ASCIZ	/UBE NON-EXISTANT MEMORY ERROR/		
9647	056100	026516	054105	051511					
9648	056106	040524	052116	046440					
9649	056114	046505	051117	020131					
9650	056122	051105	047522	000122					
9651	056130	044120	051531	041040	DH12:	.ASCIZ	/PHYS BUSADR/		
9652	056136	051525	042101	000122					
9653	056144	002			DF12:	.BYTE	2		
9654		056146				.EVEN			
9655	056146	001226	000000		DT12:	.WORD	\$GDOAT,0		

9657	056160	026516	054105	051511				
9658	056166	040524	052116	046440				
9659	056174	046505	051117	020131				
9660	056202	051105	047522	000122				
9661	056210	044120	051531	040440	DH13:	.ASCIZ	/PHYS ADDRESS/	
9662	056216	042104	042522	051523				
9663	056224	000						
9664	056225	106	047514	052101	EM14:	.ASCIZ	/FLOATING POINT ERROR/	
9665	056232	047111	020107	047520				
9666	056240	047111	020124	051105				
9667	056246	047522	000122					
9668	056252	042011	040524	030524	DH14:	.ASCIZ	/ DTAT1	DATA2/
9669	056260	004411	004411	040504				
9670	056266	040524	000062					
9671						.EVEN		
9672	056272	001306	001262	001312	DT14:	.WORD	STMP4, SREG2, STMP6, SREG3, 0	
9673	056300	001264	000000					
9674	056304	004	000	004	DF14:	.BYTE	4, 0, 4, 0	
9675	056307	000						
9676	056310	042504	044526	042503	EM15:	.ASCIZ	/DEVICE HUNG/	
9677	056316	044040	047125	000107				
9678	056324	004411	040504	040524	DH16:	.ASCIZ	/	DATA1 DATA2/
9679	056332	004461	004411	020011				
9680	056340	020040	042040	052101				
9681	056346	031101	000					
9682	056351	005	000	005	DF16:	.BYTE	5, 0, 5, 0	
9683	056354	000						
9684		056356				.EVEN		
9685	056356	001422	001262	001432	DT16:	.WORD	FLTMP0, SREG2, FLTMP1, SREG3, 0	
9686	056364	001264	000000					
9687	056370	000000			ENDTAG:	.WORD	0	
9688					:*****			
9689					:THE FOLLOWING ASCII GETS OVERLAYED WHEN THE PROGRAM RUNS.			
9690	056372	050117	051105	052101	SWITCH:	.ASCII	/OPERATIONAL SWITCH SETTINGS/<CRLF>	
9691	056400	047511	040516	020114				
9692	056406	053523	052111	044103				
9693	056414	051440	052105	044524				
9694	056422	043516	100123					
9695	056426	053523	052111	044103		.ASCII	/SWITCH	USE/<CRLF>
9696	056434	004411	052411	042523				
9697	056442	200						
9698	056443	040	030440	004465		.ASCII	/ 15	HALT ON ERROR/<CRLF>
9699	056450	044011	046101	020124				
9700	056456	047117	042440	051122				
9701	056464	051117	200					
9702	056467	040	030440	004464		.ASCII	/ 14	LOOP ON TEST/<CRLF>
9703	056474	046011	047517	020120				
9704	056502	047117	052040	051505				
9705	056510	100124						
9706	056512	020040	031461	004411		.ASCII	/ 13	INHIBIT ERROR TYPEOUTS/<CRLF>
9707	056520	047111	044510	044502				
9708	056526	020124	051105	047522				
9709	056534	020122	054524	042520				
9710	056542	052517	051524	200				
9711	056547	040	030440	004462		.ASCII	/ 12	INHIBIT UBE/<CRLF>



