

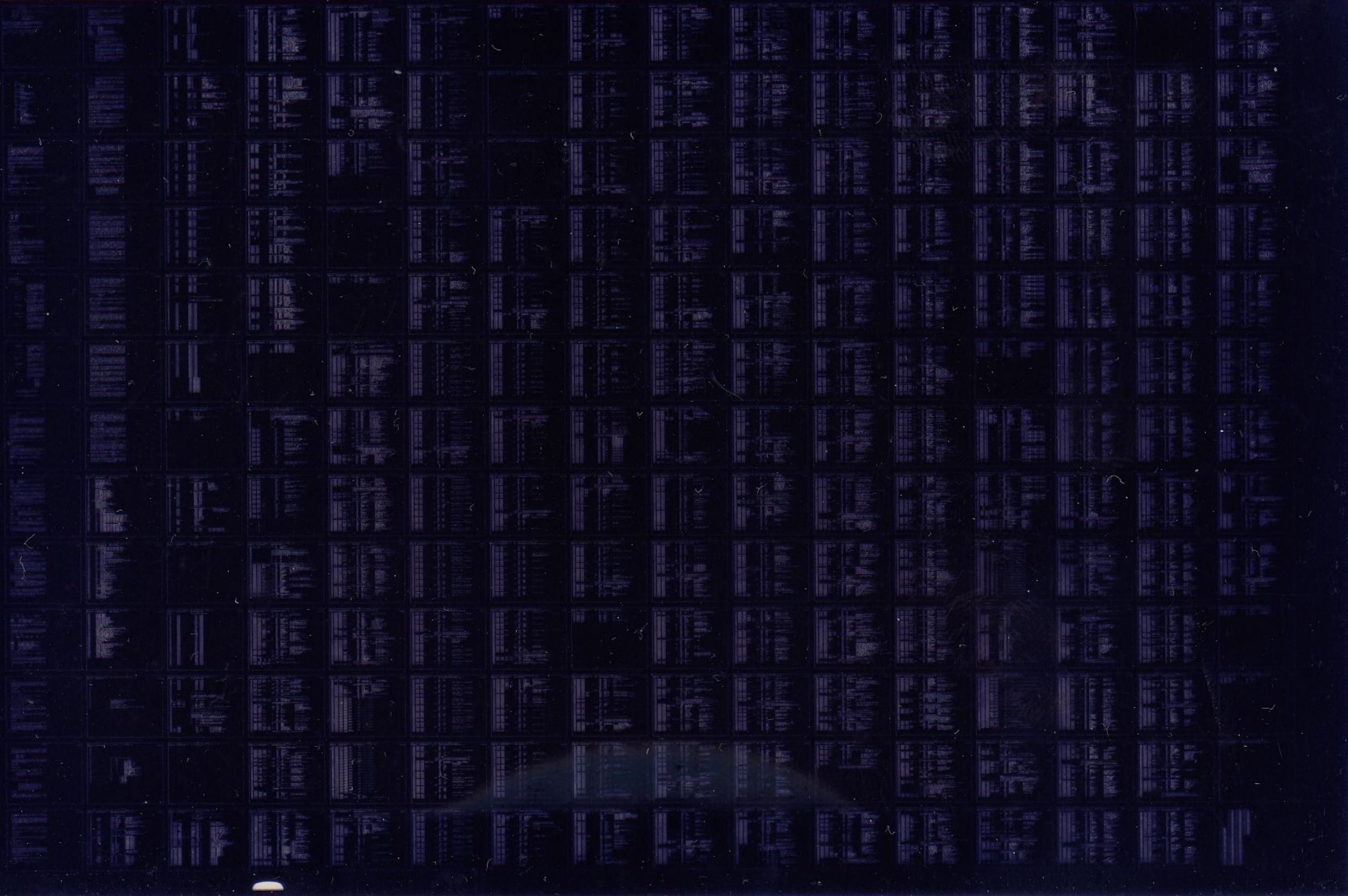
PDP-11/70

11/70-74 MP INST EXR
CEQKCD0

AH-7996D-MC

COPYRIGHT ' 75-80
FICHE 1 OF 2

JAN 1980
digital
MADE IN USA



PDP-11/70

11/70-74 MP INST EXR
CEQKCD0

AH-7996D-MC

COPYRIGHT '75-80

FICHE 2 OF 2

JAN 1980

digital

MADE IN USA

IDENTIFICATION

PRODUCT CODE: AC-7994D-MC
 PRODUCT NAME: CEOKCDO PDP11/70-74MP CPU INSTRUCTION EXERCISER
 DATE CREATED: MAY, 1979
 MAINTAINER: DIAGNOSTIC ENGINEERING
 AUTHOR(S): DONALD W. MONROE-REV B
 JOHN ADAMS-REV A

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) 1975,1979 by Digital Equipment Corporation

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT COROPORATION

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

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 Compatability
8.7	PSW and Margin Tables
8.8	I/O Device Address Modifcations
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
9.5	Relocation Algorithm

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 SELECTED, the program will relocate using available disks (RP03, RK05, RP04, 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)

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-11/70-74MP (KB11-B/C OR KB11-CM) 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/RP03, RK11/RK05, RH70/RP04, 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

Although this program is a test of the CPU cluster, it is

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

DEKBA	DEKBF.
DEKBB	DEKBG
DEKBC	DEMJA
DEKBD	DEFPA
DEKBE	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.

START AT ADDRESS 230 AS AN AID TO CUTTING THE JUMPERS FOR THE PROCESSOR ID REGISTER.

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 a description of the messages)

5.0 OPERATING PROCEDURE5.1 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 DISK	This switch when set will CAUSE RELOCATION TO BE DONE BY A DISK INSTEAD OF THE CPU. See section 9.5 for a description of relocation.

- SW7 INHIBIT SYSTEM SIZE TIMEOUT This switch when set will inhibit the timeout of the switch definitions and the disks that will be used for relocation. (Timeout 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 present if program was loaded in chain

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 (\$ERROR). 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 PHYSIC PC   PSW   MAINT  TEST NO SUB-PASS CNT  
UUUUUU VVVVVVVV WWWWWW XXXXXX YYYYYY SSSSSS PPPPPP
```

where:

```
UUUUUU - Virtual PC of the error call.  
VVVVVVVV - Physical PC of the error call.  
WWWWW - PSW at the time of the error call.
```

XXXXXX = Contents of the maintenance register(17777750).
 YYYYYY = Test number.
 SSSSSS = Sub-pass count (0 thru 5)
 PPPPPP = Pass count

HHH:MM:SS Represents the elapsed run time of the program, since the most previous start, where: HHH = 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.

6.1.1 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.
 PPPPPPPP = Physical PC calculated as described above.
 YYYYYY = PSW that was pushed on the stack.
 ZZZZZZ = Contents of the CPU error register(17777766).

6.1.2 Unexpected Trap to 114

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

V, P, and Y = are the same as described in 6.1.1.
 ZZZZZZ = Contents of the memory error register (777744).
 EEEEEEEE = Contents of the error address registers combined into a 22 bit address (777740 & 777742).

6.1.3 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 code. The source data refers to the unrelocated code and the

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).

6.1.4 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.

6.1.5 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.

6.1.6 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.

If SW<9> is not set, relocation is restarted.

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-Existant 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	CCCCC	DDDDDD	EEEEEE	FFFFFF	GGGGGG

ER	CS3
HHHHHH	JJJJJJ

AAAAAA = Control and status register #1 (760100).
 BBBBBB = word count register (760102).
 CCCCC = Bus address register (760104).
 DDDDD = Bus address extended register (760174).
 EEEEE = Maintenance register #2 (760106).
 FFFFF = Control and status register #2 (760110).
 GGGGG = Status register (760112).
 HHHHH = Error register (760114).
 JJJJJ = Control and status register #3 (760176).

6.1.10 MBT Non-Existant 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 (DEFFA DEFPB) should be used to isolate the problem.

6.1.2 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. Off

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 5.3) 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.

7.2 Exceptions

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:

RP03----176714
RK05----177404
RP04----176700
RS03/4--172040

If the system has both an RP03 and an RP04, the branch instruction at 100\$, in the "size routine" must be replaced by a nop (240) for both devices to be used. This branch is approximately at address 4552.

8.0 MISCELLANEOUS

8.1 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, RP04, and 2 RS04's ---9 minutes.

8.2 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.

8.3 Pass Count

There are two words used for effective pass count. Location "SUBPASS" and "\$PASS". Subpass contains the ASCII representation of the subpass count. This is used to index the PSW table and margin table (see section 8.7).

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.

8.4 Iterations

Sub-test iterations are not performed until the pass count (\$PASS) 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.

8.5 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.

8.6 ACT-11 Compatability

The program is fully ACT-11 compatable.

8.7 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.

8.8 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 RP03 or RP04 see section 7.2.

8.9 Power fail

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

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 and RTT instructions.

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 77 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 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 ABOUT 1.3K WORDS. 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, 0 is loaded as the bus address and the UBE is started again.

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 77 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 76 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 processor mode is restored.

9.5 Relocation Algorithm

9.5.1 Section Relocation

As each section is entered the virtual start address is saved in location 'RSTAD' 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 CLEAR, this routine will relocate the section via the CP (see 9.5.3). If SW<8> is 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 value for the PSW and maintenance registers is taken from

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<8> is CLEAR, 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:

```
1$: MOV (R0)+,(R2)+  
    CMP R0,R3  
    BNE 1$
```

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<8> is 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<8> is CLEAR, relocation is performed VIA the CP (see section 9.5.3).

If SW<8> is 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 'DEVINDX'. 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 DEVINDX, 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 handler status words and device handler unit number tables,

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" (RUNTBL). 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
<?> = 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 TABLE, and the routine exits as described above.

44	OPERATIONAL SWITCH SETTINGS
73	BASIC DEFINITIONS
178	CACHE REGISTER DEFINITIONS
209	CPU REGISTER DEFINITIONS
223	MEMORY MANAGEMENT DEFINITIONS
372	UNIBUS MAP REGISTER DEFINITIONS
530	CIS OPCODE DEFINITIONS
599	TRAP CATCHER
606	STARTING ADDRESS(ES)
618	ACT11 HOOKS
645	COMMON TAGS
780	DEVICE HANDLER STATUS WORDS
794	DEVICE HANDLER WORD COUNTS
805	DEVICE HANDLER OLD BASE ADDRESS
822	DEVICE HANDLER NEW BASE ADDRESSES
839	DEVICE HANDLER UNIT NUMBER
850	ADDRESS OF THE DEVICE HANDLERS
861	DEVICE HANDLER DISK ADDRESS TABLE
873	DEVICE HANDLER FUNCTION TABLE
883	DEVICE HANDLER RETRY COUNT
896	DEVICE REGISTER TABLES
903	RP11/RP03 REGISTERS
914	RK11/RK05 REGISTERS
924	RH70/RP04 REGISTERS
942	RH70/RS04 REGISTERS
955	UNIBUS EXERCISER REGISTER ADDRESS TABLE
968	MASS BUS TESTER REGISTER ADDRESSES
991	ERROR POINTER TABLE
1089	PROGRAM INITIALIZATION
1092	MICRO-BREAK REGISTER TEST
1615	SYSTEM SIZER
1798	T1 MEMORY VERIFICATION TEST
1807	START OF SECTION 0
1917	T2 CHECK BRANCH INSTRUCTIONS
1926	START OF SECTION 1
1982	T3 TEST UNIARY CONDITION CODES
2100	T4 CHECK REGISTER SELECTION
2224	T5 TEST UNIARY WORD INSTRUCTIONS USING ADDRESS MODE 1
2349	T6 CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1
2501	T7 CHECK UNIARY WORD OPS USING ADDRESS MODES 2 & 4
2605	T10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4
2724	T11 CHECK UNIARY WORD OPS USING ADDRESS MODES 3 & 5
2810	T12 CHECK UNIARY BYTE OPS USING ADDRESS MODES 3 & 5
2893	T13 CHECK UNIARY WORD OPS USING ADDRESS MODE 6 (PC)
2979	T14 CHECK UNIARY BYTE OPS (EVEN/ODD) USING ADDRESS MODE 6 (PC)
3103	T15 CHECK UNIARY WORD OPS USING ADDRESS MODE 7
3112	START OF SECTION 2
3226	T16 CHECK UNIARY BYTE OPS USING ADDRESS MODE 7
3317	T17 CHECK BINARY OPS USING ADDRESS MODE 0
3442	T20 CHECK BINARY OPS USING ADDRESS MODE 1
3563	T21 CHECK BINARY BYTE OPS USING ADDRESS MODE 1
3686	T22 CHECK BINARY WORD OPS USING ADDRESS MODE 2 & 4
3775	T23 CHECK BINARY BYTE OPS USING ADDRESS MODE 2 & 4
3847	T24 CHECK BINARY WORD OPS USING ADDRESS MODES 3 & 5
3912	T25 CHECK BINARY BYTE OPS USING ADDRESS MODES 3 & 5
3970	T26 CHECK BINARY OPS USING ADDRESS MODE 6

4033	T27	CHECK BINARY BYTE OPS USING ADDRESS MODE 6
4076	T30	CHECK BINARY WORD OPS USING ADDRESS MODE 7
4141	T31	SOME MISCELLANEOUS OPERATIONS INVOLVING THE PC
4182	T32	CHECK BINARY BYTE OPS USING ADDRESS MODE 0
4191		START OF SECTION 3
4237	T33	CHECK BINARY BYTE OPS USING ADDRESS MODE 7
4367	T34	CHECK JUMP INSTRUCTIONS
4460	T35	CHECK JSR INSTRUCTIONS
4564	T36	CHECK IOT TRAP (AND ROLB/ASLB)
4634	T37	CHECK EMT TRAP SEQUENCE
4691	T40	CHECK TRAP INSTRUCTION TRAP SEQUENCE
4741	T41	CHECK STACK OVERFLOW
4750		START OF SECTION 4
4863	T42	CHECK THAT ALL RESERVED INSTRUCTIONS TRAP
4952	T43	CHECK THAT ALL BITS IN THE PSW CAN BE SET AND CLEARED
5004	T44	CHECK THAT ALL BITS IN THE CURRENT STACK PTR CAN BE SET CLEARED
5072	T45	CHECK THAT 'C' BIT SETS/CLEARs PROPERLY
5122	T46	CHECK EXTENDED INSTRUCTION SET
5131		START OF SECTION 5
5295	T47	SOB TEST
5388	T50	CHECK THE MARK INSTRUCTION
5419	T51	RTT/RTI TEST
5486	T52	SECOND RTT TEST
5553	T53	CHECK ASH, ASHC, MUL, AND DIV INSTRUCTIONS
5562		START OF SECTION 6
5670	T54	CHECK MUL
5722	T55	CHECK THE DIV INSTRUCTION
5808	T56	DIVIDE AGAIN
5836	T57	CHECK SPL INSTRUCTION
5877	T60	CHECK PIRQ LOGIC
5934	T61	CHECK MICRO-BREAK REGISTER
5959	T62	CHECK MFPI/MTPI INSTRUCTIONS
6005	T63	CHECK ILLEGAL HALT
6027	T64	CHECK RESET IN SUPER/USER MODE
6050	T65	TEST STACK LIMIT REGISTER
6059		START OF SECTION 7
6144	T66	MEMORY MANAGEMENT REGISTER TESTS
6186	T67	PAR TEST
6245	T70	CHECK KT ABORT LOGIC
6308	T71	MAPPING REGISTER TESTS
6371	T72	FLOATING POINT TEST 1
6388		START OF SECTION 8
6561	T73	FLOATING POINT TEST 2
6733		FLOATING POINT MULTIFLY ROUTINE
6745		FLOATING POINT DIVIDE ROUTINE
6756		FLOATING POINT ADD ROUTINE
6814	T74	CHECK MFPT INSTRUCTION (KB11-E/EM ONLY)
6830		START OF SECTION 9
6861	T75	COMMERCIAL INSTRUCTION SET TEST
7476	T76	TELETYPE AND CLOCK TESTS
7547	T77	TURN ON UBE AND MBT
7590		STMM ROUTINE
7678		RELOCATION ROUTINE
7735		I/O RELOCATION MONITOR
8041		END OF SUB-PASS ROUTINE
8085		END OF PASS ROUTINE

8133	RP11/RP03 HANDLER
8210	RK11/RK05 HANDLER
8283	RH70/RP04 HANDLER
8350	RH70/RS04 HANDLER
8405	RP11/RP03 SERVICE ROUTINE
8522	RK11/RK05 SERVICE ROUTINE
8649	RH70/RP04 SERVICE ROUTINE
8748	RH70/RS04 SERVICE ROUTINE
8843	UNIBUS EXERCISER SERVICE ROUTINE
8922	MASS BUS TESTER SERVICE ROUTINE
8982	LINE CLOCK SERVICE ROUTINE
9006	SCOPE HANDLER ROUTINE
9065	ERROR HANDLER ROUTINE
9113	ERROR MESSAGE TYPEOUT ROUTINE
9395	TYPE ROUTINE
9469	ROUTINE TO TYPE THE ELAPSED RUN TIME OF THE PROGRAM
9529	ROUTINE TO TYPE THE AVAILABLE DEVICES AND UNIT NUMBERS
9564	BINARY TO OCTAL (ASCII) AND TYPE
9642	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
9710	DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
9753	SAVE AND RESTORE R0-R5 ROUTINES
9798	CONVERT FLOATING BINARY TO OCTAL ASCII
9871	CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCII
9925	RANDOM NUMBER GENERATOR ROUTINE
9959	FLOATING POINT NUMBER GENERATOR
9994	FLOATING POINT EXPONENT EXTENSION
10053	POWER DOWN AND UP ROUTINES
10095	TTY INPUT ROUTINE
10144	READ A DECIMAL NUMBER FROM THE TTY
10205	ROUTINE TO SIZE MEMORY
10300	TRAP DECODER
10315	TRAP TABLE
10336	UNIBUS EXERCISER INITIALIZATION ROUTINE
10361	CONVERT UNIBUS VIRTUAL ADDRESS TO PHYSICAL ADDRESS
10390	CONVERT A VIRTUAL ADDRESS TO A PHYSICAL ADDRESS
10431	ROUTINE TO CHECK RELOCATED DATA
10473	ROUTINE TO GET A MAP REGISTER
10550	GIVE MAP SUBROUTINE
10561	ROUTINE TO CLEAR 'T' BIT
10568	ROUTINE TO RESTORE THE T BIT
10579	KEYBOARD INT SERV ROUTINE
10646	TELETYPE INTERRUPT SERVICE ROUTINE
10659	PARITY ERROR SERVICE
10715	CONTEXT SWITCH DOWN SUBROUTINE
10743	CONTEXT SWITCH UP SUBROUTINE
10763	KT ABORT SUBROUTINE
10779	RESERVED INSTRUCTION ROUTINE
10792	TRAP TO 4 SERVICE ROUTINE

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

.TITLE DEQKC-D PDP 11/70-74MP CPU EXERCISER
:*COPYRIGHT (C) 1975, 1978
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY DONALD W. MONROE
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-A5-1).
:*

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70

```
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.* SWITCH USE
.* -----
.* 15 HALT ON ERROR
.* 14 LOOP ON TEST
.* 13 INHIBIT ERROR TYPEOUTS
.* 12 INHIBIT UBE
.* 11 INHIBIT ITERATIONS
.* 10 BELL ON ERROR
.* 9 LOOP ON ERROR
.* 8 ALLOW 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
.* 7...NOT USED
.*
```

```
71  
72 .SBTTL BASIC DEFINITIONS  
73  
74 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***  
75 001200 STACK= 1200 ;:FIRST ADDRESS OF THE STACK  
76 001200 KERSTK= STACK ;:KERNEL STACK  
77 000700 SUPSTK= STACK-300 ;:SUPERVISOR STACK  
78 000600 USESTK= STACK-400 ;:USER STACK  
79 .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL  
80 .EQUIV IOI,SCOPE ;:BASIC DEFINITION OF SCOPE CALL  
81 177776 PS= 177776 ;:PROCESSOR STATUS WORD  
82 .EQUIV PS,PSW  
83 177774 STKLMT= 177774 ;:STACK LIMIT REGISTER  
84 177772 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER  
85 177570 SWR= 177570 ;:SWITCH REGISTER  
86 177570 DISPLAY=SWR  
87  
88 ;*MISCELLANEOUS DEFINITIONS  
89 000011 HT= 11 ;:CODE FOR HORIZONTAL TAB  
90 000012 LF= 12 ;:CODE LINE FEED  
91 000015 CR= 15 ;:CODE CARRIAGE RETURN  
92 000200 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED  
93  
94 ;*GENERAL PURPOSE REGISTER DEFINITIONS  
95 000000 R0= %0 ;:GENERAL REGISTER  
96 000001 R1= %1 ;:GENERAL REGISTER  
97 000002 R2= %2 ;:GENERAL REGISTER  
98 000003 R3= %3 ;:GENERAL REGISTER  
99 000004 R4= %4 ;:GENERAL REGISTER  
100 000005 R5= %5 ;:GENERAL REGISTER  
101 000006 R6= %6 ;:GENERAL REGISTER  
102 000007 R7= %7 ;:GENERAL REGISTER  
103 .EQUIV R0,R10 ;:GENERAL REGISTER  
104 .EQUIV R1,R11 ;:GENERAL REGISTER  
105 .EQUIV R2,R12 ;:GENERAL REGISTER  
106 .EQUIV R3,R13 ;:GENERAL REGISTER  
107 .EQUIV R4,R14 ;:GENERAL REGISTER  
108 .EQUIV R5,R15 ;:GENERAL REGISTER  
109 000006 SP=%6 ;:KERNEL STACK POINTER  
110 .EQUIV SP,KSP ;:SUPERVISOR STACK POINTER  
111 .EQUIV SP,SSP ;:USER STACK POINTER  
112 .EQUIV SP,USP  
113 000007 PC %7  
114  
115 ;*PRIORITY LEVEL DEFINITIONS  
116 000000 PR0= 0 ;:PRIORITY LEVEL 0  
117 000040 PR1= 40 ;:PRIORITY LEVEL 1  
118 000100 PR2= 100 ;:PRIORITY LEVEL 2  
119 000140 PR3= 140 ;:PRIORITY LEVEL 3  
120 000200 PR4= 200 ;:PRIORITY LEVEL 4  
121 000240 PR5= 240 ;:PRIORITY LEVEL 5  
122 000300 PR6= 300 ;:PRIORITY LEVEL 6  
123 000340 PR7= 340 ;:PRIORITY LEVEL 7  
124  
125 ;*'SWITCH REGISTER' SWITCH DEFINITIONS  
126 100000 SW15 100000
```

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
ERRVEC- 4 ;:TIME OUT AND OTHER ERRORS

182 000004

```
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
212      ;;TO GET TO THE LAST 32 WORDS OF MEMORY
213      177762      SIZEHI = 177762   ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
214      ;;CURRENTLY ALL ZERO
215      177764      SYSTID = 177764   ;;SYSTEM ID REGISTER
216      177766      CPUERR = 177766   ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
217      ;;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

```
237      177600      UIPDR0= 177600
```

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' PAGL DESCRIPTOR REGISTORS

249	177620	UDPDR0= 177620
250	177622	UDPDR1= 177622
251	177624	UDPDR2= 177624
252	177626	UDPDR3= 177626
253	177630	UDPDR4= 177630
254	177632	UDPDR5= 177632
255	177634	UDPDR6= 177634
256	177636	UDPDR7= 177636

;*USER 'I' PAGE ADDRESS REGISTERS

260	177640	UIPAR0= 177640
261	177642	UIPAR1= 177642
262	177644	UIPAR2= 177644
263	177646	UIPAR3= 177646
264	177650	UIPAR4= 177650
265	177652	UIPAR5= 177652
266	177654	UIPAR6= 177654
267	177656	UIPAR7= 177656

;*USER 'D' PAGE ADDRESS REGISTERS

271	177660	UDPAR0= 177660
272	177662	UDPAR1= 177662
273	177664	UDPAR2= 177664
274	177666	UDPAR3= 177666
275	177670	UDPAR4= 177670
276	177672	UDPAR5= 177672
277	177674	UDPAR6= 177674
278	177676	UDPAR7= 177676

;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS

282	172200	SIPDR0= 172200
283	172202	SIPDR1= 172202
284	172204	SIPDR2= 172204
285	172206	SIPDR3= 172206
286	172210	SIPDR4= 172210
287	172212	SIPDR5= 172212
288	172214	SIPDR6= 172214
289	172216	SIPDR7= 172216

;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS

293	172220	SDPDR0= 172220
294	172222	SDPDR1= 172222

295	172224	SDPDR2= 172224
296	172226	SDPDR3= 172226
297	172230	SDPDR4= 172230
298	172232	SDPDR5= 172232
299	172234	SDPDR6= 172234
300	172236	SDPDR7= 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
350	172344	KIPAR2= 172344

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
359 172360
360 172362
361 172364
362 172366
363 172370
364 172372
365 172374
366 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'

377
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
406 170270

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
MAPL16 = 170270

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			
462	000000	AC0=	%0

463 000001
464 000002
465 000003
466 000004
467 000005
468
469 172540
470 172542
471 000104

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

;LINE CLOCK AND PROGRAMMABLE LINE CLOCK REGISTERS
PLKCSR=172540
PLKCSB=172542
PLKVEC=104

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 010000
507 002000
508 001000
509 030400
510 000200
511
512
513
514
515 000010
516 000000
517 140000
518 000000
519 030000
520 177770
521
522
523
524 076020
525 076061
526 076601
527 006600

LKS=177546
LKVEC=100
;UNIBUS EXERCISER REGISTER
UBEDB= 170000 ;DATA BUFFER
UBECC= 170002 ;CYCLE COUNT
UBEBA= 170004 ;BUS ADDRESS
UBECR1= 170006 ;CONTROL REGISTER 1
UBECLR= 170010 ;ERROR CLEAR
UBEGO= 170014 ;MULTI-EXERCISER GO
UBECR2= 170016 ;CONTROL REGISTER 2
UBEVEC= 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
CISOPT= 010000 ;1174 CIS OPTION PRESENT BIT
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
;OPCODES USED IN 1174 CISP TESTS
L2D0= 076020 ;LOAD 2 DESCRIPTORS @R0 OPCODE
L3D1= 076061 ;LOAD 3 DESCRIPTORS @R1 OPCODE
MED74C= 076601 ;CISP DIAGNOSTIC ENTRY OPCODE
CISTST= 6600 ;ADDRESS OF A U-DIAGNOSTIC INSTRUCTION

DEQKC-D PDP 11/70-74MP CPU EXERCISER
CEQKCD.P11 04-OCT-79 08:55.

MACY11 30A(1052) 04-OCT-79^{J 3} 09:00 PAGE 13
UNIBUS MAP REGISTER DEFINITIONS

SEQ 0035

528

000007

MFPT=7

;OPCODE FOR MFPT INSTRUCTION USED FOR 1174 ONLY

.SBTTL CIS OP CODE DEFINITIONS

529			
530			
531	076021	L2D1	=076021
532	076022	L2D2	=076022
533	076023	L2D3	=076023
534	076024	L2D4	=076024
535	076025	L2D5	=076025
536	076026	L2D6	=076026
537	076027	L2D7	=076027
538	076030	MOV	=076030
539	076031	MOVRC	=076031
540	076032	MOVTC	=076032
541	076040	LOCC	=076040
542	076041	SKPC	=076041
543	076042	SCANC	=076042
544	076043	SPANC	=076043
545	076044	CMPC	=076044
546	076045	MATC	=076045
547	076050	ADDN	=076050
548	076051	SUBN	=076051
549	076052	CMPN	=076052
550	076053	CVTNL	=076053
551	076054	CVTPN	=076054
552	076055	CVTNP	=076055
553	076056	ASHN	=076056
554	076057	CVTLN	=076057
555	076060	L3D0	=076060
556	076062	L3D2	=076062
557	076063	L3D3	=076063
558	076064	L3D4	=076064
559	076065	L3D5	=076065
560	076066	L3D6	=076066
561	076067	L3D7	=076067
562	076070	ADDP	=076070
563	076071	SUBP	=076071
564	076072	CMPP	=076072
565	076073	CVTPL	=076073
566	076074	MULP	=076074
567	076075	DIVP	=076075
568	076076	ASHP	=076076
569	076077	CVTLP	=076077
570	076130	MOVCI	=076130
571	076131	MOVRCI	=076131
572	076132	MOVTCI	=076132
573	076140	LOCCI	=076140
574	076141	SKPCI	=076141
575	076142	SCANCI	=076142
576	076143	SPANCI	=076143
577	076144	CMPCI	=076144
578	076145	MATCI	=076145
579	076150	ADDNI	=076150
580	076151	SUBNI	=076151
581	076152	CMJNI	=076152
582	076153	CVTNLI	=076153
583	076154	CVTPNI	=076154
584	076155	CVTNPI	=076155

585 076156
 586 076157
 587 076170
 588 076171
 589 076172
 590 076173
 591 076174
 592 076175
 593 076176
 594 076177
 595 076600

ASHNI =076156
 CVTLNI =076157
 ADDPI =076170
 SUBPI =076171
 CMPPPI =076172
 CVTPLI =076173
 MULPI =076174
 DIVPI =076175
 ASHPI =076176
 CVTLPI =076177
 MED6X =076600

.SBTTL TRAP CATCHER

600 000000

. =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

.SBTTL STARTING ADDRESS(ES)

606 000200

. =200

608 000200 000137 003542

JMP @#START ;: JUMP TO STARTING ADDRESS OF PROGRAM

609 000210 000210

. -210

JMP @#START1

610 000210 000137 002544

JMP @#START2

611 000214 000137 002554

::**26-APR-78,G.W.**

613 000220 000137 003330

JMP @#START3 ;: ENTRY FOR PID REG. CUTTING AID

614

::*****
 ;:*****

615

616

.SBTTL ACT11 HOOKS

617

::*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11

618

619

::*
 ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL
 ;*END OF THE PROGRAM.

620

621

::*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS

622

623

::*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS

624

::*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:

625

626

::* BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING

627

::* =0 NO POWER FAIL DESIRED

628

629

::* BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT

630

::* =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT

631

632

::* BITS 13-0 MUST BE ZERO'S

634

635 000224

.\$SVPC=. ;: SAVE LOCATION COUNTER

636 000046

. -46 ;: SET LOCATION COUNTER

637 000046 046570

.WORD \$ENDAD ;: SET LOC.46 TO ADDRESS \$ENDAD

638 000052

. =52 ;: SET LOCATION COUNTER

639 000052 040000

.WORD 40000 ;: SET LOC.52 TO 40000

640 000224

.=\$SVPC ;: RESTORE LOCATION COUNTER

DEQKC-D PDP 11/70-74MP CPU EXERCISER
CEQKCD.P11 04-OCT-79 08:55

M 3
MACY11 30A(1052) 04-OCT-79 09:00 PAGE 16
ACT11 HOOKS

SEQ 0038

641


```
642 ;:*****
643
644 .SBTTL COMMON TAGS
645
646 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
647 ;*USED IN THE PROGRAM.
648
649 001200 . =1200
650
651 001200 $CMTAG: ;:START OF COMMON TAGS
652 001200 000000 $PASS: .WORD 0 ;:CONTAINS PASS COUNT
653 001202 000000 $STSTM: .WORD 0 ;:CONTAINS THE TEST NUMBER
654 001204 000 $SERFLG: .BYTE 0 ;:CONTAINS ERROR FLAG
655 001206 .EVEN
656 001206 000000 $ICNT: .WORD 0 ;:CONTAINS SUBTEST ITERATION COUNT
657 001210 000000 $LPADR: .WORD 0 ;:CONTAINS SCOPE LOOP 1200
658 001212 000000 $LPERR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS
659 001214 000000 $ERTTL: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED
660 001216 000 $ITEMB: .BYTE 0 ;:CONTAINS ITEM CONTROL BYTE
661 001217 001 $ERMAX: .BYTE 1 ;:CONTAINS MAX. ERRORS PER TEST
662 001220 000000 $ERRPC: .WORD 0 ;:CONTAINS PC OF LAST ERROR INSTRUCTION
663 001222 000000 $GDADR: .WORD 0 ;:CONTAINS 1200 OF 'GOOD' DATA
664 001224 000000 $BDADR: .WORD 0 ;:CONTAINS 1200 OF 'BAD' DATA
665 001226 000000 $GDDAT: .WORD 0 ;:CONTAINS 'GOOD' DATA
666 001230 000000 $BDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA
667 001232 000000 000000 000000 .WORD 0,0,0 ;:RESERVED--NOT TO BE USED
668 001240 177560 $TKS: 177560 ;:TTY KBD STATUS
669 001242 177562 $TKB: 177562 ;:TTY KBD BUFFER
670 001244 177564 $TPS: 177564 ;:TTY PRINTER STATUS REG. 1200
671 001246 177566 $TPB: 177566 ;:TTY PRINTER BUFFER REG. 1200
672 001250 000 $NULL: .BYTE 0 ;:CONTAINS NULL CHARACTER FOR FILLS
673 001251 002 $FILLS: .BYTE 2 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
674 001252 012 $FILLC: .BYTE 12 ;:INSERT FILL CHARS. AFTER A 'LINE FFED'
675 001253 000 $TPFLG: .BYTE 0 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
676 001254 000000 $REGAD: .WORD 0 ;:CONTAINS THE 1200 FROM
677 ;:WHICH ($REGO) WAS OBTAINED
678 001256 000000 $REG0: .WORD 0 ;:CONTAINS (($REGAD)+0)
679 001260 000000 $REG1: .WORD 0 ;:CONTAINS (($REGAD)+2)
680 001262 000000 $REG2: .WORD 0 ;:CONTAINS (($REGAD)+4)
681 001264 000000 $REG3: .WORD 0 ;:CONTAINS (($REGAD)+6)
682 001266 000000 $REG4: .WORD 0 ;:CONTAINS (($REGAD)+10)
683 001270 000000 $REG5: .WORD 0 ;:CONTAINS (($REGAD)+12)
684 001272 000000 $REG6: .WORD 0 ;:CONTAINS (($REGAD)+14)
685 001274 000000 $REG7: .WORD 0 ;:CONTAINS (($REGAD)+16)
686 001276 000000 $REG10: .WORD 0 ;:CONTAINS (($REGAD)+20)
687 001300 000000 $REG11: .WORD 0 ;:CONTAINS (($REGAD)+22)
688 001302 000000 $TMP0: .WORD 0 ;:USER DEFINED
689 001304 000000 $TMP1: .WORD 0 ;:USER DEFINED
690 001306 000000 $TMP2: .WORD 0 ;:USER DEFINED
691 001310 000000 $TMP3: .WORD 0 ;:USER DEFINED
692 001312 000000 $TMP4: .WORD 0 ;:USER DEFINED
693 001314 000000 $TMP5: .WORD 0 ;:USER DEFINED
694 001316 000000 $TMP6: .WORD 0 ;:USER DEFINED
695 001320 000000 $TMP7: .WORD 0 ;:USER DEFINED
696 001322 000000 $TMP10: .WORD 0 ;:USER DEFINED
697 001324 000000 $TMP11: .WORD 0 ;:USER DEFINED
```

698	001326	000000		\$TIMES:	0		::MAX. NUMBER OF ITERATIONS
699	001330	000000		\$ESCAPE:	0		::ESCAPE ON ERROR 1200
700	001332	177607	000377	\$BELL:	.ASCII	<207><377><377>	::CODE FOR BELL
701	001336	077		\$QUES:	.ASCII	/?/	::QUESTION MARK
702	001337	015		\$CRLF:	.ASCII	<15>	::CARRIAGE RETURN
703	001340	000012		\$LF:	.ASCII	<12>	::LINE FEED
704	001342	000000		ERRRTN:	.WORD		
705	001344	000044		\$FLBUFF:	.B	KB 44	::BUFFER FOR FLOATING POINT CONVERSION
706	001410	000000		\$BUFF:	.WORD		
707	001412	000000		\$ACO:	.WORD		::EXTENDED EXPONENT VALUES
708	001414	000000		\$AC1:	.WORD		::FOR THE SIX FLOATING POINT
709	001416	000000		\$AC2:	.WORD		::ACCUMULATORS
710	001420	000000		\$AC3:	.WORD		
711	001422	000000		\$AC4:	.WORD		
712	001424	000000		\$AC5:	.WORD		
713	001426	000000		\$STMP4:	.WORD		
714	001430	000000		\$STMP6:	.WORD		
715	001432	000004		FLTMP0:	.BLKW	4	::FLOATING POINT DBL PREC BUFFER
716	001442	000004		FLTMP1:	.BLKW	4	
717	001452	001454		TKBFRP:	.WORD	TKBFR	::POINTER FOR KEYBOARD BUFFER
718	001454	000011		TKBFR:	.BLKW	11	::KEYBOARD BUFFER
719	001476	000000		NOTYPE:	.WORD		::NO TYPEOUT FLAG (INHIBIT WHEN SET)
720	001500	000000		OPT.OP:	.WORD		::CPU OPTION FLAGS
721	001502	000		KB11E:	.BYTE	0	::WITHOUT MP CACHE
722	001503	000		KB11EM:	.BYTE	0	::WITH MP CACHE
723	001504	000		KB11CM:	.BYTE	0	::KB11CM FLAG (1170 WITH MP MODS)
724	001505	000		CISP:	.BYTE	0	::CISP OPTION PRESENT FLAG
725	001506	000004		\$SAVPAR:	.BLKW	4	::USED BY INTERRUPT SERVICE ROUTINE
726	001516	000000		\$SAVPSW:	.WORD		::DITTO
727	001520	000006		\$RTRN:	RTT		::RETURN FOR T-BIT TRAP
728	001522	000000		VADR:	.WORD		::BUFFER FOR VIRTUAL ADDRESS
729	001524	000000		PA1500:	.WORD		::BUFFER FOR PHYSICAL ADDRESS BITS<15:00>
730	001526	000000		PA2116:	.WORD		::PHYSICAL ADDRESS BITS<21:16>
731	001530	000		NEXEC:	.BYTE		::NO EXECUTE FLAG(NO TEST EXECUTION WHEN SET)
732	001531	000		MMON:	.BYTE		::MEMORY MGMT FLAG(MGMT IS ON WHEN NON-ZERO)
733	001532	000		QV:	.BYTE		::QV FLAG(QV PASS WHEN SET)
734	001533	000		AA:	.BYTE		::AUTO ACCEPT FLAG (AA PASS WHEN SET)
735	001534	000000		FACTOR:	.WORD		::RELOCATION FACTOR(NUMBER OF
736	001536	000000		\$FACTOR:	.WORD		::BYTES ABOVE BASE CODE)
737	001540	000000		FRSTAD:	.WORD		::FIRST ADDRESS OF SECTION BEING EXECUTED
738	001542	000000		FRSTMEM:	.WORD		::ADDRESS OF FIRST FREE MEMORY
739	001544	000000		LSTMEM:	.WORD		::ADDRESS OF LAST FREE MEMORY(IN 28K)
740	001546	000000		NEXPAR:	.WORD		::NEXT VALUE TO PUT IN PAR0
741	001550	123456		\$LONUM:	.WORD	123456	::LOW 16 BITS OF RANDOM NUMBER
742	001552	065432		\$HINUM:	.WORD	65432	::HIGH 16 BITS OF RANDOM NUMBER
743	001554	377	377	NULLS:	.BYTE	377,377,377,0	::BUFFER FOR PRINTER TEST
744	001557	000					
745	001560	000060		SUBPASS:	.WORD	60	::SUB-PASS COUNT IN ASCII
746	001562	000000		\$ERPSW:	.WORD		::ERROR PSW FOR TYPEOUT
747	001564	000000		EXITFL:	.WORD		
748	001566	000000		OLDBASE:	.WORD		::SOURCE BASE ADDRESS FOR DEVICE RELOCATION
749	001570	000000		NWBASL:	.WORD		::DEST ADDRESS FOR DEVICE RELOC BITS<15:00>
750	001572	000000		NWBASH:	.WORD		::DEST ADDRESS FOR DEVICE RELOC BITS<21:16>
751	001574	000000		IOWC:	.WORD		::TWO'S COMPLIMENT WORD COUNT FOR DEVICE RELOC
752	001576	000000		DEVICE:	.WORD		
753	001600	000000		DEVINDX:	.WORD		::DEVICE INDEX (0 TO 7)

```
754 001602 000000 UNITNO: .WORD ;DEVICE UNIT NUMBER
755 001604 000000 RNTBINX: .WORD ;INDEX TO RUN TABLE
756 001606 000000 MXMMHI: .WORD ;BITS<21:16> OF LAST MEM ADDRESS ON SYSTEM
757 001610 000000 MXMML0: .WORD ;BITS<15:00> OF LAST MEM ADDRESS ON SYSTEM
758 001612 000000 RP310: .WORD ;DATA TO LOAD INTO RP03 CS REGISTER
759 001614 000000 RP311: .WORD ;RP03 FLAG FOR FIRST 2K OF PROGRAM
760 001616 000000 RK10: .WORD ;DATA TO LOAD INTO RK05 CS REGISTER
761 001620 000000 RK11: .WORD ;RK05 FLAG FOR FIRST 2K OF PROGRAM
762 001622 000000 RP411: .WORD ;RP04 FLAG FOR FIRST 2K OF PROGRAM
763 001624 000000 RS11: .WORD ;RS04 FLAG FOR FIRST 2K OF PROGRAM
764 001626 000000 MTICKS: .WORD ;ELAPSED RUN TIME IN MINUTES
765 001630 000000 LTICKS: .WORD ;LOW BYTE=NUMBER OF CLOCK INTERRUPTS (0 TO 59)
766 ;HIGH BYTE=ELAPSED RUN TIME IN SECONDS(0 TO 59)
767 001632 000000 LD2PNT: .WORD 0 ;NEXT 3 WORDS USED FOR CISP DETECTION
768 001634 000000 LD2PT1: .WORD 0
769 001636 000000 LD3PNT: .WORD 0
770 001640 000000 SMAINT: .WORD ;CURRENT VALUE IN MAINTENANCE REGISTER
771 001642 000010 SYSSIZE: .BLKW 10 ;SYSTEM SIZE TABLE(ONE ENTRY FOR EACH DEVICE)
772 001662 000007 RUNTBL: .BLKW 7 ;RUN TIME TABLE(ONE ENTRY FOR EACH 2K BLOCK)
773 001700 000007 RUNTRAK: .BLKW 7 ;RUN TRACK TABLE(ONE ENTRY FOR EACH 2K BLOCK)
774 001716 177777 MAP1BL: .WORD -1 ;MAP TABLE(ONE BYTE FOR EACH UNIBUS DEVICE)
775 001720 177777 ;UNUSED=377, USED=LOW 5 BITS OF MAP ADDRESS
776 001722 000002 UBESAV: .BLKW 2 ;BASE ADDRESS OF UBE TRANSFER IN PROGRESS
777 001726 000002 UBEADR: .BLKW 2 ;ADDRESS THAT GETS LOADED INTO UBE BA REG
778 001732 000002 ERRBA: .BLKW 2 ;18 BIT UNIBUS ADDRESS WHEN DEVICE DETECTED AN ERROR
779 .SBTTL DEVICE HANDLER STATUS WORDS
780 ;* EACH WORD HAS THE FOLLOWING BIT ASSIGNMENTS:
781 ;* 7 HANDLER READY
782 ;* 8 REPEAT LAST FUNCTION
783 ;* 15 ERROR
784 001736 000200 RP3HSTAT: .WORD 200 ;RP03
785 001740 000200 RKHSTAT: .WORD 200 ;RK05
786 001742 000200 SPARE0: .WORD 200
787 001744 000200 SPARE1: .WORD 200
788 001746 000200 RP4HSTAT: .WORD 200 ;RP04
789 001750 000200 RSHSTAT: .WORD 200 ;RS04
790 ;.WORD 200 ;SPARE
791 001754 000200 ;.WORD 200 ;SPARE
792
793 .SBTTL DEVICE HANDLER WORD COUNTS
794 ;* THIS TABLE GETS LOADED BY THE I/O
795 ;* RELOCATION ROUTINE WITH THE TWO'S COMPLIMENT WORD
796 ;* COUNT FOR THE TRANSFER FOR THE PARTICULAR DEVICE.
797 001756 000000 RP3HWC: .WORD ;RP03
798 001760 000000 RKHWC: .WORD ;RK05
799 001762 000000 ;.WORD ;SPARE
800 ;.WORD ;SPARE
801 001766 000000 RP4HWC: .WORD ;RP04
802 001770 000000 RSHWC: .WORD ;RS04
803
804 .SBTTL DEVICE HANDLER OLD BASE ADDRESS
805 ;* THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE
806 ;* WITH THE BASE ADDRESS OF THE SOURCE DATA FOR THE
807 ;* DEVICE THAT IS GOING TO TRANSFER THE DATA.
808 001772 000000 RP3OLD: .WORD ;RP03
809 001774 000000 ;.WORD
```

810 001776 000000
811 002000 000000
812 002002 000000
813 002004 000000
814 002006 000000
815 002010 000000
816 002012 000000
817 002014 000000
818 002016 000000
819 002020 000000

RKOLD: .WORD ;RK05
 .WORD ;SPARE
 .WORD ;SPARE
 .WORD ;SPARE
RP4OLD: .WORD ;RP04
 .WORD ;RS04
RSOLD: .WORD
 .WORD

820
821
822
823
824
825 002022 000000
826 002024 000000
827 002026 000000
828 002030 000000
829 002032 000000
830 002034 000000
831 002036 000000
832 002040 000000
833 002042 000000
834 002044 000000
835 002046 000000
836 002050 000000

.SBTTL DEVICE HANDLER NEW BASE ADDRESSES
;* THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE
;* WITH THE BASE ADDRESS OF THE DESTINATION FOR THE
;* PARTICULAR DEVICE THAT IS GOING TO DO THE TRANSFER.
RP3NWL: .WORD ;RP03
RP3NWH: .WORD
RKNEWL: .WORD ;RK05
RKNEWH: .WORD ;SPARE
 .WORD ;SPARE
 .WORD ;SPARE
RP4NWL: .WORD ;RP04
RP4NWH: .WORD
RSNEWL: .WORD ;RS04
RSNEWH: .WORD

837
838
839
840
841
842 002052 000000
843 002054 000000
844 002056 000000
845 002060 000000
846 002062 000000
847 002064 000000

.SBTTL DEVICE HANDLER UNIT NUMBER
;* THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE.
;* IT TELLS THE DEVICE HANDLER WHICH UNIT NUMBER IS
;* TO DO THE TRANSFER.
RP3UNIT: .WORD ;RP03
RKUNIT: .WORD ;RK05
 .WORD ;SPARE
 .WORD ;SPARE
RP4UNIT: .WORD ;RP04
RSUNIT: .WORD ;RS04