9713	056562	052111	052440	042502			
9714	056570	200					
9715	056571	040	030440	004461	.ASCII / 11	INHIBIT ITERATIONS/<CRLF>	
9716	056576	044411	044116	041111			
9717	056604	052111	044440	052124			
9718	056612	051105	052101	047511			
9719	056620	051516	200				
9720	056623	040	030440	004460	.ASCII / 10	BELL ON ERROR/<CRLF>	
9721	056630	041011	046105	020114			
9722	056636	047117	042440	051122			
9723	056644	051117	200				
9724	056647	040	020040	004471	.ASCII / 9	LOOP ON ERROR/<CRLF>	
9725	056654	046011	047517	020120			
9726	056662	047117	042440	051122			
9727	056670	051117	200				
9728	056673	040	020040	004470	.ASCII ? 8	INHIBIT RELOCATION VIA I/O DEVICE?<CRLF>	
9729	056700	044411	044116	041111			
9730	056706	052111	051040	046105			
9731	056714	041517	052101	047511			
9732	056722	020116	044526	020101			
9733	056730	027511	020117	042504			
9734	056736	044526	042503	200			
9735	056743	040	020040	004467	.ASCII / 7	INHIBIT TYPEOUT OF THIS TEXT AND SYS SIZE/<CRLF>	
9736	056750	044411	044116	041111			
9737	056756	052111	052040	050131			
9738	056764	047505	052125	047440			
9739	056772	020106	044124	051511			
9740	057000	052040	054105	020124			
9741	057006	047101	020104	054523			
9742	057014	020123	044523	042532			
9743	057022	200					
9744	057023	040	020040	004466	.ASCII / 6	INHIBIT RELOCATION/<CRLF>	
9745	057030	044411	044116	041111			
9746	057036	052111	051040	046105			
9747	057044	041517	052101	047511			
9748	057052	100116					
9749	057054	020040	032440	004411	.ASCII / 5	INHIBIT ROUND ROBIN RELOCATION/<CRLF>	
9750	057062	047111	044510	044502			
9751	057070	020124	047522	047125			
9752	057076	020104	047522	044502			
9753	057104	020116	042522	047514			
9754	057112	040503	044524	047117			
9755	057120	200					
9756	057121	040	020040	004464	.ASCII / 4	INHIBIT RANDOM DISK ADDRESS/<CRLF>	
9757	057126	044411	044116	041111			
9758	057134	052111	051040	047101			
9759	057142	047504	020115	044504			
9760	057150	045523	040440	042104			
9761	057156	042522	051523	200			
9762	057163	040	020040	004463	.ASCII 3	INHIBIT MBT/<CRLF>	
9763	057170	044411	044116	041111			
9764	057176	052111	046440	052102			
9765	057204	200					
9766	057205	040	020040	004462	.ASCII / 2	THESE THREE SWITCHES/<CRLF>	
9767	057212	052011	042510	042523			

9769	057226	051440	044527	041524			
9770	057234	042510	100123				
9771	057240	020040	030440	004411	.ASCII	/ 1	ARE ENCODED TO SELECT RELOCATION/<CRLF>
9772	057246	051101	020105	047105			
9773	057254	047503	042504	020104			
9774	057262	047524	051440	046105			
9775	057270	041505	020124	042522			
9776	057276	047514	040503	044524			
9777	057304	047117	200				
9778	057307	040	020040	004460	.ASCII	/ 0	ON THE FOLLOWING DEVICES:/<CRLF>
9779	057314	047411	020116	044124			
9780	057322	020105	047506	046114			
9781	057330	053517	047111	020107			
9782	057336	042504	044526	042503			
9783	057344	035123	200				
9784	057347	011	027060	027056	.ASCII	? 0...	RP11/RP03?<CRLF>
9785	057354	050122	030461	051057			
9786	057362	030120	100063				
9787	057366	030411	027056	051056	.ASCII	? 1...	RK11/RK05?<CRLF>
9788	057374	030513	027461	045522			
9789	057402	032460	200				
9790	057405	011	027062	027056	.ASCII	? 2...	NOT USED?<CRLF>
9791	057412	047516	020124	051525			
9792	057420	042105	200				
9793	057423	011	027063	027056	.ASCII	? 3...	NOT USED?<CRLF>
9794	057430	047516	020124	051525			
9795	057436	042105	200				
9796	057441	011	027064	027056	.ASCII	? 4...	RH70/RP04?<CRLF>
9797	057446	044122	030067	051057			
9798	057454	030120	100064				
9799	057460	032411	027056	051056	.ASCII	? 5...	RH70/RS04 OR RS03?<CRLF>
9800	057466	033510	027460	051522			
9801	057474	032060	047440	020122			
9802	057502	051522	031460	200			
9803	057507	011	027066	027056	.ASCII	? 6...	NOT USED?<CRLF>
9804	057514	047516	020124	051525			
9805	057522	042105	200				
9806	057525	011	027067	027056	.ASCII	? 7...	NOT USED?<CRLF>
9807	057532	047516	020124	051525			
9808	057540	042105	000200				

AA	001505	644#	1103#			
AOCB2	011170	2265	2267#			
AOCB5	012014	2495	2497#			
AOCB6	012516	2651	2652#	2654#		
AOCB7	013440	2874	2875#	2876	2878#	
AOCO	007062	1664	1665	1666	1668#	
AOC1	007762	1893	1894	1895	1897#	
AOC2	010772	2195	2197#			
AOC5	011614	2423	2424	2426#		
AOC6	012320	2594	2595	2597#		
AOC7	013306	2832	2833	2835#		
AODO	014134	3041	3042	3043	3045#	
AOD1	014240	3078	3079	3081#		
AOD1A	014464	3161	3162	3163	3165#	
AOD1B	014502	3170	3171	3173#		
AOD2	015124	3315	3316	3318#		
AOD3	015702	3507	3509#			
AOD6	016260	3608	3609	3611#		
AOD7	016736	3724	3725	3726	3728#	
ARBEX	033376	6314	6315#			
ARFIN	033364	6299	6321	6342	6344	6355#
ASHCLO	025230	5144#				
ASHCRO	025306	5165#				
ASHLO	025020	5094#				
ASHL1	025616	5276#				
ASHRO	025134	5121#				
ASHR1	025704	5299#				
ASLB1	010332	2027	2028	2030#		
ASLB1A	010556	2114	2115	2117#		
ASLB3	012004	2489	2490	2492#		
ASLB4	011274	2302	2303	2304	2306#	
ASLB6	012500	2643	2644	2645	2647#	
ASLB7	013536	2906	2907	2909#		
ASLC	007204	1708	1709	1710	1711	1713#
ASL1	010136	1956	1957	1958	1960#	
ASL3	011530	2392	2393	2395#		
ASL4	011064	2227	2228	2229	2231#	
ASL6	012270	2582	2583	2585#		
ASL7	013134	2779	2780	2782#		
ASRB1	010426	2063	2065#			
ASRB1A	010442	2069	2070	2072#		
ASRB2	011240	2287	2288	2290#		
ASRB2A	011256	2295	2296	2298#		
ASRB5	011744	2470	2471	2473#		
ASRB6	012616	2683	2684	2686#		
ASRB7	013554	2913	2914	2916#		
ASRO	007232	1722	1723	1724	1726#	
ASR1	010024	1913	1914	1915	1917#	
ASR2	011006	2201	2202	2204#		
ASR3	011514	2386	2388#			
ASR6	012152	2544	2545	2547#		
ASR7	013170	2793	2794	2796#		
BICB1	014664	3233	3234	3236#		
BICB1A	014706	3244	3247#	3305	3306	
BIC?	014046	3012	3013	3014	3016#	















E01

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 193  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0211