848
849
850
851
852
853 002066 046610
854 002070 047226
855 002072 000000
856 002074 000000
857 002076 047622
858 002100 050172

.SBTTL ADDRESS OF THE DEVICE HANDLERS
;* THIS TABLE CONTAINS THE ADDRESS OF THE DEVICE HANDLER
;* ROUTINES. IT IS USED BY THE I/O RELOCATION ROUTINE
;* TO TRANSFER CONTROL TO THE DEVICE HANDLER.
RP3HANA: .WORD RP3DRV ;RP03
RKHANA: .WORD RKDRV ;RK05
 .WORD ;SPARE
 .WORD ;SPARE
RP4HANA: .WORD RP4DRV ;RP04
RSHANA: .WORD RSDRV ;RS04

859
860
861
862
863
864 002102 000000
865 002104 000000

.SBTTL DEVICE HANDLER DISK ADDRESS TABLE
;* THIS TABLE GETS LOADED BY THE DEVICE HANDLER WITH THE
;* DISK ADDRESS(SECTOR AND CYLINDER) OF THE CURRENT
;* TRANSFER.
RP3HDA: .WORD ;RP03 DISK ADDRESS
RP3HDC: .WORD ;RP03 DESIRED CYLINDER

866 002106 000000
867 002110 000000
868 002112 000000
869 002114 000000
870 002116 000000
871
872
873
874
875
876 002120 000000
877 002122 000000
878 002124 000000
879 002126 000000
880 002130 000000
881
882
883
884
885
886
887
888 002132 000
889 002133 000
890 002134 000
891 002135 000
892 002136 000
893 002140
894
895
896
897
898
899
900
901
902
903 002140 176710
904 002142 176712
905 002144 176714
906 002146 176716
907 002150 176720
908 002152 176724
909 002154 176722
910 002156 000254
911 002160 000256
912
913
914 002162 177400
915 002164 177402
916 002166 177404
917 002170 177406
918 002172 177410
919 002174 177412
920 002176 000220
921 002200 000222

RKHDA: .WORD ;RK05 DISK ADDRESS
;WORD ;SPARE
RP4HDA: .WORD
RP4HDC: .WORD ;RP04 DESIRED CYLINDER
RSHDA: .WORD ;RS04 DISK ADDRESS
.SBTTL DEVICE HANDLER FUNCTION TABLE
:* THIS TABLE GETS LOADED BY THE DEVICE HANDLERS
:* AND THE DEVICE SERVICE ROUTINES. IT TELLS THE ROUTINES
:* WHICH FUNCTION TO DO NEXT.
RP3FUN: .WORD ;RP03
RKFUN: .WORD ;RK05
;WORD ;SPARE
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
.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
.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

```

922
923 .SBTTL RH70/RP04 REGISTERS
924 002202 176700 RP4CS1: .WORD 176700 ;CONTROL AND STATUS #1
925 002204 176702 RP4WC: .WORD 176702 ;WORD COUNT
926 002206 176704 RP4BA: .WORD 176704 ;BUS ADDRESS
927 002210 176750 RP4BAE: .WORD 176750 ;BUS ADDRESS EXTENDED
928 002212 176706 RP4DA: .WORD 176706 ;DISK ADDRESS
929 002214 176710 RP4CS2: .WORD 176710 ;CONTROL AND STATUS #2
930 002216 176752 RP4CS3: .WORD 176752 ;CONTROL AND STATUS #3
931 002220 176712 RP4DS: .WORD 176712 ;DRIVE STATUS
932 002222 176714 RP4ER1: .WORD 176714 ;ERROR REG #1
933 002224 176734 RP4DC: .WORD 176734 ;DESIRED CYLINDER
934 002226 176740 RP4ER2: .WORD 176740 ;ERROR REG #2
935 002230 176742 RP4ER3: .WORD 176742 ;ERROR REG #3
936 002232 176736 RPCC: .WORD 176736 ;CURRENT CYLINDER
937 002234 176732 RP4OF: .WORD 176732 ;OFFSET REGISTER
938 002236 000254 RP4VEC: .WORD 254 ;INTERRUPT VECTOR
939 002240 000256 RP4PSW: .WORD 256 ;INTERRUPT VECTOR+2
    
```

```

940
941 .SBTTL RH70/RS04 REGISTERS
942 002242 172040 RSCS1: .WORD 172040 ;CONTROL AND STATUS #1
943 002244 172042 RSWC: .WORD 172042 ;WORD COUNT
944 002246 172044 RSBA: .WORD 172044 ;BUS ADDRESS
945 002250 172070 RSBAE: .WORD 172070 ;BUS ADDRESS EXTENDED
946 002252 172046 RSDA: .WORD 172046 ;DISK ADDRESS
947 002254 172050 RSCS2: .WORD 172050 ;CONTROL AND STATUS #2
948 002256 172072 RSCS3: .WORD 172072 ;CONTROL AND STATUS #3
949 002260 172052 RSDS: .WORD 172052 ;DRIVE STATUS
950 002262 172054 RSER: .WORD 172054 ;ERROR REG
951 002264 000204 RSVEC: .WORD 204 ;INTERRUPT VECTOR
952 002266 000206 RSPSW: .WORD 206 ;INTERRUPT VECTOR+2
    
```

```

953
954 .SBTTL UNIBUS EXERCISER REGISTER ADDRESS TABLE
955 ;* THIS TABLE IS ASSEMBLED FOR UBE #0. IF THE UBE
956 ;* ADDRESSES ARE CUT FOR OTHER THAN UNIT #0, THE PROGRAM
957 ;* WILL CHANGE THIS TABLE. THE PROGRAM LOOKS FOR A
958 ;* UBE AT ADDRESSES 770002, 770022, 770032, AND 770042.
959 002270 170002 UBETBL: .WORD UBEC ;CYCLE COUNT
960 002272 170004 .WORD UBEB ;BUS ADDRESS REG
961 002274 170016 .WORD UBECR2 ;CONTROL REGISTER #2
962 002276 170006 .WORD UBECR1 ;CONTROL REGISTER #1
963 002300 170010 .WORD UBECLE ;UBE CLEAR ADDRESS
964 002302 000510 .WORD UBEVEC ;INTERRUPT VECTOR
965 002304 000512 .WORD UBEVEC+2 ;INTERRUPT VECTOR +2
    
```

```

966
967 .SBTTL MASS BUS TESTER REGISTER ADDRESSES
968 ;* THE PROGRAM IS ASSEMBLED WITH ADDRESSES FOR A MBT
969 ;* AT 770100. IF THE MBT IS AT ANOTHER ADDRESS THE PROGRAM
970 ;* WILL CHANGE THIS TABLE. THE PROGRAM LOOKS FOR A UBE
971 ;* AT ADDRESSES 770100, 770200, 770300, AND 770400.
972 002306 160100 MBTTBL: .WORD MBTCS1 ;CONTROL AND STATUS #1
973 002310 160102 .WORD MBTWC ;WORD COUNT
974 002312 160104 .WORD MBTBA ;BUS ADDRESS
975 002314 160174 .WORD MBTBAE ;BUS ADDRESS EXTENDED
976 002316 160106 .WORD MBTMR2 ;MAINTENANCE REGISTER #2
977 002320 160110 .WORD MBTCS2 ;CONTROL REGISTER #2
    
```

978	002322	160112	.WORD	MBTST	;STATUS REGISTER
979	002324	160114	.WORD	MSTER	;ERROR REGISTER
980	002326	160176	.WORD	MBTCS3	;CONTROL REGISTER #3
981	002330	000774	.WORD	MBTVEC	;INTERRUPT VECTOR
982	002332	000776	.WORD	MBTPSW	;INTERRUPT VECTOR+2
983	002334	160126	.WORD	MBTDT	;DRIVE TYPE REGISTER
984	002336	160200	MBTN2: .WORD	160200	;MASS BUS TESTER #2
985	002340	160300	MBTN3: .WORD	160300	;MASS BUS TESTER #3
986	002342	160400	MBTN4: .WORD	160400	;MASS BUS TESTER #4
987					

988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004 002344
1005
1006 002344 065505
1007 002346 065532
1008 002350 065576
1009 002352 065571
1010
1011 002354 065610
1012 002356 065636
1013 002360 065664
1014 002362 065571
1015
1016 002364 065674
1017 002366 065731
1018 002370 066000
1019 002372 065571
1020
1021 002374 066014
1022 002376 066043
1023 002400 066000
1024 002402 066121
1025
1026 002404 066126
1027 002406 066165
1028 002410 066236
1029 002412 066232
1030
1031 002414 066250
1032 002416 066320
1033 002420 066340
1034 002422 066121
1035
1036 002424 000000
1037 002426 000000
1038 002430 000000
1039 002432 000000
1040
1041 002434 066346
1042 002436 066417
1043 002440 066472

.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 (\$ERRKPC).
;*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 CPJERR
DT1 :VADR,VADR,\$TMP0,\$TMP2
DF1 :0,1,0,0,0
:ITEM 2
EM2 :UNEXPECTED TRAP TO 10
DH2 :PCOFTP PHYSPC PSW
DT2 :VADR,VADR,\$TMP0
DF1
:ITEM 3
EM3 :UNEXPECTED TRAP TO 250(MGMT)
DH3 :PCOFTP PHYSPC PSW MMR0 MMR2
DT3 :VADR,VADR,\$TMP0,, \$TMP2,, \$TMP3
DF1
:ITEM 4
EM4 :UNEXPECTED TRAP TO 114
DH4 :PCOFTP PHYSPC PSW ERADREG MEMERRREG
DT3 :VADR,VADR,\$TMP0,\$TMP3,\$TMP2
DF4 :0,1,0,2,0
:ITEM 5
EM5 :PARITY ERROR DURING DATA CHECK
DH5 :SRCADR DSTADR ERRADREG MEM ERR REG
DT5 :\$TMP0,PA1500,\$TMP3,\$TMP2
DF5
:ITEM 6
EM6 :ERROR DURING CHECK OF RELOCATED DATA
DH6 :SRCADR DSTADR
DT6 :\$TMP0,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
DT10 :\$TMP0,VADR,\$TMP2,\$TMP3

Line No	Code	Address	Pointer	Description
1044	002442	066466	DF10	:0,1,3,0
1045			:ITEM 11	
1046	002444	066504	EM11	:BIT(S) STUCK IN MICRO-BREAK REG
1047	002446	066551	DH11	:GOOD DAT BAD DAT
1048	002450	066574	DT11	:\$TMP0,\$TMP1
1049	002452	066572	DF11	:0,0
1050			:ITEM 12	
1051	002454	066602	EM12	:UNIBUS EXERCISER NON-EXISTANT MEMOREY
1052	002456	066640	DH12	:PHYSICAL ADDRESS
1053	002460	066656	DT12	:PA1500
1054	002462	066654	DF12	:2
1055			:ITEM 13	
1056	002464	066662	EM13	:MASS BUS TESTER NON-EXISTANT MEMORY
1057	002466	066720	DH13	:PHYSICAL ADDRESS
1058	002470	066656	DT12	
1059	002472	066654	DF12	
1060			:ITEM 14	
1061	002474	066735	EM14	:FLOATING POINT ERROR
1062	002476	066762	DH14	: DATA1 DATA2
1063	002500	067002	DT14	:\$TMP4,\$REG2,\$TMP6,\$REG3
1064	002502	067014	DF14	:4,0,4,0
1065			:ITEM 15	
1066	002504	067020	EM15	:DEVICE HUNG
1067	002506	000000	0	
1068	002510	000000	0	
1069	002512	000000	0	
1070			:ITEM 16	
1071	002514	066735	EM14	:FLOATING POINT ERROR
1072	002516	067034	DH16	
1073	002520	067066	DT16	:\$FLTMP0,\$REG2,\$FLTMP1,\$REG3
1074	002522	067061	DF16	:5,0,5,0
1075			:ITEM 17	
1076	002524	067100	EM17	:RO FAILED TO LOAD CORRECTLY ON MFPT
1077	002526	066551	DH11	:GOOD DAT BAD DAT
1078	002530	066574	DT11	:\$TMP0,\$TMP1
1079	002532	066572	DF11	:0,0
1080			:ITEM 20	
1081			EM20	:CIS INSTRUCTION FAILURE
1082	002534	067144	DH6	
1083	002536	066320	DT11	
1084	002540	066574	DF11	
1085	002542	066572	DF11	

```
1086 002544 013737 177570 177570 START1: MOV @#SWR,@#SWR
1087 002552 000774 BR START1
1088 .SBTTL PROGRAM INITIALIZATION
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105 002554 012706 001100 START2: MOV #1100,SP ;SETUP THE SP
1106 002560 012737 062144 000034 MOV #STRAP,@#TRAPVEC ;SETUP TRAP VECTOR
1107 002566 012737 054622 000030 MOV #SEERROR,@#EMTVEC;SETUP EMT VECTOR
1108 002574 012700 000377 MOV #377,R0 ;PUT MICRO-BREAK DATA IN R0
1109 002600 005737 001502 TST @#KB11E ;IS THIS A KB11-E OR KB11-EM PROCESSOR?
1110 002604 001402 BEQ 1$ ;BR IF NOT. 8 BIT U-BREAK REGISTER
1111 002606 012700 177777 MOV #177777,R0 ;KB11-E AND KB11-EM HAVE 16 BIT U-BREAK REGISTER
1112 002612 010037 177770 1$: MOV R0,@#UBREAK ;LOAD U BREAK REG
1113 002616 020037 177770 CMP R0,@#UBREAK ;LOAD OK?
1114 002622 001036 BNE UBRERR ;BRANCH IF NO
1115 002624 005000 CLR R0
1116 002626 010037 177770 MOV R0,@#UBREAK
1117 002632 020037 177770 CMP R0,@#UBREAK
1118 002636 001030 BNE UBRERR
1119 002640 012700 000125 MOV #125,R0
1120 002644 005737 001502 IST @#KB11E ;IS THIS A KB11-E OR KB11EM PROCESSOR?
1121 002650 001402 BEQ 2$ ;BR IF NOT. 8 BIT U-BREAK REGISTER
1122 002652 012700 052525 MOV #52525,R0 ;KB11-E AND KB11-EM HAVE 16 BIT U-BREAK REGISTER
1123 002656 010037 177770 2$: MOV R0,@#UBREAK
1124 002662 020037 177770 CMP R0,@#UBREAK
1125 002666 001014 BNE UBRERR
1126 002670 012700 000252 MOV #252,R0
1127 002674 005737 001502 TST @#KB11E ;IS THIS A KB11-E OR KB11-EM PROCESSOR?
1128 002700 001402 BEQ 3$ ;BR IF NOT. 8 BIT U-BREAK REGISTER
1129 002702 012700 125252 MOV #125252,R0 ;KB11-E AND KB11-EM HAVE 16 BIT U-BREAK REGISTER
1130 002706 010037 177770 3$: MOV R0,@#UBREAK
1131 002712 020037 177770 CMP R0,@#UBREAK
1132 002716 001411 BEQ UBRK2
1133 002720 010067 176356 UBRERR: MOV R0,$TMP0
1134 002724 013737 177770 00135: MOV @#UBREAK,@#$TMP1
1135 002732 012737 002554 001212 MOV #START2,@#$LPERR
1136 002740 104011 ERROR 11
1137 ;TEST TO ENSURE U BREAK COMPARATORS DO NOT COME ON.
1138 002742 012737 000100 177770 UBRK2: MOV #100,@#UBREAK ;PUT SAFE VALUE IN REG
1139 002750 104400 002756 TYPE ,65$ ;:TYPE ASCIZ STRING
1140 002754 000421 BR 64$ ;:GET OVER THE ASCIZ
1141 ;:65$: .ASCIZ /SET MAINT TO STOP ON MICRO-BREAK/<CRLF>
```