MAPLO = 170200	442#	5804	5837	5848*	8960	9101
MAPLO0 = 170200	378#	442				
MAPLO1 = 170204	380#	444				
MAPLO2 = 170210	382#	446				
MAPLO3 = 170214	384#	448				
MAPLO4 = 170220	386#	450				
MAPLO5 = 170224	388#	452				
MAPLO6 = 170230	390#	454				
MAPLO7 = 170234	392#	456				
MAPL1 = 170204	444#	5846*				
MAPL10 = 170240	394#					
MAPL11 = 170244	396#					
MAPL12 = 170250	398#					
MAPL13 = 170254	400#					
MAPL14 = 170260	402#					
MAPL15 = 170264	404#					
MAPL16 = 170270	406#					
MAPL17 = 170274	408#					
MAPL2 = 170210	446#					
MAPL20 = 170300	410#					
MAPL21 = 170304	412#					
MAPL22 = 170310	414#					
MAPL23 = 170314	416#					
MAPL24 = 170320	418#					
MAPL25 = 170324	420#					
MAPL26 = 170330	422#					
MAPL27 = 170334	424#					
MAPL3 = 170214	448#					
MAPL30 = 170340	426#					
MAPL31 = 170344	428#					
MAPL32 = 170350	430#					
MAPL33 = 170354	432#					
MAPL34 = 170360	434#					
MAPL35 = 170364	436#					
MAPL36 = 170370	438#					
MAPL37 = 170374	440#					
MAPL4 = 170220	450#					
MAPL5 = 170224	452#					
MAPL6 = 170230	454#					
MAPL7 = 170234	456#					
MAPTBL 001656	681#	1143*	1144*	9096	9099*	9128*
MAPTST 030176	5804#					
MAPTMO 030324	5803	5839	5844#			
MARKEX 024256	4935	4939	4943#			
MARK1 024232	4931	4933#				
MBRX 026446	5448#					
MBTAS = 160116	493#					
MBTBA = 160104	488#	881				
MBTBAE = 160174	497#	882				
MBTCS1 = 160100	486#	879				
MBTCS2 = 160110	490#	884				
MBTCS3 = 160176	498#	887				
MBTDB = 160120	494#					
MBTDT = 160126	496#	890				
MBTER = 160114	492#	886				











J01

CEQKCC PDP 11/70 CPU EXERCISOR  
CEQKCC.P11 03-MAR-78 13:13

MACY11 30A(1052) 03-MAR-78 13:15 PAGE 198  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0216

ROR7	013210	2801	2802	2804*														
RPCC	002172	843*																
RP3BA	002110	814*	6999*	7288*														
RP3CS	002104	812*	1328*	1329	6989*	6994*	7217	7221*	7222	7236	7246*	7252	7259*	7260				
		7289*	7295	7300*	7301	8133												
RP3DA	002112	815*	6970*	6996*	7285*													
RP3DC	002114	816*	6997*	7286*														
RP3DRV	036772	760	6929*															
RP3DS	002100	810*	1335	6987	9399													
RP3ER	002102	811*	7266															
RP3ERR	040774	7220	7228*	7257	7298													
RP3FUN	002060	783*	6993*	7202*	7203	7208*	7209	7258*	7299*									
RP3HAN	002026	760*	6657	6769														
RP3HDA	002042	771*	6996	7285														
RP3HDC	002044	772*	6969*	6997	7286													
RP3HST	001676	691*	1146	6630	6635	6681	6754*	6768*	6771	6773	6930*	6931	7215	7228*				
		7235*	7238*	7250	7293	7312*												
RP3HWC	001716	704*	6647*	6998	7287													
RP3LOO	041022	7216	7235*	7251	7294													
RP3N44	001764	733*	6661*	7278	7280*	7283												
RP3N4L	001762	732*	6660*	7275	7288													
RP3OLD	001732	715*	6659*	6662*	6975	6978	6981*	6999	7243									
RP3PSM	002120	818*	6992*															
RP3REA	041350	7212	7293*															
RP3RPT	040644	6934	7201*															
RP3SRV	040674	6991	7208*															
RP3TRY	002072	795*	6971*	7219	7224*	7241*	7256	7262*	7270*	7297	7303*							
RP3UNI	002012	749*	6655*	6982	6985*	6989	7244	7284										
RP3VEC	002116	817*	6991*															
RP3WC	002106	813*	6998*	7287*														
RP3WCK	041112	7210	7250*															
RP3WTR	037310	6987*	6988	7206	7226													
RP31	041046	7204	7218	7241*														
RP310	001564	668*	6977*	6980*	6984*	6994	7242*	7243*	7244*	7246	7282*	7283*	7284*	7289				
RP311	001566	669*	6935*	6950	7230	7254												
RP32	041076	7245*	7264															
RP33	041310	7207	7285*	7305														
RP4BA	002146	833*	7139*	7520*														
RP4BAE	002150	834*	7138*	7519*														
RP4CS1	002142	831*	1319	1349*	7130*	7470*	7491*	7521*	9403									
RP4CS2	002154	836*	1348*	1350	1358*	6764*	7133*	7463*	7502*	7508	7532*							
RP4CS3	002156	837*																
RP4DA	002152	835*	7136*															
RP4DC	002164	840*	7135*															
RP4DRV	040004	764	7079*															
RP4DS	002160	838*	1352	7459	7471	7482	7488	7497	7516	7527								
RP4ERR	042504	7462	7474*	7500	7530													
RP4ER1	002162	839*																
RP4ER2	002166	841*																
RP4ER3	002170	842*																
RP4FUN	002066	786*	7129*	7446*	7447	7451*	7452	7531*										
RP4HAN	002036	764*																
RP4HDA	002052	775*	7123*	7136														
RP4HDC	002054	776*	7122*	7135														
RP4HST	001706	695*	7080*	7081	7086*	7457	7468	7474*	7481*	7484*	7495	7525	7538*					