1142	003020				64\$:			
1143	003020	104400	003026			TYPE	,67\$::TYPE ASCIZ STRING
1144	003024	000407				BR	66\$::GET OVER THE ASCIZ
1145					::67\$:	.ASCIZ	/HIT CONTINUE/<CRLF>	
1146	003044				66\$:			
1147	003044	000000				HALT		
1148	003046	012737	000012	000010		MOV	#12,@#RESVEC	
1149	003054	012737	000002	000012		MOV	#2,@#RESVEC+2	
1150	003062	012705	003122			MOV	#2\$,R5	;SET UP R5 FOR MARK INSTR
1151	003066	012701	000010			MOV	#10,R1	;SET SOB COUNT
1152	003072	012702	003277			MOV	#UBRTBL+1,R2	;GET ADRS OF UBREAK DATA TABLE
1153	003076	112237	177770		1\$:	MOVB	(R2)+,@#UBREAK	;LOAD MICRO-BREAK FROM TABLE
1154	003102	000010				10		;EXEC RES INSTR (ROM ADRS 000)
1155	003104	005037	177770			CLR	@#UBREAK	
1156	003110	077106				SOB	R1,1\$;CONTINUE
1157	003112	012737	000125	177770		MOV	#125,@#UBREAK	;SET MICRO-BREAK DATA PATTERN
1158	003120	006400				MARK	0	;EXEC MARK (ROM ADRS 252)
1159	003122	005037	177770		2\$:	CLR	@#UBREAK	
1160	003126	012706	001100			MOV	#1100,SP	;RESTORE SP
1161	003132	012737	000006	000004		MOV	#6,@#ERRVEC	
1162	003140	012737	000002	000006		MOV	#2,@#ERRVEC+2	
1163	003146	052737	040000	177776		BIS	#BIT14,@#PSW	;GO TO SUPER MODE
1164	003154	012706	000700			MOV	#700,SP	;SET SUPER SP
1165	003160	012746	003202			MOV	#3\$,-(SP)	;SETUP STACK FOR JSR INSTR
1166	003164	005000				CLR	R0	;SETUP R0
1167	003166	012701	000007			MOV	#7,R1	;SET SOB COUNT
1168	003172	012702	003312			MOV	#INSTBL+2,R2	;GET ADRS OF TABLE OF INSTRUCTIONS
1169	003176	012217			4\$:	MOV	(R2)+,(PC)	;GET INSTRUCTION
1170	003200	000000				.WORD		;EXECUTE INSTRUCTION
1171	003202	077103			3\$:	SOB	R1,4\$;CONTINUE
1172	003204	012737	000100	177770		MOV	#100,@#UBREAK	;PUT SAFE VALUE IN UBREAK REG
1173	003212	005000				CLR	R0	
1174	003214	012702	003276			MOV	#UBRTBL,R2	
1175	003220	012703	003310			MOV	#INSTBL,R3	
1176	003224	012701	000010			MOV	#10,R1	
1177	003230	012746	003244			MOV	#5\$,-(SP)	
1178	003234	112237	177770		6\$:	MOVB	(R2)+,@#UBREAK	;LOAD UBREAK REG FROM TABLE
1179	003240	012317				MOV	(R3)+,(PC)	;GET INSTR FROM TABLE
1180	003242	000000				.WORD		;EXECUTE INSTR. PROCESSOR SHOULD STOP
1181								;WITH THE CORRECT ROM ADR IN THE LIGHTS
1182	003244	077105			5\$:	SOB	R1,6\$;CONTINUE
1183	003246	111237	177770			MOVB	(R2),@#UBREAK	;PUT SAFE VALUE IN UBREAK REG
1184	003252	005037	177776			CLR	@#PSW	;GO BACK TO KERNEL MODE
1185	003256	104400	003264			TYPE	,69\$::TYPE ASCIZ STRING
1186	003262	000403				BR	68\$::GET OVER THE ASCIZ
1187					::69\$:	.ASCIZ	/DONE/<CRLF>	
1188	003272				68\$:			
1189	003272	000000				HALT		
1190	003274	000522				BR	START	
1191	003276	000	001	002	UBRTBL:	.BYTE	0,1,2,4,10,20,40,200,100	
1192	003301	004	010	020				
1193	003304	040	200	100				
1194		003310				.EVEN	-	
1195	003310	000010	005010	005020	INSTBL:	.WORD	10,5010,5020,5040,0,5200,207,5010	
1196	003316	005040	000000	005200				
1197	003324	000207	005010					

```
1198 003330 012706 001100 START3: MOV #1100,SP ;SET UP STACK
1199 003334 012737 062144 000034 MOV #STRAP,@#TRAPVEC ;SET UP TRAP VECTOR
1200 003342 104400 003350 TYPE ,65$ ;;TYPE ASCIZ STRING
1201 003346 000415 BR 64$ ;;GET OVER THE ASCIZ
1202 ;;65$: .ASCIZ <15><12>/PID REGISTER SETUP AID/
1203 U03402 64$:
1204 003402 104400 003410 TYPE ,67$ ;;TYPE ASCIZ STRING
1205 003406 000430 BR 66$ ;;GET OVER THE ASCIZ
1206 ;;67$: .ASCIZ <15><12>/TYPE IN THE DESIRED PROCESSOR SERIAL NUMBER: /
1207 003470 66$:
1208 003470 104416 RDDEC ;GET THE NUMBER.
1209 003472 104400 003500 TYPE ,69$ ;;TYPE ASCIZ STRING
1210 003476 000417 BR 68$ ;;GET OVER THE ASCIZ
1211 ;;69$: .ASCIZ <15><12>/THE OCTAL EQUIVALENT IS : /
1212 003536 68$:
1213 003536 104402 TYPOC ;TYPE THE NUMBER IN OCTAL
1214 003540 000673 BR START3
1215
1216 ;;*****
1217
1218 003542 012706 001200 START: MOV #KERSTK,SP ;SET KERNEL STACK PTR
1219 003546 012737 076543 001552 MOV #76543,@#SHINUM ;INITIALIZE RANDOM NUM GEN
1220 003554 012737 123456 001550 MOV #123456,@#LONUM
1221
1222 ;DETERMINE HOW PROGRAM WAS LOADED AND WHAT MODE (IF ACT11)
1223 ;AND SET MEMORY PROTECTION.
1224 003562 005037 001532 CLR @#QV ;SET NOT QV NOR AA MODE
1225 003566 005027 CLR (PC)+ ;SET NOT XXDP
1226 003570 000 XXDP: .BYTE 0 ;XXDP INDICATOR
1227 003571 000 XXDPC: .BYTE 0 ;XXDP CHAIN MODE INDICATOR
1228 003572 005027 CLR (PC)+ ;CLEAR MEMORY PROTECTION LIMIT
1229 003574 000000 PROT: .WORD 0 ;WILL CONTAIN MEM PROT LIMIT
1230 003576 005737 046574 TST @#SENDAD+4 ;BRANCH IF NOT QV
1231 003602 100003 BPL 1$
1232 003604 110637 001532 MOVB SP,@#QV ;SET ACT11 QV MODE
1233 003610 000411 BR 3$
1234
1235 003612 001003 1$: BNE 2$
1236 003614 110637 001533 MOVB SP,@#AA ;SET ACT11 AA MODE
1237 003620 000405 BR 3$
1238
1239 003622 005737 000042 2$: TST @#42 ;BRANCH IF NOT IN CHAIN MODE
1240 003626 001402 BEQ 3$
1241 003630 110637 003571 MOVB SP,@#XXDPC ;SET CHAIN MODE INDICATOR
1242
1243 ;SET MEMORY PROTECTION LIMITS
1244 003634 005737 001532 3$: TST @#QV ;BRANCH IF QV OR AA
1245 003640 001006 BNE MEMSIZ
1246 003642 005737 003570 TST @#XXDP ;BRANCH IF NOT VIA XXDP
1247 003646 001403 BEQ MEMSIZ
1248 003650 012737 005700 003574 MOV #5700,@#PROT ;PROTECT XXDP MONITOR
1249 003656 012737 157776 001544 MEMSIZ: MOV #157776,@#LSTMEM ;SET VALUE INTO LSTMEM
1250 003664 163737 003574 001544 SUB @#PROT,@#LSTMEM ;SET PROTECTION
1251 003672 012737 067176 001542 MOV #ENDTAG+2,@#FRSTMEM ;SET FIRST RELOCATION ADDRESS
1252
1253 ;GET ADDRESS OF THE LAST MEMORY LOCATION ON THE SYSTEM
```

```
1254 ;SIZE MEMORY AND COMPARE IT WITH THE SYSTEM SIZE REGISTER
1255 ;PRINT A WARNING IF THEY DISAGREE.
1256
1257 003700 052767 000200 055716 BIS #BIT07,$KT11
1258 003706 004767 055644 JSR PC,$SIZE
1259 003712 062767 000037 056222 ADD #37,$LSTBK
1260 003720 016702 056216 MOV $LSTBK,R2 ;COPY LAST BLOCK COUNT
1261 003724 023702 177760 CMP @#SIZELO,R2 ;EQUAL?
1262 003730 001551 BEQ OKSIZ
1263 003732 012737 062144 000034 MOV #STRAP,@#TRAPVEC ;SET UP TRAP VECTOR
1264 003740 104400 003746 TYPE ,65$ ;:TYPE ASCIZ STRING
1265 003744 000433 BR 64$ ;:GET OVER THE ASCIZ
1266 ;:65$: .ASCIZ <15><12>/WARNING- THE SIZE OF MEMORY IS DIFFERENT FROM THAT/
1267 004034 64$:
1268 004034 104400 004042 TYPE ,67$ ;:TYPE ASCIZ STRING
1269 004040 000425 BR 66$ ;:GET OVER THE ASCIZ
1270 ;:67$: .ASCIZ <15><12>/INDICATED BY THE SYSTEM SIZE REGISTER./
1271 004114 66$:
1272 004114 104400 004122 TYPE ,69$ ;:TYPE ASCIZ STRING
1273 004120 000421 BR 68$ ;:GET OVER THE ASCIZ
1274 ;:69$: .ASCIZ <15><12>/ SIZEHI SIZELO ACTUAL/
1275 68$:
1276 004164 TYPE ,$CRLF
1277 004170 013746 177762 MOV @#SIZEHI,-(SP) ;:SAVE @#SIZEHI FOR TYPEOUT
1278 004174 104404 TYPOS ;:GO TYPE--OCTAL ASCII
1279 004176 006 .BYTE 6 ;:TYPE 6 DIGIT(S)
1280 004177 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
1281 004200 104400 004206 TYPE ,71$ ;:TYPE ASCIZ STRING
1282 004204 000404 BR 70$ ;:GET OVER THE ASCIZ
1283 ;:71$: .ASCIZ / /
1284 70$:
1285 004216 MOV @#SIZELO,-(SP) ;:SAVE @#SIZELO FOR TYPEOUT
1286 004222 104404 TYPOS ;:GO TYPE--OCTAL ASCII
1287 004224 006 .BYTE 6 ;:TYPE 6 DIGIT(S)
1288 004225 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
1289 004226 104400 004234 TYPE ,73$ ;:TYPE ASCIZ STRING
1290 004232 000404 BR 72$ ;:GET OVER THE ASCIZ
1291 ;:73$: .ASCIZ / /
1292 72$:
1293 004244 MOV $LSTBK,-(SP) ;:SAVE $LSTBK FOR TYPEOUT
1294 004244 016746 055672 TYPOS ;:GO TYPE--OCTAL ASCII
1295 004250 104404 .BYTE 6 ;:TYPE 6 DIGIT(S)
1296 004253 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
1297
1298 ;:FORM MXMMHI, MXMML0, AND THE HIGHEST MEMORY ADDRESS BASED ON THE SIZE OF
1299 ;:THE MEMORY
1300
1301 OKSIZ:
1302 004254 CLR R2
1303 004256 005002 MOV @#$LSTBK,R3
1304 004262 073227 000006 ASHC #6,R2 ;:SHIFT TO FORM CORRECT ADDRESS
1305 004266 052703 000077 BIS #77,R3 ;:ENSURE LOWER SIX BITS SET
1306 004272 062703 000001 ADD #1,R3
1307 004276 005502 ADC R2
1308 ;:*****
1309 004300 010237 001606 MOV R2,@#MXMMHI ;:SAVE UPPER SIX BITS
```

```

1310 004304 010337 001610      MOV     R3,@MMXMMLO      ;SAVE LOWER 16 BITS
1311
1312 004310 012706 001200      MOV     #KERSTK,SP      ;SET STACK PTR
1313 004314 005037 001200      CLR     @#SPASS        ;CLEAR PASS COUNT
1314 004320 105037 001531      CLRB   @#MMON         ;SET MEM MGMT ON IND-NOT ON
1315 004324 012737 000700 001546  MOV     #700,@#NEXPAR   ;SET FIRST 'PAR' VALUE
1316 004332 005737 003574      TST    @#PROT
1317 004336 001403      BEQ    1$
1318 004340 012737 001600 001546  MOV     #1600,@#NEXPAR
1319 004346      1$:
1320 004346 012700 000027      MOV     #27,R0         ;SET SOB COUNT
1321 004352 005001      CLR     R1             ;SETUP INDEX
1322 004354 005061 001626      2$:  CLR     MTICKS(R1)   ;CLEAR TABLES
1323 004360 062701 000022      ADD     #2,R1
1324 004364 077005      SOB    R0,2$          ;CONTINUE
1325 004366 012737 177777 001716  MOV     #-1,@#MAPTBL   ;INITIALIZE MAP TABLE
1326 004374 012737 177777 001720  MOV     #-1,@#MAPTBL+2
1327 004402 012700 000010      MOV     #10,R0        ;SET SOB COUNT
1328 004406 012701 001,36      MOV     #RPHSTAT,R1   ;GET ADDRESS OF HANDLER STAT
1329 004412 012721 000200      3$:  MUV     #200,(R1)+   ;INITIALIZE STATUS TABLE
1330 004416 077003      SOB    R0,3$          ;CONTINUE
1331 004420 012737 000060 001560  MOV     #60,@#SUBPASS  ;INIT SUBPASS TO ASCII 0
1332 004426 012700 056734      MOV     #TIMEBUF,R0   ;GET ADR OF TIME BUFFER
1333 004432 012701 000022      MOV     #12,R1        ;SET SOB COUNT
1334 004436 112720 000060      4$:  MOVB   #60,(R0)+     ;INIT TIME BUFFER
1335 004442 077103      SOB    R1,4$
1336 004444 105040      CLRB   -(R0)          ;INSERT TERMINATOR
1337 004446 112737 000072 056737  MOVB   #72,@#TIMEBUF+3 ;INSERT COLON
1338 004454 112737 000072 056742  MOVB   #72,@#TIMEBUF+6
1339 004462 012737 000340 177776  MOV     #340,@#PS     ;;LOCK OUT ALL INTERRUPTS
1340 004470 012706 001200      MOV     #SCMTAG,R6    ;;FIRST LOCATION TO BE CLEARED
1341 004474 005026      CLR     (R6)+        ;;CLEAR MEMORY LOCATION
1342 004476 022736 001240      CMP    #STKS,R6     ;;DONE?
1343 004502 001374      BNE    #-6          ;;LOOP BACK IF NO
1344 004504 012706 001200      MOV     #STACK,SP    ;;SETUP THE STACK POINTER
1345 004510 012737 054370 000020  MOV     #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1346 004516 012737 000340 000022  MOV     #340,@#ICTVEC+2 ;;LEVEL 7
1347 004524 012737 054622 000030  MOV     #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1348 004532 012737 000370 000032  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1349 004540 012737 062144 000034  MOV     #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1350 004546 012737 000340 000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
1351 004554 012737 061062 000024  MOV     #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1352 004562 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1353 004570 016767 041640 041630  MOV     $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
1354 004576 005067 174524      CLR     $TIMES        ;;INITIALIZE NUMBER OF ITERATIONS
1355 004602 005067 174522      CLR     $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1356 004606 112767 000001 174403  MOVB   #1,$ERMAX     ;;ALLOW ONE ERROR PER TEST
1357 004614 012767 004614 174366  MOV     #,$SLPADR    ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1358 004622 012767 004622 174362  MOV     #,$SLPERR    ;;SETUP THE ERROR LOOP ADDRESS
1359
1360      ;CLEAR PROGRAM INDICATORS
1361 004630 052777 000100 174402  BIS    #100,$STKS    ;SET IE BIT IN KEYBOARD STATUS REG
1362 004636 012737 063222 000060  MOV     #TKISR,@#TKVEC ;SETUP KEYBOARD VECTOR
1363 004644 012737 000200 000062  MOV     #PR4,@#TKVEC+2
1364 004652 012737 063434 000064  MOV     #TPISR,@#TPVEC
1365 004660 012737 000200 000066  MOV     #PR4,@#TPVEC+2

```

```

1366 004666 005037 001476 CLR @NOTYPE ;CLEAR 'NO TYPING' INDICATOR
1367
1368 ;THE BELOW ROUTINE ASCERTAINS WHICH CP & CP OPTIONS THE PROGRAM IS RUN-
1369 ;NING ON AND SETS AN INDICATOR IN OPT.CP ACCORDINGLY.
1370 004672 012737 000006 000004 CPCHK: MOV #ERRVEC+2,@ERRVEC ;SET UP ERROR TRAP TO RETURN
1371 004700 012737 000002 000006 MOV #2,@ERRVEC+2
1372 004706 012737 000012 000010 MOV #RESVEC+2,@RESVEC ;AND ALSO RESERVED INST TRAP
1373 004714 012737 000002 000012 MOV #2,@RESVEC+2
1374 004722 012702 144006 MOV #144006,R2 ;SET 11/70 NON-OPTION BITS
1375 004726 000261 SEC
1376 004730 170500 TSTF R0 ;WILL CLEAR CARRY IF 11/70 FLOATING POINT
1377 004732 170000 CFCC ;IS AVAIL. COPY FLOATING CC'S INTO PSW
1378 004734 103402 BCS 6$ ;BRANCH IF NO FLOATING POINT
1379 004736 052702 020000 BIS #FPOPT,R2 ;SET FP OPTION AVAIL INDICATOR
1380 004742 000261 6$: SEC
1381 004744 005737 177546 TST @WLKS ;BRANCH IF NO KW11-L
1382 004750 103402 BCS 7$
1383 004752 052702 001000 BIS #LKOPT,R2 ;SET OPTION INDICATOR
1384 004756 000261 7$: SEC
1385 004760 005777 174260 TST @STPS ;BRANCH IF NO CONSOLE TTY
1386 004764 103402 BCS 9$
1387 004766 052702 000400 BIS #TTOPT,R2
1388 004772 005003 9$: CLR R3
1389 004774 000261 SEC
1390 004776 005737 170000 TST @UBEDB ;IS UBE1 THERE?
1391 005002 103410 BCS 12$ ;BRANCH IF NO
1392 005004 105037 170006 CLRB @UBECR1 ;IS THIS A TESTER OR EXERCISER?
1393 005010 105737 170006 TSTB @UBECR1
1394 005014 100045 BPL 15$ ;BRANCH IF TESTER
1395 005016 052702 000200 BIS #UBEOPT,R2 ;SET INDICATOR
1396 005022 000425 BR 17$
1397 005024 000261 12$: SEC
1398 005026 005737 170020 TST @UBEDB+20 ;IS UBE2 THERE?
1399 005032 103403 BCS 13$ ;BRANCH IF NO
1400 005034 012703 000020 MOV #20,R3 ;SET OFFSET IN R3
1401 005040 000766 BR 16$
1402 005042 000261 13$: SEC
1403 005044 005737 170040 TST @UBEDB+40 ;IS UBE3 THERE?
1404 005050 103403 BCS 14$ ;BRANCH IF NO
1405 005052 012703 000040 MOV #40,R3 ;PUT OFFSET IN R3
1406 005056 000757 BR 16$
1407 005060 000261 14$: SEC
1408 005062 005737 170060 TST @UBEDB+60 ;IS UBE4 THERE?
1409 005066 103420 BCS 15$ ;BRANCH IF NO
1410 005070 012703 000060 MOV #60,R3 ;PUT OFFSET IN R3
1411 005074 000750 BR 16$
1412 005076 005227 177777 17$: INC #-1
1413 005102 001012 BNE 15$
1414 005104 012704 002270 MOV #UBETBL,R4 ;GET ADDRESS OF UBE TABLE
1415 005110 012705 000005 MOV #5,R5 ;SET SOB COUNT
1416 005114 060324 18$: ADD R3,(R4)+ ;ADJUST UBE TABLE ENTRIES
1417 005116 077502 SOB R5,18$ ;CONTINUE
1418 005120 006003 ROR R3
1419 005122 006003 ROR R3 ;ADJUST OFFSET FOR UBE VECTOR
1420 005124 060324 ADD R3,(R4)+ ;ADJUST UBEVEC ENTRY
1421 005126 060314 ADD R3,(R4) ;ADJUST UBEVEC PSW ENTRY
  
```

```

1422 005130 005003      15$: CLR      R3              ;INIT R3
1423 005132 000261      SEC
1424 005134 005777 175146  TST      @MBTTBL          ;IS MASS BUS TESTER THERE?
1425 005140 103403      BCS      20$              ;BRANCH IF NO
1426 005142 052702 002000 21$: BIS      @MBTOPT,R2    ;SET OPTION AVAILABLE
1427 005146 000422      BR       24$
1428 005150 005777 175162 20$: TST      @MBTN2        ;IS MBT2 THERE?
1429 005154 103403      BCS      22$              ;BRANCH IF NO
1430 005156 012703 000100  MOV      #100,R3         ;SETUP R3
1431 005162 000767      BR       21$
1432 005164 005777 175150 22$: TST      @MBTN3        ;IS MBT3 THERE?
1433 005170 103403      BCS      23$              ;BRANCH IF NO
1434 005172 012703 000200  MOV      #200,R3
1435 005176 000761      BR       21$
1436 005200 005777 175136 23$: TST      @MBTN4        ;IS MBT4 THERE?
1437 005204 103427      BCS      30$              ;BRANCH IF NO
1438 005206 012703 000300  MOV      #300,R3
1439 005212 000753      BR       21$
1440 005214      24$:
1441 005214 000240      NOP
1442 005216 000240      NOP
1443 005220 000240      NOP
1444 005222 012704 002306  MOV      @MBTTBL,R4      ;GET ADDRESS OF MBT TABLE
1445 005226 012705 000011  MOV      #11,R5          ;SET SOB COUNT
1446 005232 060324      25$: ADD      R3,(R4)+        ;ADJUST MBT TABLE
1447 005234 077502      SOB      R5,25$         ;CONTINUE
1448 005236 060337 002334  ADD      R3,@MBTTBL+26   ;ADJUST DRIVE TYPE ADDRESS
1449 005242 112777 000007 175050  MOVB     #7,@MBTTBL+12   ;SET UNIT NUMBER
1450 005250 122777 000040 175056  CMPB     #40,@MBTTBL+26  ;IS THIS REALLY A MBT?
1451 005256 001402      BEQ     30$              ;BRANCH IF YES
1452 005260 042702 002000  BIC      @MBTOPT,R2      ;CLEAR OPTION AVAILABLE BIT
1453 005264 012737 064270 000004 30$: MOV      @ERRPT,@ERRVEC ;RESTORE ERROR TRAP
1454
1455      ;*** TEST FOR VARIOUS KB11 PROCESSORS ***
1456
1457      ;*THIS ROUTINE POLES THE RESULTS OF ATTEMPTS TO SET TO ONE
1458      ;*CERTAIN CRITICAL BITS THAT ARE KNOWN TO BE OPERATIVE ON A KB11CM,
1459      ;*OR KB11EM PROCESSOR. IF TWO OUT OF FOUR OF THE TESTS ARE
1460      ;*POSITIVE THEN THE KB11CM OR KB11EM FLAG IS SET,IF LESS THAN TWO OF THE
1461      ;*TESTS ARE POSITIVE THEN THE KB11E FLAG OR NO FLAG IS SET. THE DETERMINATION
1462      ;*OF WHICH PAIR IS VALID IS BASED ON THE RESULTS OF EXECUTING AN MFPT OPCODE
1463      ;*(OPCODE 7). IF THIS INSTRUCTION TRAPS THIS IS AN KB11CM OR
1464      ;*A PLAIN 1170 (KB11-B OR KB11-C). IF THE INSTRUCTION DOES NOT TRAP THEN
1465      ;*THIS IS A KB11-E OR KB11-EM.
1466
1467 005272 104420      SAVREG      ;SAVE GPRS R5-R0
1468 005274 105037 001504  CLRB     @KB11CM        ;RESET THE MP FLAG
1469 005300 005037 001502  CLR      @KB11E         ;CLEAR KB11E AND KB11EM FLAGS
1470 005304 012737 005552 000010  MOV      @MFPTTR,@RESVEC ;SET UP TRAP ADDRESS FOR MFPT AT RESERV VECTOR
1471 005312 000007      MFPT        ;EXECUTE MFPT. WILL TRAP ON 1170 (KB11B/C) OR
1472      ;KB11CM (11/74 )
1473 005314 012737 000001 001502  MOV      #1,@KB11E      ;HERE IF KB11E OR KB11EM. SET FLAG
1474 005322 005037 177750  T1: CLR      @MAINT      ;CLEAR THE MAINTENANCE REGISTER
1475 005326 005005      CLR      R5             ;RESET THE TEST COUNTER
1476 005330 012700 177746  MOV      @CONTRL,R0     ;GET THE ADDRESS OF...
1477 005334 012701 177750  MOV      @MAINT,R1      ;CCR,MAINT,AND MAPH00...
    
```


1478	005340	012702	170202		MOV	#MAPH00,R2	:AND PLACE IN R0-R2
1479	005344	052710	040000		BIS	#BIT14,(R0)	:TRY TO SET IVSS BIT
1480	005350	032710	04000C		BIT	#BIT14,(R0)	:DID IT SET?
1481	005354	001403			BEQ	T2	:NO,GO TO NEXT TEST
1482	005356	042710	040000		BIC	#BIT14,(R0)	:CLEAR IT.
1483	005362	005205			INC	R5	:TEST IS POSITIVE
1484	005364	052711	000001	T2:	BIS	#BIT0,(R1)	:SET EDMA IN MAINT REGISTER
1485	005370	032711	000001		RIT	#BIT0,(R1)	
1486	005374	001410			BEQ	T3	
1487	005376	052710	004000		BIS	#BIT11,(R0)	:TRY TO SET DMMA IN CCR
1488	005402	032710	004000		BIT	#BIT11,(R0)	
1489	005406	001403			BEQ	T3	
1490	005410	042710	004000		BIC	#BIT11,(R0)	
1491	005414	005205			INC	R5	
1492	005416	042711	000001	T3:	BIC	#BIT0,(R1)	:MAKE SURE EDMA IS CLEAR
1493	005422	052767	100000	164650	BIS	#BIT15,KIPDR0	:TRY TO SET BYP ON A PDR
1494	005430	032767	100000	164642	BIT	#BIT15,KIPDR0	
1495	005436	001404			BEQ	T4	
1496	005440	042767	100000	164632	BIC	#BIT15,KIPDR0	
1497	005446	005205			INC	R5	
1498	005450	052712	100000	T4:	BIS	#BIT15,(R2)	:TRY TO SET BYP ON UNIBUS MAP
1499	005454	032712	100000		BIT	#BIT15,(R2)	
1500	005460	001403			BEQ	T.END	
1501	005462	042712	100000		BIC	#BIT15,(R2)	
1502	005466	005205			INC	R5	
1503	005470	022705	000002	T.END:	CMP	#2,R5	:IS THE RESULT OF THE TEST >=2
1504	005474	101021			BHI	3\$:BR IF NO,THIS IS A KB11E OR KB11-B/C (11/70)
1505	005476	005000			CLR	R0	
1506	005500	005037	177746		CLR	@CONTRL	
1507	005504	013701	177746	4\$:	MOV	@CONTRL,R1	
1508	005510	001402			BEQ	5\$	
1509	005512	005200			INC	R0	
1510	005514	001373			BNE	4\$	
1511	005516			5\$:			
1512	005516	005737	001502		TST	@KB11E	:IS IS A KB11-E OR KB11-EM?

1513 005522 001404
1514 005524 012737 000400 001502

BEQ 1\$:BR IF NEITHER. MUST BE KB11CM
MOV #BIT8,@KB1E :SET UPPER BYTE (KB11-EM)

1515 005532 000405
1516 005534 105237 001504
1517 005540 005737 001502
1518 005544 001472
1519 005546 104422
1520 005550 000403

BR 2\$
1\$: INCB @KB11CM
5\$: TST @KB11E
BEQ RESTORE
2\$: RESREG
BR ENDKB

:DONE
:YES, FLAG THIS AS A MODIFIED PROCESSOR
:IS THIS A KB11E?
:BR IF NOT, THIS IS AN 1170
:RESTORE R5-R0
:DONE DETERMINEING WHICH CPU

```

1521
1522 005552          MFPTTR:          :HERE IF MFPT TRAPPED. SEE IF 1170 OR KB11CM
1523 005552 012716 005322      MOV      #T1,(SP)      :SET UP RTI RETURN ADDRESS
1524 005556 000002          RTI          :RETURN
1525 005560          ENDK8:
1526
1527          :SEE IF CISP IS PRESENT. TRY TO EXECUTE 3 CISP INSTRUCTIONS. IF TWO OUT
1528          :OF THE THREE DON'T TRAP, IT IS ASSUMED THAT THE CISP OPTION IS PRESENT AND
1529          :A FLAG IS SET TO INDICATE THIS. ALSO A BIT IS SET IN OPT.CP AND A MESSAGE
1530          :IS PRINTED.
1531
1532
1533 005560 052702 010000          BIS      #CISOPT,R2      :SET CISP OPTION BIT FOR OPT.CP
1534 005564 104420          SAVREG          :SAVE R5-R0
1535 005566 012737 005716 000010      MOV      #TRPRTN,@#RESVEC :SET UP TRAP ADDRESS AT RESERVED VECTOR
1536 005574 105037 001505          CLRB     @#CISP          :COUNT HOW MANY CIS OPCODES DON'T TRAP
1537 005600 012737 076020 005712      MOV      #L2D0,@#CISOP   :CIS OPCODE TO TEST (LOAD 2 DESCRIPTORS @R0)
1538 005606 012700 001632          MOV      #LD2PNT,R0      :R0 MUST BE EVEN AND POINT TO A WORD WHICH IS ALSO EVEN
1539 005612 004737 005712          JSR      PC,@#CISOP      :TEST OPCODE FOR A TRAP
1540 005616 012737 076061 005712      MOV      #L3D1,@#CISOP   :SET UP OPCODE FOR LOAD 3 DESCRIPTORS @R1
1541 005624 012701 001632          MOV      #LD2PNT,R1      :LOAD R1 WITH EVEN WORD AND POINT TO EVEN CONTENTS
1542 005630 004737 005712          JSR      PC,@#CISOP      :TEST OPCODE FOR TRAP
1543 005634 052737 100000 177770      BIS      #BIT15,@#UBREAK  :SET MAINT MODE IN U-BREAK REGISTER
1544 005642 012737 076601 005712      MOV      #MED74C,@#CISOP :OPCODE FOR DIAGNOSTIC ENTRY
1545 005650 012705 006600          MOV      #CISTST,R5      :ADDRESS OF DIAGNOSTIC U-CODE
1546 005654 004737 005712          JSR      PC,@#CISOP      :TEST OPCODE FOR TRAP
1547 005660 104422          RESREG          :RESTORE R5-R0
1548 005662 122737 000002 001505      CMPEB   #2,@#CISP        :IS RESULT >=2?
1549 005670 101404          BLOS    1$           :BR IF CISP IS PRESENT
1550 005672 105037 001505          CLRB     @#CISP          :CLEAR CISP PRESENT FLAG
1551 005676 042702 010000          BIC      #CISOPT,R2      :AND ALSO IN OPT.CP
1552 005702 042737 100000 177770 1$:      BIC      #BIT15,@#UBREAK  :CLEAR MAINT BIT IN U-BREAK REGISTER
1553 005710 000411          BR       SETOP         :GO TO RESTORE VECTOR AND SET OPT.CP
1554
1555 005712 000000          CISOP: .WORD 0          :CISP OPCODE WILL GO HERE FOR EXECUTION
1556 005714 000403          BR       NOTRAP        :WILL COME HERE IF NO TRAP
1557 005716 012716 005730      TRPRTN: MOV      #CISTRP,(SP) :SET UP RTI RETURN ADDRESS
1558 005722 000002          RTI          :RETURN TO LOCATION FROM TRAP
1559 005724 105237 001505      NOTRAP: INCB    @#CISP      :INCREMENT CISP INDICATOR
1560 005730 000207          CISTRP: RTS      PC       :RETURN
1561
1562 005732 104422          RESTOR: RESREG          :RESTORE R5-R0
1563 005734          SETOP:
1564 005734 012737 064216 000010      MOV      #RESERR,@#RESVEC :AND ALSO RESERVED INST TRAP
1565 005742 010237 001500          MOV      R2,@#OPT.CP     :LOAD INDICATOR
1566 005746 005227 177777          INC      #-1             :;FIRST TIME?
1567 005752 001034          BNE     64$             :;BRANCH IF NO
1568 005754 022737 046570 000042      CMP      #SENDAD,@#42    :;ACT-11?
1569 005762 001430          BEQ     64$             :;BRANCH IF YES
1570 005764 104400 005772          TYPE    ,65$           :;TYPE ASCIZ STRING
1571 005770 000425          BR      64$             :;GET OVER THE ASCIZ
1572          ;;65$: .ASCIZ <CRLF>'CEQKC-D...PDP 11/70-74MP CPU EXERCISER'<CRLF>
1573          64$:
1574 006044 005227 177777          INC      #-1             :;FIRST TIME?
1575 006050 001036          BNE     100$            :;BR IF NO
1576 006052 104400 071075          TYPE    ,MSG34          :;<15><12>CPU UNDER TEST FOUND TO BE A
    
```

```
1577 006056 005737 001502      TST      @#KB11E      ;IS THIS A KB11-E OR KB11-EM?
1578 006062 001011      BNE      101$        ;BR IF EITHER ONE
1579 006064 105737 001504      TSTB     @#KB11CM    ;IS IT A 11/74      (KB11CM)
1580 006070 001003      BNE      1$         ;BR IF IT IS
1581 006072 104400 071135      TYPE     ,MSG35     ;KB11-B/C<15><12>
1582 006076 000423      BR       100$      ;SKIP OTHER MESSAGE
1583 006100 104400 071043      1$:     TYPE     ,MSG32    ;11/74      (KB11CM)<15><12>
1584 006104 000420      BR       100$      ;SKIP CISP MESSAGE
1585 006106 105737 001502      101$:   TSTB     @#KB11E    ;IS IT A KB11-E?
1586 006112 001403      BEQ      102$      ;BR IF NOT. MUST BE KB11-EM
1587 006114 104400 071150      TYPE     ,MSG36     ;KB11-E<15><12>
1588 006120 000402      BR       104$      ;SKIP KB11-EM MESSAGE
1589 006122 104400 071031      102$:   TYPE     ,MSG31    ;KB11-EM<15><12>
1590 006126 105767 173353      104$:   TSTB     CISP      ;IS CISP PRESENT?
1591 006132 001003      BNE      103$      ;BR IF CISP PRESENT
1592 006134 104400 071161      TYPE     ,MSG37     ;CISP OPTION NOT FOUND<15><12>
1593 006140 000402      BR       100$      ;SKIP OTHER MESSAGE
1594 006142 104400 071211      103$:   TYPE     ,MSG38    ;CISP OPTION FOUND<15><12>
1595 006146
1596 006146 104400 006154      TYPE     ,67$      ;;TYPE ASCIZ STRING
1597 006152 000415      BR       66$      ;;GET OVER THE ASCIZ
1598
1599 006206      66$:    .ASCIZ  <15><12>/PROCESSOR ID REGISTER =/
1600 006206 013746 177764      MOV      @#177764,-(SP) ;;SAVE @#177764 FOR TYPEOUT
1601 006212 104402      TYPOC
1602 006214 104400 006222      TYPE     ,69$      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1603 006220 000406      BR       68$      ;;TYPE ASCIZ STRING
1604
1605 006236      68$:    .ASCIZ  / (OCTAL) /
1606 006236 013746 177764      MOV      @#177764,-(SP) ;;SAVE @#177764 FOR TYPEOUT
1607 006242 104410      TYPDS
1608 006244 104400 006252      TYPE     ,71$      ;;GO TYPE--DECIMAL ASCII WITH SIGN
1609 006250 000406      BR       70$      ;;TYPE ASCIZ STRING
1610
1611 006266      70$:    .ASCIZ  / (DECIMAL) /
1612 006266 104400 001337      TYPE     ,SCLRF
1613
1614      ;*****
1615      .SBITL  SYSTEM SIZER
1616      ; THIS ROUTINE DETERMINES WHAT DRIVES ARE AVAILABLE ON
1617      ; THE FOLLOWING DEVICES: RK05, RP03, RP04, AND RS04. THE
1618      ; INFORMATION IS STORED IN THE TABLE 'SYSSIZE' IN THE FOLLOWING FORMAT:
1619      ; A. EACH DEVICE IS ASSIGNED A WORD
1620      ; B. THE LOW BYTE OF THIS WORD INDICATES WHICH DRIVES ARE AVAILABLE
1621      ; C. THE HIGH BYTE INDICATES WHICH DRIVES HAVE BEEN USED
1622      ; BY THE RELOCATION ROUTINE.
1623 006272 012737 006404 000004      SIZE:   MOV      #21$,@#ERRVEC ;SETUP TIMEOUT VECTOR
1624 006300 005037 001302      CLR     @#STMP0     ;ENSURE $TMP0 CLEAR
1625 006304 005000      CLR     R0         ;USED TO SET THE UNIT AVAIL BITS
1626 006306 012701 000010      MOV     #10,R1     ;SOB COUNT
1627 006312 013777 001302 173654      9$:    MOV     @#STMP0,@#RKDA ;SET UNIT NUMBER
1628 006320 012777 000015 173640      MOV     #15,@#RKCS ;SEND DRIVE RESET
1629 006326 032777 000200 173630      BIT     #BIT7,@#RKER ;NON EXISTANT DISK?
1630 006334 001011      BNE     7$         ;BRANCH IF YES
1631 006336 017702 173620      MOV     @#RKDS,R2  ;GET DRIVE STATUS
1632 006342 042702 177537      BIC     #177537,R2 ;GET BITS 5 & 7 ONLY
```

```
1633 006346 022702 000200          CMP      #200,R2          ;IS DRIVE READY?
1634 006352 001002          BNE      7$              ;BRANCH IF NO
1635 006354 052700 000400          BIS      #BIT8,R0        ;SET UNIT AVAILABLE
1636 006360 006000          ROR      R0              ;
1637 006362 012777 000001 173576 7$:  MOV      #1,@R0CS        ;CLEAR THE ERRORS
1638 006370 062737 020000 001302  ADD      #20000,@$STMP0  ;SELECT NEXT UNIT
1639 006376 077133          SOB      R1,9$          ;CONTINUE
1640 006400 110037 001644          MOVSB   R0,@$SYSSIZE+2  ;STORE IN TABLE
1641
1642
1643
1644
1645
1646
1647
1648 006404 012737 007012 000004 21$:  MOV      #11$,@$ERRVEC   ;SET THE ERROR VECTOR
1649 006412 005737 176710          TST      @#176710       ;IS THERE AN RP ON THE SYSTEM?
1650
1651 006416 012737 006432 000004          MOV      #1$,@$ERRVEC   ;
1652 006424 005777 173552          TST      @RP4CS1        ;IS THERE AN RP04 ON SYSTEM?
1653 006430 000441          BR       10$           ;BRANCH IF YES
1654
1655
1656
1657 006432 012737 005534 000004 1$:  MOV      #10$,@$ERRVEC  ;SETUP TIMEOUT VEC FOR RP03 TEST
1658 006440 012737 000001 001302  MOV      #1,@$STMP0     ;SETUP TEMPO
1659 006446 005000          CLR      R0              ;USED TO SET UNIT AVAILABLE BITS
1660 006450 012701 000010          MOV      #10,R1         ;SOB COUNT
1661 006454 013777 001302 173462 3$:  MOV      @$STMP0,@RP3CS  ;SET FUNCTION IDLE WITH UNIT NO
1662 006462 005777 173456          TST      @RP3CS        ;WAS THERE AN ERROR?
1663 006466 100006          BPL      6$            ;BRANCH IF NO
1664 006470 006000          ROR      R0              ;UNIT NOT AVAILABLE
1665 006472 062737 000400 001302 4$:  ADD      #400,@$STMP0   ;SELECT NEXT UNIT
1666 006500 077113          SOB      R1,3$         ;CONTINUE
1667 006502 000412          BR       5$            ;
1668 006504 017702 173430          MOV      @RP3DS,R2      ;GET STATUS REGISTER
1669 006510 042702 037777          BIC      #37777,R2      ;GET BITS 14, 15 ONLY
1670 006514 022702 140000          CMP      #140000,R2     ;IS DRIVE READY?
1671 006520 001363          BNE      4$            ;BRANCH IF NO
1672 006522 052700 000400          BIS      #BIT8,R0        ;SET DRIVE AVAILABLE BIT
1673 006526 000760          BR       4$            ;CONTINUE
1674 006530 110037 001642 5$:  MOVSB   R0,@$SYSSIZE   ;STORE IN TABLE
1675
1676
1677 006534 012737 007012 000004 10$:  MOV      #11$,@$ERRVEC  ;SETUP ERROR VEC FOR RP04 TEST
1678 006542 005037 001302          CLR      @$STMP0        ;
1679 006546 005000          CLR      R0              ;UNIT AVAILABLE WORD
1680 006550 012701 000010          MOV      #10,R1         ;SOB COUNT
1681 006554 113777 001302 173432 14$:  MOVSB   @$STMP0,@RP4CS2 ;SET UNIT NUMBER
1682 006562 012777 000021 173412  MOV      #21,@RP4CS1    ;TRY READ-IN-PRESET
1683 006570 032777 010000 173416  BIT      #BIT12,@RP4CS2 ;NON EXISTANT DRIVE?
1684 006576 001071          BNE      12$           ;BRANCH IF YES
1685 006600 017702 173414          MOV      @RP4DS,R2      ;GET DRIVE STATUS
1686 006604 032702 001000          BIT      #BIT9,R2        ;IS DRIVE IN PROGRAMMABLE MODE?
1687 006610 001455          BEQ      8$            ;NO
1688 006612 104400 006620          TYPE    ,65$           ;:TYPE ASCIZ STRING
```

```

1689 006616 000410          BR      64$          ;;GET OVER THE ASCIZ
1690          ;;65$: .ASCIZ <15><12>/RP04 DRIVE #/
1691 006640          64$:
1692 006640 013746 001302    MOV     @#STMP0,-(SP)    ;;SAVE @#STMP0 FOR TYPEOUT
1693 006644 104402          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1694 006646 104400 006654    TYPE    ,67$          ;;TYPE ASCIZ STRING
1695 006652 000433          BR      66$          ;;GET OVER THE ASCIZ
1696          ;;67$: .ASCIZ / FOUND IN PROGRAMMABLE MODE-DRIVE WILL NOT BE USED/<15><12>
1697 006742          66$:
1698 006742 000407          BR      12$
1699 006744 042702 163277    BIC     #163277,R2      ;GET BITS 12, 11, 8, & 6 ONLY
1700 006750 022702 010500    CMP     #10500,R2      ;IS DRIVE READY?
1701 006754 001002          BNE     12$          ;BRANCH IF NO
1702 006756 052700 000400    BIS     #BIT8,R0       ;SET UNIT AVAILABLE
1703 006762 006000          12$: ROR     R0
1704 006764 052777 000040 173222    BIS     #BIT5,@RP4CS2  ;CLEAR ERROR BITS
1705 006772 005237 001302    INC     @#STMP0        ;SELECT NEXT DRIVE
1706 006776 005301          DEC     R1
1707 007000 001402          BEQ     ,+6
1708 007002 000167 177546    JMP     14$
1709 007006 110037 001652    MOVB   R0,@#SYSSIZE+10 ;STORE IN TABLE
1710
1711          ;*****
1712 007012 012737 007122 000004 11$: MOV     #15$,@#ERRVEC  ;SETUP ERROR VEC FOR RS04 TEST
1713 007020 005037 001302          CLR     @#STMP0
1714 007024 005000          CLR     R0
1715 007026 012701 000010          MOV     #10,R1        ;SOB COUNT
1716 007032 113777 001302 173214 18$: MOVB   @#STMP0,@RSCS2 ;SET UNIT NUMBER
1717 007040 012777 000001 173174    MOV     #1,@RSCS1     ;TRY NOP OPERATION
1718 007046 032777 010000 173200    BIT     #BIT12,@RSCS2 ;NON EXISTANT DRIVE?
1719 007054 001011          BNE     16$          ;BRANCH IF YES
1720 007056 017702 173176    MOV     @RSDS,R2      ;GET DRIVE STATUS
1721 007062 042702 163577    BIC     #163577,R2    ;GET BITS 12, 11, & 7 ONLY
1722 007066 022702 010200    CMP     #10200,R2     ;IS DRIVE READY?
1723 007072 001002          BNE     16$          ;BRANCH IF NO
1724 007074 052700 000400    BIS     #BIT8,R0     ;SET DRIVE AVAILABLE BIT
1725 007100 006000          16$: ROR     R0
1726 007102 052777 000040 173144    BIS     #BIT5,@RSCS2  ;CLEAR ANY ERROR BITS
1727 007110 005237 001302    INC     @#STMP0        ;SELECT NEXT UNIT
1728 007114 077132          SOB    R1,18$        ;CONTINUE
1729 007116 110037 001654    MOVB   R0,@#SYSSIZE+12 ;STORE IN TABLE
1730
1731          ;NEXT, DELETE XXDP UNIT 0 FROM TABLE
1732 007122 122737 000002 000041 15$: CMPB   #2,@#41      ;RK?
1733 007130 001004          BNE     19$          ;BRANCH IF NO
1734 007132 042737 000001 001644    BIC     #BIT0,@#SYSSIZE+2 ;MAKE UNIT ZERO NOT AVAILABLE
1735 007140 000420          BR      20$
1736 007142 113700 000041 19$: MOVB   @#41,R0       ;GET LOCATION 41
1737 007146 042700 177770    BIC     #177770,R0    ;GET LEAST SIG 3 BITS
1738 007152 000241          CLC
1739 007154 006100          ROL     R0            ;ENSURE C CLEAR
1740 007156 122700 000002          CMPB   #2,R0         ;ADJUST
1741 007162 002404          BLT     40$
1742 007164 042737 000001 001642    BIC     #BIT0,@#SYSSIZE ;BRANCH IF NO
1743 007172 000403          BR      20$
1744 007174 042760 000001 001646 40$: BIC     #BIT0,SYSSIZE+4(R0)

```

```
1745 007202 005227 177777 20$: INC # -1
1746 007206 001057 BNF LOOP1 ;; BRANCH IF NOT FIRST TIME
1747 007210 104400 065475 TYPE ,MSG25
1748 007214 013746 001500 MOV @#OPT.CP,-(SP)
1749 007220 104402 TYPOC
1750 007222 104400 001337 TYPE ,$CRLF
1751 007226 005737 001532 TST @#QV ;ACT11?
1752 007232 001045 BNE LOOP1 ;BRANCH IF YES
1753 007234 105737 003571 50$: TSTB @#XXDPC ;XXDP CHAIN MODE?
1754 007240 001042 BNE LOOP1 ;; BRANCH IF YES
1755 007242 105737 001200 TSTB @#SPASS ;FIRST PASS?
1756 007246 001037 BNE LOOP1 ;; BRANCH IF NO
1757 007250 032737 000200 177570 BIT #SW7,@#SWR ;INHIBIT SIZE TIMEOUT?
1758 007256 001031 BNE SW8MSG ;; BRANCH IF YES
1759 007260 004767 047462 JSR PC,TYPS17 ;GO TYPE SYSTEM SIZE
1760 007264 104400 007272 TYPE ,69$ ;; TYPE ASCIZ STRING
1761 007270 000417 BR 68$ ;; GET OVER THE ASCIZ
1762 ;; 69$. .ASCIZ /TYPE A CHARACTER TO CONTINUE/<CRLF>
1763 68$:
1764 007330 005037 177776 CLR @#PSW
1765 007334 000001 WAIT
1766 007336 000137 006272 JMP @#SIZE ;GO CHECK SYSTEM AGAIN
1767 007342 104400 070517 SW8MSG: TYPE ,MSG30 ;TYPE SWITCH 8 REVERSAL MESSAGE
1768 007346 000167 000426 LOOP1: JMP LOOP
1769 010000
1770 ;PROGRAM RESTARTS HERE AFTER RELOCATION ABOVE 28K IS COMPLETE.
1771 ;INITIALIZE TRAP VECTORS
1772 010000 012706 000700 LOOP: MOV #SUPSTK,SP ;SET THE STACK...WILL BE DIFFERENT
1773 ;THAN KERN STACK WHEN IN OUTER MODE
1774 010004 012700 000004 MOV #ERRVEC,R0
1775 010010 013701 177776 MOV @#PSW,R1 ;GET CURRENT PSW
1776 010014 012720 064270 MOV #ERPRT,(R0)+ ;SET ERROR VEC
1777 010020 052701 000340 BIS #PR7,R1 ;SET PRIORITY 7 IN CURRENT PSW
1778 010024 042701 000020 BIC #BIT4,R1 ;CLEAR T BIT
1779 010030 010120 MOV R1,(R0)+
1780 010032 012720 064216 MOV #RESERR,(R0)+ ;SET RESERVED INST TRAP VECTOR
1781 010036 010120 MOV R1,(R0)+
1782 010040 012720 001520 MOV #SRTN,(R0)+ ;SET T BIT VEC
1783 010044 042701 000340 BIC #PR7,R1
1784 010050 005020 CLR (R0)+ ;SET TBIT VEC+2
1785 010052 005720 TST (R0)+ ;BUMP R0 TO SCOPE VEC+2
1786 010054 005020 CLR (R0)+ ;SET SCOPE VEC+2
1787 010056 062700 000006 ADD #6,R0 ;SET R0 TO ERROR TRAP VEC
1788 010062 012720 000340 MOV #PR7,(R0)+ ;SET ERROR VEC
1789 010066 005720 TST (R0)+
1790 010070 012720 000340 MOV #PR7,(R0)+ ;SET TRAP VEC+2
1791 010074 012737 063464 000114 MOV #.PARSRV,@#CACHVEC ;SET PARITY ERROR VECTOR
1792 010102 052701 000340 BIS #PR7,R1
1793 010106 010137 000116 MOV R1,@#CACHVEC+2
1794 010112 012737 064122 000250 MOV #KTABRT,@#MMVEC ;SET KT11 ABORT VECTOR
1795 010120 010137 000252 MOV R1,@#MMVEC+2
1796 010124 042737 000340 177776 BIC #PR7,@#PSW
1797
1798 ;*****
1799 ;*TEST 1 MEMORY VERIFICATION TEST
1800 010132 012767 000001 171166 MOV #1,$TIMES ;;DO 1 ITERATION
```


1857	010536	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1858	010544	177777	000000	000000		
1859	010552	000000	000000			
1860	010556	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1861	010564	177777	000000	000000		
1862	010572	000000	000000			
1863	010576	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1864	010604	177777	000000	000000		
1865	010612	000000	000000			
1866	010616	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1867	010624	177777	000000	000000		
1868	010632	000000	000000			
1869	010636	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1870	010644	177777	000000	000000		
1871	010652	000000	000000			
1872	010656	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1873	010664	177777	000000	000000		
1874	010672	000000	000000			
1875	010676	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1876	010704	177777	000000	000000		
1877	010712	000000	000000			
1878	010716	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1879	010724	177777	000000	000000		
1880	010732	000000	000000			
1881	010736	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1882	010744	177777	000000	000000		
1883	010752	000000	000000			
1884	010756	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1885	010764	177777	000000	000000		
1886	010772	000000	000000			
1887	010776	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1888	011004	177777	000000	000000		
1889	011012	000000	000000			
1890	011016	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1891	011024	177777	000000	000000		
1892	011032	000000	000000			
1893	011036	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1894	011044	177777	000000	000000		
1895	011052	000000	000000			
1896	011056	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1897	011064	177777	000000	000000		
1898	011072	000000	000000			
1899	011076	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1900	011104	177777	000000	000000		
1901	011112	000000	000000			
1902	011116	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0,0,0
1903	011124	177777	000000	000000		
1904	011132	000000	000000			
1905	011136	177777	177777	177777	.WORD	-1,-1,-1,-1,0,0
1906	011144	177777	000000	000000		
1907	011152					
1908	011152	000004				
1909	011154	010702				
1910	011156	062702	000012			
1911	011162	012707	043764			
1912	011166	000000				