TRAP1	021402	4285	4300#				
TRAP1C	021442	4299	4312#				
TRTVEC#	000014	185#					
TST81	010536	2105	2106	2108#			
TST82	011342	2327	2329#				
TST82A	011352	2332	2334#				
TST86	012410	2616	2617	2618	2620#		
TST0	007022	1645	1646	1647	1648	1650#	
TST1	005524	1449#					
TST10	011104	2242#					
TST11	011422	2359#					
TST12	011644	2443#					
TST13	012064	2524#					
TST14	012344	2608#					
TST15	012736	2730#					
TST16	013340	2851#					
TST17	013634	2940#					
TST2	006554	1566#					
TST20	014172	3063#					
TST21	014516	3182#					
TST22	015046	3303#					
TST23	015332	3390#					
TST24	015536	3460#					
TST25	015726	3523#					
TST26	016106	3579#					
TST27	016404	3640#					
TST3	006750	1629#					
TST30	016560	3682#					
TST31	016776	3745#					
TST32	017126	3783#					
TST33	017322	3836#					
TST34	020030	3964#					
TST35	020322	4055#					
TST36	020644	4159#					
TST37	021070	4176	4225#				
TST4	007266	1745#					
TST40	021276	4280#					
TST41	021506	4328#					
TST42	022214	4448#					
TST43	022374	4494#					
TST44	022602	4542#					
TST45	023010	4608#					
TST46	023142	4656#					
TST47	023734	4829#					
TST5	007662	1867#					
TST50	024164	4918#					
TST51	024260	4968#					
TST52	024466	5011#					
TST53	024726	5076#					
TST54	025372	5194#					
TST55	025522	5245#					
TST56	025770	5328#					
TST57	026046	5351#					
TST6	010214	1990#					
TST60	026216	5394#					