1\$:
RELO: SCOPE
MOV PC,R2
ADD #12,R2
MOV #RELOC,PC ;GO RELOCATE PROGRAM CODE
RELO: .WORD 0

```

1913      :00000000000000 LAST ADDRESS OF CODE TO BE RELOCATED 0000000000
1914
1915      :*****
1916      :*TEST 2      CHECK BRANCH INSTRUCTIONS
1917      :*****
1918      011170 012767 000001 170130      MOV      #1,$TIMES      ;;DO 1 ITERATION
1919      011176 000004      TST2:   SCOPE
1920      011200 112737 000002 001202      MOV      #2,@$STSTNM      ;LOAD TEST NUMBER
1921      011206 013737 001202 177570      MOV      @$STSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER
1922
1923      .SBTTL START OF SECTION 1
1924      ;11111111111111 FIRST ADDRESS TO BE RELOCATED 1111111111
1925      011214 010700      REL1:   MOV      PC,RO      ;GET PC
1926      011216 005740      TST      -(RO)      ;RO CONTAINS THE ADDRESS OF REL1
1927      011220 010037 001540      MOV      RO,@#FRSTAD      ;SAVE
1928      011224 010700      MOV      PC,RO      ;GET CURRENT PC
1929      011226 162700 011226      SUB      #.,RO      ;SUBTRACT RELOCATION FACTOR
1930      011232 010037 001534      MOV      RO,@#FACTOR      ;SAVE RELOCATION FACTOR
1931      011236 010737 001212      MOV      PC,@#SLPERR      ;SET LOOP ADDRESS
1932      011242 062737 000030 001212      ADD      #30,@#SLPERR      ;ADJUST
1933      011250 013737 001212 001210      MOV      @$SLPERR,@#SLPADR
1934      011256 105737 001530      TSTB    @#NEXEC      ;BR IF TEST CODE TO BE EXECUTED
1935      011262 001402      BEQ      .+6
1936      011264 000167 004146      JMP      REL1
1937      ;
1938      011270 000257      CCC      ;CC'S=0000
1939      011272 103407      BCS      CCO      ;SAME AS BLO
1940      011274 102406      BVS      CCO
1941      011276 001405      BEQ      CCO
1942      011300 100404      BMI      CCO
1943      011302 002403      BLT      CCO
1944      011304 003402      BLE      CCO
1945      011306 101401      BLOS     CCO
1946      011310 101001      BHI      .+4
1947      011312 104000      CCO:    HLT      ;ONE OF THE ABOVE BRANCHES FAILED
1948
1949      ;CONTINUE
1950      011314 000270      SEN      ;CC'S=1000
1951      011316 100003      BPL      CC1
1952      011320 002002      BGE      CC1
1953      011322 003001      BGT      CC1
1954      011324 002401      BLT      .+4
1955      011326 104000      CC1:    HLT      ;ONE OF THE ABOVE BRANCHES FAILED
1956
1957      ;CONTINUE
1958      011330 000262      SEV      ;CC'S=1010
1959      011332 102003      BVC      CC2
1960      011334 002402      BLT      CC2
1961      011336 003401      BLE      CC2
1962      011340 002001      BGE      .+4
1963      011342 104000      CC2:    HLT      ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
1964
1965      ;CONTINUE
1966      011344 000261      SEC      ;CC'S=1011
1967      011346 103002      BCC      CC3
1968      011350 101001      E+I     CC3
    
```

```
1969 011352 003001          BGT      .+4
1970 011354 104000          HLT
;CONTINUE
1971
1972
1973 011356 000264          SEZ
1974 011360 001003          BNE      CC4
1975 011362 003002          BGT      CC4
1976 011364 101001          BHI      CC4
1977 011366 003401          BLE      .+4
1978 011370 104000          HLT
;ERROR! ONE OF THE ABOVE BRANCHES FAILED
1979
1980
1981
1982 011372 000004          TST3:   SCOPE
1983 011374 112737 000003 001202      MOV      #3, @RSTSTNM ;LOAD TEST NUMBER
1984 011402 013737 001202 177570      MOV      @RSTSTNM, @RDISPLAY ;:DISPLAY TEST NUMBER
1985
1986 011410 000277          ;CLR    R0
1987 011412 000244          SCC
1988 011414 005000          CLZ
1989 011416 103404          CLR      R0 ;R0=0,CC'S=0100
1990 011420 102403          BCS      CLRO
1991 011422 001002          BVS      CLRO
1992 011424 100401          BNE      CLRO
1993 011426 003401          BMI      CLRO
1994 011430 104000          BLE      .+4
1995
1996 011432 000277          CLRO:   HLT ;ERROR! INCORRECT CC'S AFTER CLR
1997 011434 000244          SCC
1998 011436 005700          CLZ
1999 011440 103404          TST      R0 ;R0=0,CC'S=0100
2000 011442 102403          BCS      TSTO
2001 011444 001002          BVS      TSTO
2002 011446 100401          BNE      TSTO
2003 011450 101401          BMI      TSTO
2004 011452 104000          BLOS     .+4
2005
2006 011454 000257          TSTO:   HLT ;ERROR! INCORRECT CC'S AFTER TST
2007 011456 000266          CCC
2008 011460 005100          +SEZ:SEV
2009 011462 103004          COM      R0 ;R0=-1,CC'S=1001
2010 011464 102403          BCC      COMO
2011 011466 001402          BVS      COMO
2012 011470 100001          BEQ      COMO
2013 011472 002401          BPL      COMO
2014 011474 104000          BLT      .+4
2015
2016 011476 000261          COMO:   HLT ;ERROR! INCORRECT CC'S AFTER COM
2017 011500 005500          SEC
2018 011502 103003          ADC      R0 ;R0=000000,CC'S=0101
2019 011504 102402          BCC      ADCO
2020 011506 001001          BVS      ADCO
2021 011510 002001          BNE      ADCO
2022 011512 104000          BGE      .+4
2023
2024 011514 000261          ADCO:   HLT ;ERROR! INCORRECT CC'S AFTER ADC
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
```

```
2025 011516 006000 ROR RO ;RO=100000,CC'S=1010
2026 011520 103404 BCS RORO
2027 011522 102003 BVC RORO
2028 011524 001402 BEQ RORO
2029 011526 100001 BPL RORO
2030 011530 003001 BGT .+4
2031 011532 104000 RORO: HLT ;ERROR! INCORRECT CC'S AFTER ROR
2032 011534 000277 SCC
2033 011536 000242 CLV
2034 011540 005300 DEC RO ;RO-077777,CC'S=0011
2035 011542 103004 BCC DECO
2036 011544 102003 BVC DECO
2037 011546 001402 BEQ DECO
2038 011550 100401 BMI DECO
2039 011552 003401 BLE .+4
2040 011554 104000 DECO: HLT ;ERROR! INCORRECT CC'S AFTER DEC
2041
2042 011556 000257 CCC
2043 011560 005200 INC RO ;RO-100000,CC'S=1010
2044 011562 103404 BCS INCO
2045 011564 102003 BVC INCO
2046 011566 001402 BEQ INCO
2047 011570 100001 BPL INCO
2048 011572 003001 BGT .+4
2049 011574 104000 INCO: HLT ;ERROR! INCORRECT CC'S AFTER INC
2050
2051 011576 000277 SCC
2052 011600 000242 CLV
2053 011602 005400 NEG RO ;RO=100000,CC'S=1011
2054 011604 103003 BCC NEGO
2055 011606 102002 BVC NEGO
2056 011610 001401 BEQ NEGO
2057 011612 002001 BGE .+4
2058 011614 104000 NEGO: HLT ;ERROR! INCORRECT CC'S AFTER NEG
2059
2060 011616 000261 SEC
2061 011620 006300 ASL RO ;RO=000000,CC'S=0111
2062 011622 103004 BCC ASLO
2063 011624 102003 BVC ASLO
2064 011626 001002 BNE ASLO
2065 011630 100401 BMI ASLO
2066 011632 101401 BLOS .+4
2067 011634 104000 ASLO: HLT ;ERROR! INCORRECT CC'S AFTER ASL
2068
2069 011636 006100 ROL RO ;RO=000001,CC'S=0000
2070 011640 103402 BCS ROLO
2071 011642 003401 BLE ROLO
2072 011644 002001 BGE .+4
2073 011646 104000 ROLO: HLT ;ERROR! INCORRECT CC'S AFTER ROL
2074
2075 011650 006200 ASR RO ;RO=000000,CC'S=0111
2076 011652 103003 BCC ASRO
2077 011654 102002 BVC ASRO
2078 011656 001001 BNE ASRO
2079 011660 002401 BLT .+4
2080 011662 104000 ASRO: HLT ;ERROR! INCORRECT CC'S AFTER ASR
```

```
2081
2082 011664 000277          SCC
2083 011666 005600          SBC      R0          ;R0=-1,CC'S=1001
2084 011670 103002          BCC      SBC0
2085 011672 102401          BVS      SBC0
2086 011674 003401          BLE      .+4
2087 011676 104000          SBC0:  HLT          ;ERROR! INCORRECT CC'S AFTER SBC
2088
2089 011700 005400          NEG      R0          ;R0=000001,CC'S=00001
2090 011702 000300          SWAB     R0          ;R0=000400,CC'S=0100
2091 011704 103403          BCS      SWAB0
2092 011706 102402          BVS      SWAB0
2093 011710 001001          BNE      SWAB0
2094 011712 002001          BGE      .+4
2095 011714 104000          SWAB0: HLT          ;ERROR! INCORRECT CC'S AFTER SWAB
2096
2097
2098
2099 011716 000004          ;*****
2100 011720 112737 000004 001202 ;*TEST 4 CHECK REGISTER SELECTION
2101 011726 013737 001202 177570 ;*****
2102 011734 012737 000005 001326 TST4:  SCOPE
2103 011742 005000          MOVB     #4,@$TSTNM ;LOAD TEST NUMBER
2104 011744 000277          MOV      @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
2105 011746 006100          MOV      #5,@$TIMES ;SET ITERATION COUNT TO 5
2106 011750 010002          CLR      R0
2107 011752 006302          ROL      R0          ;R0-1
2108 011754 010203          ASL      R2          ;R2-2
2109 011756 006303          MOV      R2,R3
2110 011760 010304          ASL      R3          ;R3=4
2111 011762 006304          MOV      R3,R4
2112 011764 010405          ASL      R4          ;R4=10
2113 011766 006305          MOV      R4,R5
2114 011770 010546          ASL      R5          ;R5=20
2115 011772 050416          MOV      R5,-(SP) ;SET BITS SET IN REGISTERS
2116 011774 050316          BIS      R4,(SP) ;INTO STACK ADDRESS
2117 011776 050216          BIS      R3,(SP)
2118 012000 050016          BIS      R2,(SP)
2119 012002 022726 000037          BIS      R0,(SP)
2120 012006 001401          CMP      #37,(SP)+
2121 012010 104000          BEQ      .+4          ;WERE SET
2122
2123
2124 ;CHECK THAT ALL BITS CAN BE SET & CLEARED IN ALL REGISTERS
2125 012012 000257          CCC
2126 012014 112700 000377          MOVB     #377,R0 ;SET ALL BITS (MOVB EXTENDS SIGN)
2127 012020 006100          1$:  ROL      R0          ;ROTATE A 0 THROUGH ALL BIT
2128 012022 103776          BCS      1$          ;POSITIONS
2129 012024 005200          INC      R0          ;FINAL RESULT IS -1
2130 012026 001401          BEQ      .+4
2131 012030 104000          HLT          ;ERROR!
2132
2133 012032 012700 000020          MOV      #16.,R0 ;SET SHIFT COUNT
2134 012036 005002          CLR      R2
2135 012040 000261          2$:  SEC
2136 012042 006002          ROR      R2          ;ROTATE 1 THROUGH ALL BIT POSITS
```

```

2137 012044 005300          DEC      R0          ;DECREMENT SHIFT COUNT
2138 012046 001374          BNE     2$          ;
2139 012050 005102          COM     R2          ;R2 SHOULD CONTAIN -1
2140 012052 001401          BEQ    .+4         ;
2141 012054 104000          HLT                    ;ERROR! CHECK R2 SHOULD 0
2142
2143 012056 012703 100000    MOV     #100000,R3
2144 012062 006203 3$:    ASR     R3          ;EXTEND 1 BIT THROUGH ALL POSITIONS
2145 012064 103376          BCC    3$          ;
2146 012066 005203          INC     R3          ;
2147 012070 001401          BEQ    .+4         ;
2148 012072 104000          HLT                    ;ERROR!
2149
2150 012074 112704 177401    MOVB   #177401,R4   ;R4=1
2151 012100 060404 4$:    ADD     R4,R4       ;HAS THE AFFECT OF SHIFTING A BIT
2152 012102 103376          BCC    4$          ;THROUGH ALL POSITIONS
2153 012104 005704          TST    R4          ;RESULT SHOULD BE 0
2154 012106 001401          BEQ    .+4         ;
2155 012110 104000          HLT                    ;
2156
2157 012112 012705 000001    MOV     #1,R5
2158 012116 006305 5$:    ASL     R5          ;
2159 012120 102376          BVC    5$          ;
2160 012122 006305          ASL     R5          ;
2161 012124 103002          BCC    6$          ;
2162 012126 005705          TST    R5          ;
2163 012130 001401          BEQ    .+4         ;
2164 012132 104000 6$:    HLT                    ;
2165
2166          ;CHECK REGISTER VOLITILITY
2167 012134 005002          CLR     R2          ;
2168 012136 005102          COM     R2          ;R2=-1
2169 012140 010203          MOV     R2,R3      ;
2170 012142 000257          CCC                    ;
2171 012144 006002          ROR     R2          ;R2=LOOP COUNT
2172 012146 006202          ASR     R2          ;
2173 012150 010304 7$:    MOV     R3,R4      ;
2174 012152 005302          DEC     R2          ;DECREMENT LOOP COUNT
2175 012154 001375          BNE     7$          ;
2176 012156 005203          INC     R3          ;CHECK R3
2177 012160 001002          BNE     8$          ;
2178 012162 005204          INC     R4          ;CHECK R4
2179 012164 001401          BEQ    .+4         ;
2180 012166 104000 8$:    HLT                    ;
2181
2182          ;CHECK TRANSFER OF REGISTER DATA BETWEEN THE GS AND GD REGISTERS
2183 012170 032737 000020 177776 GSTST: BIT     #20,2$PSW ;CHECK IF 'T' BIT IS SET
2184 012176 001050          BNE     7$          ;SKIP TEST IF 'T' BIT SET
2185 012200 010627          MOV     SP,(PC)+   ;SAVE STACK PTR
2186 012202 000000 1$:    .WORD 0          ;CONTAINS SAVED STACK PTR
2187 012204 010727          MOV     PC,(PC)+   ;LOAD DATA. THE CURRENT PC IS USED AS
2188 012206 000000 2$:    .WORD 0          ;DATA. IF THIS TEST FAILS 2$ CON-
2189          ;TAINS THE DATA BEING USED.
2190 012210 005267 177772          INC     2$          ;MAKE ODD TO CHECK BIT 0
2191 012214 016700 177766 3$:    MOV     2$,R0      ;LOAD GD REGISTER 0
2192 012220 010001          MOV     R0,R1      ;TRANSFER GS REG 0 TO GD REG 1
    
```

```
2193 012222 010102      MOV      R1,R2      ;AND GS REG 1 TO GD REG 2
2194 012224 010203      MOV      R2,R3      ;ETC...
2195 012226 010304      MOV      R3,R4
2196 012230 010405      MOV      R4,R5
2197 012232 152737 000340 177776  BISB     #340,@#PSW   ;SET PRIORITY LEVEL 7
2198 012240 010506      MOV      R5,SP      ;TRANSFER GS REG 5 TO GD STK PTR
2199 012242 010627      MOV      SP,(PC)+   ;TRANSFER GS STK PTR TO MEMORY
2200 012244 000000      .WORD   0          ;CONTAINS GS STACK PTR
2201 012246 016706 177730      MOV      1$,SP      ;RESTORE STK PTR NEEDED FOR HLT/SCOPE
2202 012252 142737 000340 177776  BICB     #340,@#PSW   ;SET PRIORITY LEVEL 0
2203 012260 026700 177760      CMP      4$,R0      ;COMPARE GS/GD STACK WITH GS REG 0
2204 012264 001004      BNE      5$         ;BRANCH IF THEY WERE NOT -
2205 012266 006367 177714      ASL      2$         ;SHIFT TEST DATA UNTIL = 000000
2206 012272 001350      BNE      3$
2207 012274 000411      BR       6$
2208 012276 010046      MOV      R0,-(SP)   ;GET GS REG 0
2209 012300 010146      MOV      R1,-(SP)   ;ETC...
2210 012302 010246      MOV      R2,-(SP)
2211 012304 010346      MOV      R3,-(SP)
2212 012306 010446      MOV      R4,-(SP)
2213 012310 010546      MOV      R5,-(SP)
2214 012312 104000      HLT
2215                                ;ERROR! DATA IN GS STK PTR NOT - GS REG 0
2216 012314 016706 177662      MOV      1$,SP      ;GS REG 0-GS REG 5 ARE ON THE STACK
2217                                ;RESTORE STACK PTR
2218                                6$:
2219                                7$:
2220                                ;*****
2221                                ;*TEST 5 TEST UNIARY WORD INSTRUCTIONS USING ADDRESS MODE 1
2222                                ;*****
2223 012320 000004      TST5:   SCOPE
2224 012322 112737 000005 001202  MOVB     #5,@#STSTNM ;LOAD TEST NUMBER
2225 012330 013737 001202 17757C  MOV      @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
2226 012336 012737 000005 001326  MOV      #5,@#TIMES
2227 012344 000401      BR       .+4
2228 012346 000000      .WORD   0          ;RESERVE ADDRESS FOR TESTS
2229 012350 010702      MOV      PC,R2
2230 012352 162702 000004  SUB      #4,R2      ;R2 POINTS TO RESERVED WORD
2231 012356 005012      CLR      (R2)      ;PRESET (R2)
2232 012360 000261      SEC
2233 012362 006012      ROR      (R2)      ;(R2)=100000,CC=1010
2234 012364 101402      BLOS     ROR1
2235 012366 100001      BPL     ROR1
2236 012370 002001      BGE     .+4
2237 012372 104000      ROR1:   HLT        ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2238
2239 012374 000257      CCC
2240 012376 000261      SEC
2241 012400 005312      DEC      (R2)      ;(R2)=077777,CC=0011
2242 012402 103001      BCC     DEC1
2243 012404 003401      BLE     .+4
2244 012406 104000      DEC1:   HLT        ;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2245
2246 012410 000257      CCC
2247 012412 000261      SEC
2248 012414 005512      ADC      (R2)      ;(R2)=100000,CC=1010
```


2249	012416	103403	BCS	ADC1	
2250	012420	102002	BVC	ADC1	
2251	012422	100001	BPL	ADC1	
2252	012424	001001	BNE	+.4	
2253	012426	104000	ADC1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2254					
2255	012430	006112	ROL	(R2)	;(R2)=000000,CC=0111
2256	012432	103003	BCC	ROL1	
2257	012434	102002	BVC	ROL1	
2258	012436	001001	BNE	ROL1	
2259	012440	100001	BPL	+.4	
2260	012442	104000	ROL1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2261					
2262	012444	006112	ROL	(R2)	;(R2)=000001,CC=0000
2263	012446	101402	BLOS	ROL1A	;BRANCH IF C OR Z IS SET
2264	012450	102401	BVS	ROL1A	
2265	012452	100001	BPL	+.4	
2266	012454	104000	ROL1A:	HLT	
2267					
2268	012456	006212	ASR	(R2)	;(R2)=000000,CC=0111
2269	012460	103003	BCC	ASR1	
2270	012462	102002	BVC	ASR1	
2271	012464	001001	BNE	ASR1	
2272	012466	100001	BPL	+.4	
2273	012470	104000	ASR1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2274					
2275	012472	006012	ROR	(R2)	;(R2)=100000,CC=1010
2276	012474	103403	BCS	ROR1A	
2277	012476	102002	BVC	ROR1A	
2278	012500	001401	BEQ	ROR1A	
2279	012502	100401	BMI	+.4	
2280	012504	104000	ROR1A:	HLT	
2281					
2282	012506	000261	SEC		
2283	012510	005212	INC	(R2)	;(R2)=100001,CC=1001
2284	012512	103003	BCC	INC1	
2285	012514	102402	BVS	INC1	
2286	012516	001401	BEQ	INC1	
2287	012520	100401	BMI	+.4	
2288	012522	104000	INC1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2289					
2290	012524	005612	SBC	(R2)	;(R2)=100000,CC=1000
2291	012526	103403	BCS	SBC1	
2292	012530	102402	BVS	SBC1	
2293	012532	001401	BEQ	SBC1	
2294	012534	100401	BMI	+.4	
2295	012536	104000	SBC1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2296					
2297	012540	000261	SEC		
2298	012542	005612	SBC	(R2)	;(R2)=077777,CC=0010
2299	012544	103403	BCS	SBC1A	
2300	012546	102002	BVC	SBC1A	
2301	012550	001401	BEQ	SBC1A	
2302	012552	100001	BPL	+.4	
2303	012554	104000	SBC1A:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2304					

```
2305 012556 000261 SEC
2306 012560 005512 ADC (R2) ;(R2)=100000,CC=1010
2307 012562 100401 BMI .+4
2308 012564 104000 HLT
2309
2310 012566 000261 SEC
2311 012570 006312 ASL (R2) ;(R2)=000000,CC=0111
2312 012572 103003 BCC ASL1
2313 012574 102002 BVC ASL1
2314 012576 001001 BNE ASL1
2315 012600 100001 BPL .+4
2316 012602 104000 ASL1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2317
2318 012604 005112 COM (R2) ;(R2)=177777,CC=1001
2319 012606 103002 BCC COM1
2320 012610 102401 BVS COM1
2321 012612 100401 BMI .+4
2322 012614 104000 COM1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2323
2324 012616 000250 CLN
2325 012620 005712 TST (R2) ;(R2)=177777,CC=1000
2326 012622 103403 BCS TEST1
2327 012624 102402 BVS TEST1
2328 012626 100001 BPL TEST1
2329 012630 001001 BNE .+4
2330 012632 104000 TEST1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2331
2332 012634 000262 SEV
2333 012636 005412 NEG (R2) ;(R2)=000001,CC=0000
2334 012640 103002 BCC NEG1
2335 012642 102401 BVS NEG1
2336 012644 001001 BNE .+4
2337 012646 104000 NEG1: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2338
2339 012650 005312 DEC (R2) ;(R2)=000000,CC=0101
2340 012652 103001 BCC DEC1A
2341 012654 001401 BEQ .+4
2342 012656 104000 DEC1A: HLT ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2343
2344
2345
2346 012660 000004 TST6: SCOPE
2347 012662 112737 000006 001202 MOVB #6, @#STSTNM ;LOAD TEST NUMBER
2348 012670 013737 001202 177570 MOV @#STSTNM, @#DISPLAY ;:DISPLAY TEST NUMBER
2349 012676 000401 BR .+4 ;RESERVE A WORD
2350 012700 000000 .WORD 0 ;ADDRESS RESERVED FOR TESTS
2351 012702 010703 MOV PC, R3
2352 012704 162703 000004 SUB #4, R3 ;R3 POINTS TO EVEN BYTE OF WORD
2353 012710 010304 MOV R3, R4 ;R4 POINTS TO ODD BYTE OF WORD
2354 012712 005204 INC R4
2355 012714 005013 CLR (R3) ;PRESET DATA
2356
2357 012716 000261 18: SEC
2358 012720 105513 ADCB (R3) ;ADD CARRY TO EVEN BYTE
2359 012722 100402 BMI 28 ;UNTIL EVEN BYTE BECOMES NEGATIVE
2360 012724 105214 INCB (R4) ;INCREMENT ODD BYTE
```

2361	012726	000773		BR	1\$	
2362	012730	102401	2\$:	BVS	+.4	;(R3)=077600=[0774][200],CC=1010
2363	012732	104000		HLT		
2364	012734	000242		CLV		
2365	012736	105214		INCB	(R4)	;(R3)=100200=[1000][200],CC=1010
2366	012740	103402		BCS	INCB1	
2367	012742	102001		BVC	INCB1	
2368	012744	100401		BMI	+.4	
2369	012746	104000	INCB1:	HLT		;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2370						
2371	012750	106114		ROLB	(R4)	;(R3)=000200=[0000][200],CC=0111
2372	012752	103002		BCC	ROLB1	
2373	012754	102001		BVC	ROLB1	
2374	012756	001401		BEQ	+.4	
2375	012760	104000	ROLB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2376						
2377	012762	105614		SBCB	(R4)	;(R3)=177600=[1774][200], CC-1001
2378	012764	103002		BCC	SBCB1	
2379	012766	102401		BVS	SBCB1	
2380	012770	100401		BMI	+.4	
2381	012772	104000	SBCB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2382						
2383	012774	106313		ASLB	(R3)	;(R3)=177400,CC=0111
2384	012776	103002		BCC	ASLB1	
2385	013000	102001		BVC	ASLB1	
2386	013002	001401		BEQ	+.4	
2387	013004	104000	ASLB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2388						
2389	013006	105413		NEGB	(R3)	;(R3)=177400,CC=0100
2390	013010	103402		BCS	NEGB1	
2391	013012	102401		BVS	NEGB1	
2392	013014	001401		BEQ	+.4	
2393	013016	104000	NEGB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2394						
2395	013020	000277		SCC		
2396	013022	105313		DECB	(R3)	;(R3)=177777,CC-1001
2397	013024	103002		BCC	DECB1	
2398	013026	102401		BVS	DECB1	
2399	013030	001001		BNE	+.4	
2400	013032	104000	DECB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2401						
2402	013034	000241		CLC		
2403	013036	106013		RORB	(R3)	;(R3)=177577,CC=0011
2404	013040	103002		BCC	RORB1	
2405	013042	102001		BVC	RORB1	
2406	013044	100001		BPL	+.4	
2407	013046	104000	RORB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2408						
2409	013050	000241		CLC		
2410	013052	105114		COMB	(R4)	;(R3)=000177,CC=0101
2411	013054	103002		BCC	COMB1	
2412	013056	102401		BVS	COMB1	
2413	013060	001401		BEQ	+.4	
2414	013062	104000	COMB1:	HLT		;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2415						
2416	013064	106213	1\$:	ASRB	(R3)	;SHIFT EVEN BYTE UNTIL V CLEARS

2417	013066	102002	BVC	2\$	
2418	013070	105514	ADCB	(R4)	;AND ADD CARRY TO ODD BYTE
2419	013072	000774	BR	1\$	
2420	013074	103401	2\$: BCS	ASRB1	
2421	013076	001401	BEQ	+.4	
2422	013100	104000	ASRB1: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2423					
2424	013102	106214	ASRB	(R4)	
2425	013104	106214	ASRB	(R4)	;(R3)=0004C0,CC=0011
2426	013106	103002	BCC	ASRB1A	
2427	013110	102001	BVC	ASRB1A	
2428	013112	001001	BNE	+.4	
2429	013114	104000	ASRB1A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2430					
2431	013116	105314	DECB	(R4)	;(R3)=000000,CC=0100
2432	013120	001401	BEQ	+.4	
2433	013122	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2434					
2435	013124	000261	SEC		
2436	013126	106014	RORB	(R4)	;(R3)=100000,CC=1010
2437	013130	103402	BCS	RORB1A	
2438	013132	102001	BVC	RORB1A	
2439	013134	100401	BMI	+.4	
2440	013136	104000	RORB1A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2441					
2442	013140	000242	CLV		
2443	013142	105314	DECB	(R4)	;(R3)=077400,CC=0100
2444	013144	102401	BVS	+.4	
2445	013146	104000	HLT		
2446					
2447	013150	000261	SEC		
2448	013152	105313	DECB	(R3)	;(R3)=077777,CC=1001
2449	013154	103002	BCC	DECB1A	
2450	013156	102401	BVS	DECB1A	
2451	013160	100401	BMI	+.4	
2452	013162	104000	DECB1A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2453					
2454	013164	000277	SCC		
2455	013166	000313	SWAB	(R3)	;(R3)=177577=[1774][177],CC=0000
2456	013170	103402	BCS	SWAB1	
2457	013172	102401	BVS	SWAB1	
2458	013174	100001	BPL	+.4	
2459	013176	104000	SWAB1: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2460					
2461	013200	105714	TSTB	(R4)	;(R3)=177577=[1774][177],CC=1000
2462	013202	103402	BCS	TSTB1	
2463	013204	102401	BVS	TSTB1	
2464	013206	100401	BMI	+.4	
2465	013210	104000	TSTB1: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2466					
2467	013212	105014	CLRB	(R4)	;(R3)=000177=[0000][177],CC=0100
2468	013214	001401	BEQ	+.4	
2469	013216	104000	HLT		
2470	013220	106313	ASLB	(R3)	;(R3)=000376,CC=1010
2471	013222	103402	BCS	ASLB1A	
2472	013224	102001	BVC	ASLB1A	

```
2473 013226 100401      BMI      .+4
2474 013230 104000      ASLB1A: HLT                      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2475
2476 013232 105113      COMB      (R3)                    ;(R3)=000001,CC=0001
2477 013234 103002      BCC      COMB1A
2478 013236 102401      BVS      COMB1A
2479 013240 100001      BPL      .+4
2480 013242 104000      COMB1A: HLT                      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2481
2482 013244 000313      SWAB      (R3)                    ;(R3)=000400, CC=0100
2483 013246 001401      BEQ      .+4
2484 013250 104000      HLT
2485
2486 013252 105213      INCB      (R3)
2487 013254 000261      SEC
2488 013256 105613      SBCB      (R3)                    ;(R3)=000400,CC=0100
2489 013260 001401      BEQ      .+4
2490 013262 104000      HLT
2491 013264 022713 000400      CMP      #400,(R3)                ;CHECK REMAINING RESULT
2492 013270 001401      BEQ      .+4
2493 013272 104000      HLT
2494
2495
2496
2497 013274 000004      TST7: SCOPE
2498 013276 112737 000007 001202      MOV      #7,@#STSTNM              ;LOAD TEST NUMBER
2499 013304 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
2500 013312 000401      BR      .+4
2501 013314 000000      .WORD 0                          ;ADDRESS RESERVED FOR TESTS
2502 013316 010704      MOV      PC,R4
2503 013320 162704 000004      SUB      #4,R4                    ;R4 AND R5 POINT TO
2504 013324 010405      MOV      R4,R5                    ;RESERVED WORD
2505 013326 005015      CLR      (R5)                     ;PRESET DATA=0
2506
2507 013330 000277      SCC
2508 013332 000244      CLZ
2509 013334 005725      TST      (R5)+                    ;(R5)=000000,CC=0100
2510 013336 103402      BCS      TEST2
2511 013340 102401      BVS      TEST2
2512 013342 001401      BEQ      .+4
2513 013344 104000      TEST2: HLT                      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2514
2515 013346 005145      COM      -(R5)                    ;(R5)=177777,CC=1001
2516 013350 103001      BCC      COM4
2517 013352 100401      BMI      .+4
2518 013354 104000      COM4: HLT                      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2519
2520 013356 000241      CLC
2521 013360 006024      ROR      (R4)+                    ;(R4)=077777,CC=0011
2522 013362 103002      BCC      ROR2
2523 013364 102001      BVC      ROR2
2524 013366 100001      BPL      .+4
2525 013370 104000      ROR2: HLT                      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2526
2527 013372 000257      CCC
2528 013374 005244      INC      -(R4)                    ;(R4)-100000,CC=1010
```

2529	013376	102002	BVC	INC4	
2530	013400	001401	BEQ	INC4	
2531	013402	100401	BMI	+.4	
2532	013404	104000	INC4:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2533					
2534	013406	000261	SEC		
2535	013410	000324	SWAB	(R4)+	; (R4)=0002C0,CC=1000
2536	013412	103401	BCS	SWAB2	
2537	013414	100401	BMI	+.4	
2538	013416	104000	SWAB2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2539					
2540	013420	005425	NEG	(R5)+	; (R5)=177600,CC=1001
2541	013422	103001	BCC	NEG2	
2542	013424	100401	BMI	+.4	
2543	013426	104000	NEG2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2544					
2545	013430	005044	CLR	-(R4)	; (R4)=000000,CC=0100
2546	013432	001401	BEQ	+.4	
2547	013434	104000	HLT		
2548					
2549	013436	000261	SEC		
2550	013440	006045	ROR	-(R5)	; (R5)=100000,CC=1010
2551	013442	000261	SEC		
2552	013444	005525	ADC	(R5)+	; (R5)=100001,CC=1000
2553	013446	102401	BVS	ADC2	
2554	013450	100401	BMI	+.4	
2555	013452	104000	ADC2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2556					
2557	013454	000262	SEV		
2558	013456	006224	ASR	(R4)+	; (R4)=140000,CC=1001
2559	013460	103002	BCC	ASR2	
2560	013462	102401	BVS	ASR2	
2561	013464	100401	BMI	+.4	
2562	013466	104000	ASR2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2563					
2564	013470	000262	SEV		
2565	013472	006144	ROL	-(R4)	; (R4)=100001,CC=1001
2566	013474	103002	BCC	ROL4	
2567	013476	102401	BVS	ROL4	
2568	013500	100401	BMI	+.4	
2569	013502	104000	ROL4:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2570					
2571	013504	005645	SBC	-(R5)	; (R5)=100000,CC=1000
2572	013506	103001	BCC	+.4	
2573	013510	104000	HLT		;ERROR! 'C' BIT FAILED TO CLEAR
2574					
2575	013512	005325	DEC	(R5)+	; (R5)=077777,CC=0010
2576	013514	103402	BCS	DEC2	
2577	013516	102001	BVC	DEC2	
2578	013520	100001	BPL	+.4	
2579	013522	104000	DEC2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2580					
2581	013524	006324	ASL	(R4)+	; (R4)=177776,CC=1010
2582	013526	102401	BVS	+.4	
2583	013530	104000	HLT		
2584	013532	006344	ASL	-(R4)	; (R4)=177774,CC=1001

```
2585 013534 103003          BCC    ASL4
2586 013536 102402          BVS    ASL4
2587 013540 001401          BEQ    ASL4
2588 013542 100401          BMI    .+4
2589 013544 104000          ASL4:  HLT                    ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2590
2591 013546 022724 177774      CMP    #177774,(R4)+
2592 013552 001401          BEQ    .+4
2593 013554 104000          HLT
2594 013556 020405          CMP    R4,R5
2595 013560 001401          BEQ    .+4
2596 013562 104000          HLT
2597
2598
2599
2600 013564 000004          :*****
2601 013566 112737 000010 001202      :*TEST 10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4
2602 013574 013737 001202 177570      :*****
2603 013602 000401          TST10: SCOPE
2604 013604 000000          MOV    #10,#STSTNM          ;LOAD TEST NUMBER
2605 013606 010705          MOV    @STSTNM,@DISPLAY    ;:DISPLAY TEST NUMBER
2606 013610 162705 000004          BR     .+4                  ;RESERVE A WORD
2607 013614 010500          .WORD 0                    ;RESERVED WORD
2608 013616 010002          MOV    PC,R5
2609 013620 005202          SUB    #4,R5                ;R5 POINTS TO EVEN BYTE OF RESERVED WORD
2610 013622 005010          MOV    R5,R0
2611
2612 013624 000277          MOV    R0,R2
2613 013626 000241          INC    R2                    ;R2 POINTS TO ODD BYTE OF RESERVED WORD
2614 013630 105125          CLR    (R0)                 ;PRESET
2615 013632 103002          SCC
2616 013634 102401          CLC
2617 013636 100401          COMB (R5)+                 ;(R0)=000377,CC=1001
2618 013640 104000          BCC   COMB2
2619
2620 013642 105542          BVS   COMB2
2621 013644 001401          BMI  .+4
2622 013646 104000          COMB2: HLT                  ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2623 013650 105525          ADCB  -(R2)                 ;(R0)=000000,CC=0101
2624 013652 103401          BEQ  .+4
2625 013654 001001          HLT                          ;ERROR! INCORRECT RESULT AS SHOWN ABOVE
2626 013656 104000          ADCB (R5)+                 ;(R0)=000400,CC=0000
2627
2628 013660 000263          BNE  .+4
2629 013662 106045          ADCB2: HLT                  ;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2630 013664 103003          +SEC!SEV
2631 013666 102402          RORB -(R5)                 ;(R0)-100000,CC-1001
2632 013670 001401          BCC  RORB4
2633 013672 100401          BVS  RORB4
2634 013674 104000          BEQ  RORB4
2635
2636 013676 000277          BMI  .+4
2637 013700 106122          RORB4: HLT                  ;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2638 013702 103403          SCC
2639 013704 102402          ROLB (R2)+                 ;(R0)=100001,CC-0000
2640 013706 001401          BCS  ROLB2
          BVS  ROLB2
          BEQ  ROLB2
```

2641	013710	100001			
2642	013712	104000	ROLB2:	BPL .+4	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2643					
2644	013714	000257		CCC	
2645	013716	106225		ASRB (R5)+	;(R0)=140001, CC=1010
2646	013720	103402		BCS ASRB2	
2647	013722	102001		BVC ASRB2	
2648	013724	100401		BMI .+4	
2649	013726	104000	ASRB2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2650					
2651	013730	105242		INCB -(R2)	;(R0)=140002, CC=0000
2652	013732	000277		SCC	
2653	013734	106222		ASRB (R2)+	;(R0)=140001, CC=0000
2654	013736	103402		BCS ASRB2A	
2655	013740	102401		BVS ASRB2A	
2656	013742	100001		BPL .+4	
2657	013744	104000	ASRB2A:	HLT	;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2658					
2659	013746	000266		+SEZ .SEV	;SET Z,V
2660	013750	106345		ASLB -(R5)	;(R0)=100001, CC=1001
2661	013752	103003		BCC ASLB4	
2662	013754	102402		BVS ASLB4	
2663	013756	001401		BEQ ASLB4	
2664	013760	100401		BMI .+4	
2665	013762	104000	ASLB4:	HLT	;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2666					
2667	013764	105322		DECB (R2)+	;(R0)-077401=[0774][001] ,CC=0010
2668	013766	103002		BCC DECB2	
2669	013770	102001		BVC DECB2	
2670	013772	100001		BPL .+4	
2671	013774	104000	DECB2:	HLT	;ERRUP! INCORRECT CC'S AS SHOWN ABOVE
2672					
2673	013776	105645		SBCB -(R5)	;(R0)=077400, CC=0100
2674	014000	103402		BCS SBCB4	
2675	014002	102401		BVS SBCB4	
2676	014004	001401		BEQ .+4	
2677	014006	104000	SBCB4:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2678					
2679	014010	105442		NEGB -(R2)	;(R0)-10400, CC=1001
2680	014012	103002		BCC NEGB4	
2681	014014	102401		BVS NEGB4	
2682	014016	100401		BMI .+4	
2683	014020	104000	NEGB4:	HLT	;ERROR. INCORRECT CC'S AS SHOWN ABOVE
2684					
2685	014022	105725		TSTB (R5)+	;(R0)=100400, CC=0100
2686	014024	103401		BCS TSTB2	
2687	014026	001401		BEQ .+4	
2688	014030	104000	TSTB2:	HLT	
2689					
2690	014032	105722		TSTB (R2)+	;(R0)=100400, CC=1000
2691	014034	001401		BEQ TSTB2A	
2692	014036	100401		BMI .+4	
2693	014040	104000	TSTB2A:	HLT	
2694					
2695	014042	000261		SEC	
2696	014044	000342		SWAB -(R2)	;(R0)=000201, CC=1000

2697 014046 103401
2698 014050 100401
2699 014052 104000
2700
2701 014054 000277
2702 014056 105225
2703 014060 103003
2704 014062 102402
2705 014064 001401
2706 014066 100001

BCS SWAB4
BMI .+4
SWAB4: HLT
SCC
INCB (R5)+
BCC INCB2
BVS INCB2
BEQ INCB2
BPL .+4

;(R0)-000601=[0004][201],CC=0000

DEQKC-D PDF 11/70-74MP CPU EXERCISER
CEQKCD.P11 04-OCT-79 08:55

MACY11 30A(1052) 04-OCT-79^{C 7} 09:00 PAGE 58
T10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 8 4

SEQ 0080

2707 014070 104.000

INCB2: HLT

DEQKC-D PDP 11/70-74MP CPU EXERCISER
CEQKCD.P11 04-OCT-79 08:55

MACY11 30A(1052) 04-OCT-79^{D 7} 09:00 PAGE 59
T10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4

SEQ 0081

2708

2709 014072 022227 000601
2710 014076 001401
2711 014100 104000
2712 014102 020205
2713 014104 001401
2714 014106 104000
2715
2716
2717
2718 014110 000004
2719 014112 112737 000011 001202
2720 014120 013737 001202 177570
2721 014126 000402
2722 014130 000000
2723 014132 000000
2724 014134 010703
2725 014136 162703 000004
2726 014142 005013
2727 014144 010300
2728 014146 005743
2729 014150 010013
2730 014152 010304
2731
2732 014154 000257
2733 014156 005733
2734 014160 001401
2735 014162 104000
2736
2737 014164 000261
2738 014166 006053
2739 014170 103402
2740 014172 102001
2741 014174 100401
2742 014176 104000
2743
2744 014200 000257
2745 014202 006234
2746 014204 102001
2747 014206 100401
2748 014210 104000
2749
2750 014212 000250
2751 014214 006333
2752 014216 103002
2753 014220 102401
2754 014222 100401
2755 014224 104000
2756
2757 014226 000277
2758 014230 005354
2759 014232 103003
2760 014234 102002
2761 014236 001401
2762 014240 100001
2763 014242 104000
2764

```

CMP      (R2)+,#000601 ;CHECK END RESULT
BEQ      +4
HLT
CMP      R2,R5          ;CHECK REGISTERS
BEQ      .+4
HLT
:*****
:*TEST 11 CHECK UNIARY WORD OPS USING ADDRESS MODES 3 & 5
:*****
TST11:  SCOPE
        MOV# #11,@TSTSTNM ;LOAD TEST NUMBER
        MOV @TSTSTNM,@NDISPLAY ;:DISPLAY TEST NUMBER
        BR .+6 ;RESERVE 2 WORDS
        .WORD 0 ;1 FOR THE ADDRESS
        .WORD 0 ;AND 1 FOR DATA
        MOV PC,R3
        SUB #4,R3
        CLR (R3) ;PRESET DATA
        MOV R3,R0 ;R0 POINTS TO DATA WORD
        TST -(R3)
        MOV R0,(R3)
        MOV R3,R4
        CCC
        TST @-(R3)+ ;(R0)=000000,CC=0100
        BEQ .+4
        HLT
        SEC
        ROR @-(R3) ;(R0)=100000,CC=1010
        BCS ROR5
        BVC ROR5
        BMI .+4
ROR5:  HLT
        CCC
        ASR @-(R4)+ ;(R0)=140000,CC=1010
        BVC ASR3
        BMI .+4
ASR3:  HLT
        CLN
        ASL @-(R3)+ ;(R0)=100000,CC=1001
        BCC ASL3
        BVS ASL3
        BMI .+4
ASL3:  HLT
        SCC
        DEC @-(R4) ;(R0)=077777, CC=0010
        BCC DEC5
        BVC DEC5
        BEQ DEC5
        BPL .+4
DEC5:  HLT
```

```
2765 014244 005453      NEG      @-(R3)          ;(R0)=100001, CC=1001
2766 014246 103002      BCC      NEG5
2767 014250 102401      BVS      NEG5
2768 014252 100401      BMI      .+4
2769 014254 104000      NEG5:   HLT
2770
2771 014256 000262      SEV
2772 014260 005134      COM      @-(R4)+        ;(R0)=077776, CC=0001
2773 014262 103001      BCC      COM3
2774 014264 102001      BVC      .+4
2775 014266 104000      COM3:   HLT
2776
2777 014270 005233      INC      @-(R3)+        ;(R0)=077777, CC=0001
2778 014272 103001      BCC      INC3
2779 014274 100001      BPL      .+4
2780 014276 104000      INC3:   HLT
2781
2782 014300 005554      ADC      @-(R4)         ;(R0)=100000, CC=1010
2783 014302 103402      BCS      ADC5
2784 014304 102001      BVC      ADC5
2785 014306 100401      BMI      .+4
2786 014310 104000      ADC5:   HLT
2787
2788 014312 000257      CCC
2789 014314 006134      ROL      @-(R4)+        ;(R0)-000000,CC=0111
2790 014316 103002      BCC      ROL3
2791 014320 102001      BVC      ROL3
2792 014322 001401      BEQ      .+4
2793 014324 104000      ROL3:   HLT
2794
2795 014326 005253      INC      @-(R3)         ;(R0)=000001, CC=0001
2796 014330 005654      SBC      @-(R4)         ;(R0)=000000, CC=0100
2797 014332 103401      BCS      SBC5
2798 014334 001401      BEQ      .+4
2799 014336 104000      SBC5:   HLT
```

```
*****
:*TEST 12 CHECK UNIARY BYTE OPS USING ADDRESS MODES 3 & 5
*****
```

```
TST12: SCOPE
2803 014340 000004      MOV      #12,@$TSTNM    ;LOAD TEST NUMBER
2804 014342 112737 000012 001202      MOV      @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
2805 014350 013737 001202 177570      BR       .+10           ;RESERVE 3 WORDS
2806 014356 000403      .WORD   0               ;1 FOR EVEN BYTE ADDRESS
2807 014360 000000      .WORD   0               ;1 FOR ODD BYTE ADDRESS
2808 014362 000000      .WORD   0               ;AND 1 FOR DATA
2809 014364 000000      MOV      PC,R2
2810 014366 010702      TST     -(R2)           ;BACK R2 UP TO
2811 014370 005742      TST     -(R2)           ;DATA WORD
2812 014372 005742      MOV     R2,R0           ;R0 POINTS TO THE DATA WORD
2813 014374 010200      CLR     (R0)           ;PRESET DATA
2814 014376 005010      TST     -(R2)           ;BACK R2 UP TO
2815 014400 005742      TST     -(R2)           ;EVEN BYTE ADDRESS WORD
2816 014402 005742      MOV     R0,(R2)+       ;LOAD ADDRESS
2817 014404 010022      INC     R0              ;ODD BYTE ADDRESS
2818 014406 005200      MOV     R0,(R2)+       ;LOAD ODD BYTE ADDRESS
2819 014410 010022      MOV     R2,R0          ;RESET R0
2820 014412 010200
```