UIPAR4=	177670	275#												
UIPAR5=	177672	276#												
UIPAR6=	177674	277#												
UIPAR7=	177676	278#												
UIPOR0=	177620	249#	5723											
UIPOR1=	177622	250#												
UIPOR2=	177624	251#												
UIPOR3=	177626	252#												
UIPOR4=	177630	253#												
UIPOR5=	177632	254#												
UIPOR6=	177634	255#												
UIPOR7=	177636	256#												
UIPAR0=	177640	260#	5476	5728	6453*	6454	6456							
UIPAR1=	177642	261#	6454*	6455*										
UIPAR2=	177644	262#	6456*	6457*										
UIPAR3=	177646	263#												
UIPAR4=	177650	264#	5747*											
UIPAR5=	177652	265#												
UIPAR6=	177654	266#	5476*											
UIPAR7=	177656	267#	6458*											
UIPOR0=	177600	238#	5720	6449*										
UIPOR1=	177602	239#	6450*											
UIPOR2=	177604	240#	6451*											
UIPOR3=	177606	241#												
UIPOR4=	177610	242#	5748*											
UIPOR5=	177612	243#												
UIPOR6=	177614	244#	5477*											
UIPOR7=	177616	245#	6452*											
UM	= 140000	516#	4188	4364	5017	5023	5049	5367	5381	5466	5600	5609		
UNITNO	001554	664#	6646*	6650	6655									
USESTK=	000600	78#												
UM6	012076	2527#	2531*	2538*	2543*	2550*	2558*	2565*	2572*	2579*	2581*	2587*	2593*	2599*
UM7	013034	2750#	2853	2856*	2861*									
UM7	013040	2748#	2753#											
VADR	001474	638#	6532*	6719*	6796*	7938	7939*	7945*	7947	7949*	8987	9038*	9248*	9249*
		9335*	9336*	9351*	9352*	9364*	9365*	9542	9553	9568	9600	9613	9631	
X	053124	9282	9285#											
XOR0	023310	4691	4692	4693	4696#									
XOR1	023376	4716	4717	4718	4719	4721	4724#							
XOR24	023432	4738#												
XOR35	023702	4807	4808	4812#										
XOR6	023520	4743	4746	4748	4755#									
XOR6A	023524	4741*	4742*	4747*	4749*	4753	4759#	4764*	4775*	4777*				
XOR6B	023526	4750*	4751*	4752*	4753	4760#								
XOR7	023732	4823#												
XXDP	003240	1093#	1113											
XXDPC	003241	1094#	1108#	1405										
SAC0	001402	623#	5893*	5896*	5903*	5905	5914*	5915	5924*	5928*	5932*	5936*	5951*	5966*
		5974*	5978*	5984*	5986	5992*	5994	5999*	6012*	6026*	6033*	6062*	6065*	6072*
		6074	6083*	6084	6093*	6097*	6101*	6105*	6120*	6135*	6143*	6147*	6153*	6155
		6161*	6163	6168*	6181*	6195*	6202*	6219	6230	6244	6249	6260	6262	8827
SAC1	001404	624#	5892*	5896	5899	5905*	5909	5915*	5918	5925*	5939	5947	5953*	5956*
		5957	5965*	5969	5972*	5981	5986*	5989	5994*	5996	6002	6008	6014*	6017*
		6018	6025*	6030	6036*	6038*	6039	6061*	6065	6068	6074*	6078	6084*	6087
		6094*	6108	6116	6122*	6125*	6126	6134*	6138	6141*	6150	6155*	6158	6163*



\$LPERR 001212

578#	1022#	1176#	1461#	1462#	1463	1578#	1579#	1580	2742#	2743#	2744	3795#
3796#	3797	4340#	4341#	4342	4668#	4669#	4670	5088#	5089#	5090	5561#	5562#
5563	5879#	5880#	5881	6546#	6692	6745#	7684	7685#	7696#	7742	7743#	7752#
7826	7841#	7850	7894	9266	9267#	9269#	9285	9339	9340#	9342#	9344	9353
9354#	9356#	9357	9368	9369#	9371#	9374						

\$MAINT 001604

\$MXCNT 044772

\$NULL 001250

\$NWTST= 000001

677#	6855#	6865#	7960									
7839	7850#											
592#	8225	8256										
1446#	1563#	1626#	1742#	1864#	1987#	2137#	2239#	2356#	2440#	2521#	2605#	2727#
2848#	2937#	3060#	3179#	3300#	3387#	3457#	3520#	3576#	3637#	3678#	3680	3741#
3743	3780#	3833#	3961#	4052#	4154#	4156	4222#	4277#	4325#	4445#	4491#	4539#
4605#	4653#	4824#	4826	4915#	4944#	4946	5008#	5073#	5188#	5190	5238#	5240
5322#	5324	5348#	5387#	5389	5442#	5444	5461#	5505#	5525#	5546#	5638#	5640
5678#	5680	5735#	5737	5796#	5798	5857#	5859	6044#	6046	6289#	6358#	6360

\$OCNT 047462

\$OCTVL 050032

\$OMODE 047464

\$OVER 044742

\$PASS 001200

\$POWER 051364

\$PWROD 051352

\$PWROD 051230

\$PWRTG 051346

\$PWRRP 051276

\$QUES 001326

\$RAND 050604

\$ROCHR= \*\*\*\*\*

\$RODEC= \*\*\*\*\*

\$ROLIN= \*\*\*\*\*

\$RODOCT= \*\*\*\*\*

\$REGAD 001254

\$REGO 001256

U  
U  
U

8382#	8411#	8424#										
8509	8537#											
8377#	8381#	8386	8389#	8400#	8426#							
7807	7827	7837	7844#									
572#	1131#	1407	6390	6890#	6891#	6902	6923	7833	7851	7972		
8870	8877#											
8872#												
1169	8843#	8867										
8870#												
8852	8857#											
617#	7900											
6941	7019	7092	7158	8722#	8758	9076						

\$REG1 001260

596#	5892	5903	5925	5965	5972	5992	6025	6033	6034	6036	6061	6072
598#	6134	6141	6161	6194	6202	6203	6205	8807				
6094	5893	5914	5924	5966	5974	5984	6026	6062	6083	6093	6135	6143
599#	6195											

\$REG2 001262

\$REG3 001264

\$REG4 001266

\$REG5 001270

\$REG6 001272

\$REG7 001274

\$RESRE 050106

\$RTAN 001472

\$SAVPA 053264

\$SAVPS 053272

\$SAVRE 050050

\$SAVR6 051362

\$SCOPE 044534

\$SETUP= 000037

\$STUP = 177777

\$SVLAD 044720

\$SVPC = 000220

\$SWR = 167377

600#	5899#	5951	5981#	6012	6030#	6068#	6120	6150#	6181	6199#	9672	9685
601#	5939#	5953	6002#	6014	6034#	6108#	6122	6171#	6183	6203#	9672	9685
602#	9266#	9269	9339#	9342	9353#	9356	9368#	9371				
603#	6302#	6861#	6871#	9172#	9180#	9190#	9222#	9223	9227#			
604#												
605#												
8572#	8911											
637#	1431	5004	6860									
9304	9318	9329#										
9295#	9326	9330#										
8556#	8910											
8851#	8857	8858#	8859#	8876#								
1163	4174	4180	4274	5503	7805#							
1138#	1163	1165	1167	1169	1171	1172	1173	1175	1274	6888	7886	
1138#												
7815	7840#											
555#	560											
1#	32	47	48	49	50	51	52	53	54	614	615	616
1172	1173	1175	1176	1452	1569	1631	1747	1869	1992	2142	2244	2361
2445	2526	2610	2733	2853	2942	3065	3184	3305	3392	3462	3525	3581









.\$OB20	1#		
.\$OB20	1#	6#	8495
.\$DIV	1#		
.\$EOP	1#	6876	
.\$ERRO	1#	7851	
.\$ERRT	1#		
.\$MULT	1#		
.\$POWE	1#	8838	
.\$RAND	1#	7#	8710
.\$RDEE	1#		
.\$RDOC	1#		
.\$READ	1#		
.\$SAVE	1#	8538	
.\$SB20	1#		
.\$SB20	1#		
.\$SCOP	1#	2#	7792
.\$SIZE	1#		
.\$SUPR	1#		
.\$TRAP	1#	8880	
.\$TYPB	1#		
.\$TYPD	1#	8427	
.\$TYPE	1#	562#	8181
.\$TYPO	1#	8349	
.\$1170	1#	71	

. ABS. 057544 000

ERRORS DETECTED: 0

CEQKCC.BIN,CEQKCC.LST/CRF/SOL/NL:TOC=DSKZ:CEQKCC.SML,CEQKCC.P11  
RUN-TIME: 32 38 3 SECONDS  
RUN-TIME RATIO: 1039/75=13.7  
CORE USED: 32K (63 PAGES)



L02

M02