2821	014414	010205	MOV	R2,R5		
2822	014416	105152	COMB	@-(R2)	;(R0)=177400,CC=1001	
2823	014420	103001	BCC	COMB5		
2824	014422	100401	BMI	+.4		
2825	014424	104000	COMB5:	HLT		
2826	014426	105752	TSTB	@-(R2)	;(R0)=177400,CC=0100	
2827	014430	001401	BEQ	+.4		
2828	014432	104000	HLT			
2829	014434	000262	SEV			
2830	014436	106255	ASRB	@-(R5)	;(R0)=177400,CC=1001	
2831	014440	103002	BCC	ASRB5		
2832	014442	102401	BVS	ASRB5		
2833	014444	100401	BMI	+.4		
2834	014446	104000	ASRB5:	HLT		
2835						
2836	014450	105232	INCB	@(R2)+	;(R0)=177401,CC=000	
2837	014452	103001	BCC	INCB3		
2838	014454	100001	BPL	+.4		
2839	014456	104000	INCB3:	HLT		
2840						
2841	014460	000241	CLC			
2842	014462	106055	RORB	@-(R5)	;(R0)=177400,CC=0111	
2843	014464	103003	BCC	RORB5		
2844	014466	102002	BVC	RORB5		
2845	014470	001001	BNE	RORB5		
2846	014472	100001	BPL	+.4		
2847	014474	104000	RORB5:	HLT		
2848						
2849	014476	106332	ASLB	@(R2)+	;(R0)=177000,CC=1001	
2850	014500	103002	BCC	ASLB3		
2851	014502	102401	BVS	ASLB3		
2852	014504	100401	BMI	+.4		
2853	014506	104000	ASLB3:	HLT		
2854						
2855	014510	105552	ADCB	@-(R2)	;(R0)=177400,CC=1000	
2856	014512	103401	BCC	ADCB5		
2857	014514	100401	BMI	+.4		
2858	014516	104000	ADCB5:	HLT		
2859						
2860	014520	000277	SCC			
2861	014522	106135	ROLB	@(R5)+	;(R0)=177401,CC=0000	
2862	014524	101402	BLOS	ROLB3	;(R0)=177401,CC=0000	
2863	014526	102401	BVS	ROLB3	;(R0)=177401,CC=0000	
2864	014530	100001	BPL	+.4	;(R0)=177401,CC=0000	
2865	014532	104000	ROLB3:	HLT	;(R0)=177401,CC=0000	
2866						
2867	014534	000352	SWAB	@-(R2)	;(R0)=000777,CC=1000	
2868	014536	100401	BMI	+.4		
2869	014540	104000	HLT			
2870						
2871	014542	000261	SEC			
2872	014544	105635	SBCB	@(R5)+	;(R0)=000377,CC=0100	
2873	014546	103401	BCC	SBCB3		
2874	014550	001401	BEQ	+.4		
2875	014552	104000	SBCB3:	HLT		
2876						

```
2877 014554 105432          NEGB @ (R2)+          ;(R0)=000001
2878 014556 105352          DECB @-(R2)          ;(R0)=000000, CC=0101
2879 014560 103001          BCC  DECB5
2880 014562 001401          BEQ  .+4
2881 014564 104000          DECB5: HLT
2882
2883
2884
2885 014566 000004          *TEST 13 CHECK UNIARY WORD OPS USING ADDRESS MODE 6 (PC)
2886 014570 112737 000013 001202          TST13: SCOPE
2887 014576 013737 001202 177570          MOV  #13, @RSTSTNM ;LOAD TEST NUMBER
2888 014604 005027          MOV  @RSTSTNM, @RDISPLAY ;:DISPLAY TEST NUMBER
2889 014606 000000          CLR  (PC)+          ;PRESET DATA = 0
2890 014610 010700          UWM6: .WORD 0 ;RESERVED FOR DATA
2891 014612 024040          MOV  PC, R0
2892 014614 000277          CMP  -(R0), -(R0) ;R0 POINTS TO DATA WORD
2893 014616 006167 177764          SCC
2894 014622 103403          ROL  UWM6          ;(R0)=000001, CC=0000
2895 014624 102402          BCS  ROL6
2896 014626 001401          BVS  ROL6
2897 014630 100001          BEQ  ROL6
2898 014632 104000          BPL  .+4
2899          ROL6: HLT
2900 014634 005167 177746          COM  UWM6          ;(R0)=177776, CC=1001
2901 014640 103002          BCC  COM6
2902 014642 102401          BVS  COM6
2903 014644 100401          BMI  .+4
2904 014646 104000          COM6: HLT
2905 014650 006267 177732          ASR  UWM6          ;(R0)=177777, CC=1010
2906 014654 103402          BCS  ASR6
2907 014656 102001          BVC  ASR6
2908 014660 100401          BMI  .+4
2909 014662 104000          ASR6: HLT
2910
2911 014664 000277          SCC
2912 014666 005467 177714          NEG  UWM6          ;(R0)=000001, CC=0001
2913 014672 103003          BCC  NEG6
2914 014674 102402          BVS  NEG6
2915 014676 001401          BEQ  NEG6
2916 014700 100001          BPL  .+4
2917 014702 104000          NEG6: HLT
2918
2919 014704 000277          SCC
2920 014706 006067 177674          ROR  UWM6          ;(R0)=100000, CC=1001 >
2921 014712 103003          BCC  ROR6
2922 014714 102402          BVS  ROR6
2923 014716 001401          BEQ  ROR6
2924 014720 100401          BMI  .+4
2925 014722 104000          ROR6: HLT
2926
2927 014724 005667 177656          SBC  UWM6          ;(R0)=077777, CC=0010
2928 014730 103402          BCS  SBC6
2929 014732 102001          BVC  SBC6
2930 014734 100001          BPL  .+4
2931 014736 104000          SBC6: HLT
2932
```

```
2933 014740 000242          CLV
2934 014742 005267 177640    INC      UWM6          ;(R0)=100000, CC=1011
2935 014746 103403          BCS     INC6
2936 014750 102002          BVC     INC6
2937 014752 001401          BEQ     INC6
2938 014754 100401          BMI     .+4
2939 014756 104000          INC6:   HLT
2940
2941 014760 006267 177622    ASR     UWM6          ;(R0)=140000, CC=1010
2942 014764 000261          SEC
2943 014766 006367 177614    ASL     UWM6          ;(R0)=100000, CC=1001
2944 014772 1030C2          BCC     ASL6
2945 014774 102401          BVS     ASL6
2946 014776 100401          BMI     .+4
2947 015000 104000          ASL6:   HLT
2948
2949 015002 005367 177600    DEC     UWM6          ;(R0)=077777, CC=0011
2950 015006 103002          BCC     DEC6
2951 015010 102001          BVC     DEC6
2952 015012 100001          BPL     .+4
2953 015014 104000          DEC6:   HLT
2954
2955 015016 005567 177564    ADC     UWM6          ;(R0)=100000, CC=1010
2956 015022 103402          BCS     ADC6
2957 015024 102001          BVC     ADC6
2958 015026 100401          BMI     .+4
2959 015030 104000          ADC6:   HLT
2960 015032 000242          CLV
2961 015034 000367 177546    SWAB   UWM6
2962 015040 100401          BMI     .+4
2963 015042 104000          HLT
2964 015044 022710 000200    CMP     #200,(R0)
2965 015050 001401          BEQ     .+4
2966 015052 104000          HLT
```

```
:::*****
::: *TEST 14      CHECK UNIARY BYTE OPS (EVEN/ODD) USING ADDRESS MODE 6 (PC)
:::*****
```

```
2967
2968
2969
2970 015054 000004          TST14:  SCOPE
2971 015056 112737 000014 001202  MOV     #14,@#STSTNM      ;LOAD TEST NUMBER
2972 015064 013737 001202 177570  MOV     @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
2973 015072 012700 015434          MOV     #UBM6,R0
2974 015076 063700 001534          ADD     @#FACTOR,R0      ;R0 POINTS TO ADDRESS OF DATA
2975 015102 005067 000326          CLR     JBM6             ;CLEAR DATA
2976 015106 000277          SCC
2977 015110 000244          CLZ
2978 015112 105767 000316    TSTB   UBM6
2979 015116 103403          BCS     TSTB6
2980 015120 102402          BVS     TSTB6
2981 015122 001001          BNE     TSTB6
2982 015124 100001          BPL     .+4
2983 015126 104000          TSTB6: HLT
2984
2985 015130 000257          CCC
2986 015132 105767 000277    TSTB   UBM6+1          ;TEST ODD BYTE
2987 015136 001401          BEQ     .+4
2988 015140 104000          HLT
```



```

3045 015322 106267 000106      ASRB      UBM6          ;(R0)=100777, CC=1001
3046 015326 103002      BCC      ASRB6
3047 015330 102401      BVS      ASRB6
3048 015332 100401      BMI      .+4
3049 015334 104000      ASRB6:   HLT
3050
3051 015336 105267 000072      INCB      UBM6          ;(R0)=100400, CC=0101
3052 015342 103002      BCC      INCB6A
3053 015344 102401      BVS      INCB6A
3054 015346 001401      BEQ      .+4
3055 015350 104000      INCB6A:  HLT
3056
3057 015352 105367 000057      DECB      UBM6+1      ;(R0)=100000, CC=1001
3058 015356 103003      BCC      DECB6A
3059 015360 102402      BVS      DECB6A
3060 015362 001401      BEQ      DECB6A
3061 015364 100401      BMI      .+4
3062 015366 104000      DECB6A:  HLT
3063
3064 015370 000367 000040      SWAB      UBM6          ;(R0)=000200, CC=1000
3065 015374 103401      BCS      SWAB6
3066 015376 100401      BMI      .+4
3067 015400 104000      SWAB6:   HLT
3068
3069 015402 106167 000026      ROLB      UBM6          ;(R0)=000000, CC=0111
3070 015406 103002      BCC      ROLB6A
3071 015410 102001      BVC      ROLB6A
3072 015412 001401      BEQ      .+4
3073 015414 104000      ROLB6A:  HLT
3074
3075 015416 005767 000012      TST      UBM6          ;(R0)=000000, CC=0100
3076 015422 103402      BCS      TEST6
3077 015424 102401      BVS      TEST6
3078 015426 001401      BEQ      .+4
3079 015430 104000      TEST6:   HLT
3080
3081 015432 000401      BR      .+4          ;RESERVE A WORD
3082 015434 000000      UBM6:   .WORD      0          ;WORD RESERVED FOR DATA
3083 015436 000004      RELE1:  SCOPE
3084 015440 010702      MOV      PC,R2
3085 015442 062702 000012      ADD      #12,R2
3086 015446 012707 043764      MOV      #RELOC,PC      ;GO RELOCATE PROGRAM CODE
3087 015452 000000      REL11:  .WORD      0
3088      ;11111111111111 LAST ADDRESS OF CODE TO BE RELOCATED 111111111111
3089
3090      ;*****
3091      ;*TEST 15 CHECK UNIARY WORD OPS USING ADDRESS MODE 7
3092      ;*****
3093 015454 012767 000001 163644      MOV      #1,$TIMES      ;;DO 1 ITERATION
3094 015462 000004      TST15:  SCOPE
3095 015464 112737 000015 001202      MOV      #15,@$STSTM      ;LOAD TEST NUMBER
3096 015472 013737 001202 177570      MOV      @$STSTM,@$DISPLAY ;:DISPLAY TEST NUMBER
3097
3098      .SBTTL START OF SECTION 2
3099      ;22222222222222 FIRST ADDRESS TO BE RELOCATED 2222222222
3100 015500 010700      REL2:   MOV      PC,R0      ;GET PC
    
```

```

3101 015502 005740          TST      -(R0)          ;R0 CONTAINS THE ADDRESS OF REL2
3102 015504 010037 001540    MOV      R0,@#FRSTAD   ;SAVE
3103 015510 010700          MOV      PC,R0         ;GET CURRENT PC
3104 015512 162700 015512    SUB      #.,R0         ;SUBTRACT RELOCATION FACTOR
3105 015516 010037 001534    MOV      R0,@#FACTOR   ;SAVE RELOCATION FACTOR
3106 015522 010737 001212    MOV      PC,@#SLPERR   ;SET LOOP ADDRESS
3107 015526 062737 000030 001212  ADD      #30,@#SLPERR  ;ADJUST
3108 015534 013737 001212 001210  MOV      @#SLPERR,@#SLPADR
3109 015542 105737 001530    TSTB    @#NEXEC       ;BR IF TEST CODE TO BE EXECUTED
3110 015546 001402          BEQ     .+6
3111 015550 000167 004170    JMP     RELE2
3112 015554 000403          BR      UW7           ;RESERVE 3 WORDS FOR ADDRESSES & DATA
3113 015556 000000          .WORD  0             ;CONTAINS ADDRESS OF UW7
3114 015560 000000          .WORD  0             ;CONTAINS DATA
3115 015562 000000          .WORD  0             ;CONTAINS ADDRESS OF UW7
3116
3117 015564 010700          UW7:    MOV      PC,R0
3118 015566 005740          TST      -(R0)
3119 015570 005740          TST      -(R0)
3120 015572 005040          CLR      -(R0)       ;CLEAR TEST DATA
3121 015574 010002          MOV      R0,R2
3122 015576 010240          MOV      R2,-(R0)    ;SET UP ADDRESS
3123 015600 005720          TST      (R0)+       ;MOVE R0 TO NEXT ADDRESS
3124 015602 005720          TST      (R0)+
3125 015604 010210          MOV      R2,(R0)    ;SET NEXT ADDRESS
3126 015606 010200          MOV      R2,R0      ;SET R0 POINTING TO DATA
3127 015610 000277          SCC
3128 015612 000244          CLZ
3129 015614 005772 000002    TST      @2(2)       ;(R0)=000000, CC=0100
3130 015620 001401          BEQ     .+4
3131 015622 104000          HLT
3132
3133 015624 000277          SCC
3134 015626 005672 177776    SBC      @-2(2)     ;(R0)-177777, CC=1001
3135 015632 103002          SBC7
3136 015634 102401          BVS     SBC7
3137 015636 100401          BMI     .+4
3138 015640 104000          SBC7:    HLT
3139
3140 015642 000277          SCC
3141 015644 000241          CLC
3142 015646 006372 000002    ASL      @2(2)       ;(R0)-177776, CC=1001
3143 015652 103002          BCC     ASL7
3144 015654 102401          BVS     ASL7
3145 015656 100401          BMI     .+4
3146 015660 104000          ASL7:    HLT
3147
3148 015662 000257          CCC
3149 015664 005372 000002    DEC      @2(2)       ;(R0)-177775, CC=1000
3150 015670 103402          BCS     DEC7
3151 015672 102401          BVS     DEC7
3152 015674 100401          BMI     .+4
3153 015676 104000          DEC7:    HLT
3154
3155 015700 000262          SEV
3156 015702 006272 177776    ASR      @-2(2)     ;(R0)=177776, CC=1001

```

3157	015706	103002		BCC	ASR7		
3158	015710	102401		BVS	ASR7		
3159	015712	100401		BMI	.+4		
3160	015714	104000		ASR7:	HLT		
3161							
3162	015716	000241		CLC			
3163	015720	000262		SEV			
3164	015722	006072	177776	ROR	@-2(2)	;(R0)=077777, CC=0000	
3165	015726	101402		BLOS	ROR7	;BRANCH IF C OR Z IS SET	
3166	015730	102401		BVS	ROR7		
3167	015732	100001		BPL	.+4		
3168	015734	104000		ROR7:	HLT		
3169							
3170	015736	000262		SEV			
3171	015740	005472	000002	NEG	@2(2)	;(R0)=100001, CC=1001	
3172	015744	103002		BCC	NEG7		
3173	015746	102401		BVS	NEG7		
3174	015750	100401		BMI	.+4		
3175	015752	104000		NEG7:	HLT		
3176							
3177	015754	000250		CLN			
3178	015756	000372	177776	SWAB	@-2(2)	;(R0)=000600, CC=1000	
3179	015762	103401		BCS	SWAB7		
3180	015764	100401		BMI	.+4		
3181	015766	104000		SWAB7:	HLT		
3182							
3183	015770	000262		SEV			
3184	015772	005172	000002	COM	@2(2)	;(R0)=177177, CC=1001	
3185	015776	103002		BCC	COM7		
3186	016000	102401		BVS	COM7		
3187	016002	100401		BMI	.+4		
3188	016004	104000		COM7:	HLT		
3189							
3190	016006	000372	000002	SWAB	@2(2)	;(R0)=077776, CC=1000	
3191	016012	100401		BMI	.+4		
3192	016014	104000		HLT			
3193							
3194	016016	000277		SCC			
3195	016020	005572	177776	ADC	@-2(2)	;(R0)=077777, CC=0000	
3196	016024	103402		BCS	ADC7		
3197	016026	102401		BVS	ADC7		
3198	016030	100001		BPL	.+4		
3199	016032	104000		ADC7:	HLT		
3200							
3201	016034	005272	000002	INC	@2(2)	;(R0)=100000, CC=1010	
3202	016040	102001		BVC	INC7		
3203	016042	100401		BMI	.+4		
3204	016044	104000		INC7:	HLT		
3205							
3206	016046	000257		CCC			
3207	016050	006172	177776	ROL	@-2(2)	;(R0)=000000, CC=0111	
3208	016054	103002		BCC	ROL7		
3209	016056	102001		BVC	ROL7		
3210	016060	001401		BEQ	.+4		
3211	016062	104000		ROL7:	HLT		
3212				::	*****		

```

3213 : *TEST 16 CHECK UNIARY BYTE OPS USING ADDRESS MODE 7
3214 : *****
3215 016064 000004 TST16: SCOPE
3216 016066 112737 000016 001202 MOVB #16,@RSTSTNM ;LOAD TEST NUMBER
3217 016074 013737 001202 177570 MOV @RSTSTNM,@RDISPLAY ;:DISPLAY TEST NUMBER
3218 016102 012700 015560 MOV #UWM7,R0
3219 016106 063700 001534 ADD @WFACTOR,R0
3220 016112 010002 MOV R0,R2
3221 016114 010067 177442 MOV R0,UWM7+2
3222 016120 005720 TST (R0)+
3223 016122 005210 INC (R0) ;WORD FOLLOWING UWM7 CONTAINS ADDRESS
3224 016124 005740 TST -(R0) ;OF ODD BYTE, R0 POINTS TO DATA WORD
3225 016126 005010 CLR (R0) ;PRESET DATA
3226 016130 010067 177422 MOV R0,UWM7-2
3227 :NOTE: @2(2) REFERENCES THE ODD BYTE, AND @-2(2) REFERENCES THE EVEN BYTE.
3228
3229 016134 000263 +SEC!SEV ;SET C AND V
3230 016136 105672 000002 SBCB @2(2) ;(R0)=177400, CC=1001
3231 016142 103003 BCC SBCB7
3232 016144 102402 BVS SBCB7
3233 016146 001401 BEQ SBCB7
3234 016150 100401 BMI .+4
3235 016152 104000 SBCB7: HLT
3236
3237 016154 000277 SCC ;SET CONDITION CODES
3238 016156 105572 177776 ADCB @-2(2) ;(R0)=177401, CC=0000
3239 016162 103403 BCS ADCB7
3240 016164 102402 BVS ADCB7
3241 016166 001401 BEQ ADCB7
3242 016170 100001 BPL .+4
3243 016172 104000 ADCB7: HLT
3244
3245 016174 105172 177776 COMB @-2(2) ;(R0)=177776, CC=1001
3246 016200 103002 BCC COMB7
3247 016202 102401 BVS COMB7
3248 016204 100401 BMI .+4
3249 016206 104000 COMB7: HLT
3250
3251 016210 000241 CLC ;CLEAR CARRY
3252 016212 106072 000002 RORB @2(2) ;(R0)-077776, CC=0011
3253 016216 103002 BCC RORB7
3254 016220 102001 BVC RORB7
3255 016222 100001 BPL .+4
3256 016224 104000 RORB7: HLT
3257
3258 016226 105272 000002 INCB @2(2) ;(R0)=100376, CC=1011
3259 016232 103002 BCC INCB7
3260 016234 102001 BVC INCB7
3261 016236 100401 BMI .+4
3262 016240 104000 INCB7: HLT
3263
3264 016242 105372 177776 DECB @-2(2) ;(R0)-100375, CC=1001
3265 016246 103002 BCC DECB7
3266 016250 102401 BVS DECB7
3267 016252 100401 BMI .+4
3268 016254 104000 DECB7: HLT
  
```

```
3269
3270 016256 106372 000002      ASLB      @2(2)          ;(R0)-000375, CC=0111
3271 016262 103002          BCC      ASLB7
3272 016264 102001          BVC      ASLB7
3273 016266 001401          BEQ      .+4
3274 016270 104000      ASLB7:   HLT
3275
3276 016272 000241          CLC
3277 016274 106272 177776      ASRB      @-2(2)        ;CLEAR CARRY
3278 016300 103002          BCC      ASRB7        ;(R0)=000376, CC=1001
3279 016302 102401          BVS      ASRB7
3280 016304 100401          BMI
3281 016306 104000      ASRB7:   HLT
3282
3283 016310 105472 000002      NEGB     @2(2)          ;(R0)=000376, CC=0100
3284 016314 103402          BCS      NEGB7
3285 016316 102401          BVS      NEGB7
3286 016320 001401          BEQ      .+4
3287 016322 104000      NEGB7:   HLT
3288
3289 016324 000262          SEV
3290 016326 106172 177776      ROLB     @-2(2)        ;(R0)=00374, CC=1001
3291 016332 103002          BCC      ROLB7
3292 016334 102401          BVS      ROLB7
3293 016336 100401          BMI
3294 016340 104000      ROLB7:   HLT
3295
3296 016342 105272 177776      INCB     @-2(2)        ;(R0)=000375, CC=1001
3297 016346 105272 177776      INCB     @-2(2)        ;(R0)=000376, CC=1001
3298 016352 105572 177776      ADCB     @-2(2)        ;(R0)=000377, CC=1000
3299 016356 105172 177776      COMB     @-2(2)        ;(R0)=000000, CC=0100
3300 016362 001401          BEQ      .+4
3301 016364 104000          HLT
3302
3303      ;*****
3304      ;*TEST 17      CHECK BINARY OPS USING ADDRESS MODE 0
3305      ;*****
3306      TST17:   SCOPE
3307      MOV      #17,@$TSTNM          ;LOAD TEST NUMBER
3308      MOV      @$TSTNM,@$DISPLAY    ;DISPLAY TEST NUMBER
3309      SCC
3310      MOV      PC,R0                ;SET CONDITION CODES
3311      BCC      MOV0                  ;R0=PC, CC=X001
3312      BVS      MOV0
3313      BNE      .+4
3314      MOV0:   HLT
3315
3316      MOV      R0,R2                ;R2=R0
3317      SEV
3318      SUB     R0,R2                ;SET V
3319      BCS      SUB0                  ;R2=000000, CC=0100
3320      BVS      SUB0
3321      BEQ      .+4
3322      SUB0:   HLT
3323
3324      CLZ
3325      MOV      R2,R3                ;R2=R3=000000, CC=0100
```

```

3325 016442 103401          BCS      MOV0A
3326 016444 001401          BEQ      .+4
3327 016446 104000          MOV0A:  HLT
3328
3329 016450 000257          CCC
3330 016452 000272          +SEV:SEN          ;SET V & N
3331 016454 020203          CMP      R2,R3      ;R2=R3=000000, CC=0100
3332 016456 103403          BCS      CMO
3333 016460 102402          BVS      CMO
3334 016462 001001          BNE      CMO
3335 016464 100001          BPL      .+4
3336 016466 104000          CMO:    HLT
3337
3338 016470 010002          MOV      R0,R2      ;R0=R2
3339 016472 010203          MOV      R2,R3      ;R0=R2=R3
3340 016474 060203          ADD      R2,R3      ;R3=2*R0
3341 016476 006302          ASL      R2          ;R2=2*R0
3342 016500 020203          CMP      R2,R3      ;R2=R3=2*R0
3343 016502 001401          BFQ      .+4
3344 016504 104000          HLT          ;ERROR! CHECK ADD INSTRUCTION
3345
3346          ;THE FOLLOWING SUBTEST SHIFTS A BIT THROUGH R2 AND R5 AND DOES A
3347          ;BIT TEST (BIT) USING R2 AND R5.
3348 016506 005002          CLR      R2
3349 016510 005202          INC      R2          ;R2=1
3350 016512 000402          BR      2$
3351 016514 006302          1$:    ASL      R2
3352 016516 100407          BMI      4$
3353 016520 010205          2$:    MOV      R2,R5
3354 016522 000277          SCC          ;SET CC'S
3355 016524 030205          BIT      R2,R5      ;R2=R5, CC=X001
3356 016526 103002          BCC      3$
3357 016530 102401          BVS      3$
3358 016532 001370          BNE      1$
3359 016534 104000          3$:    HLT
3360 016536 010205          4$:    MOV      R2,R5      ;R2 AND R5=100000(OCTAL)
3361 016540 000257          CCC          ;CLEAR CC'S
3362 016542 030205          BIT      R2,R5      ;R2=R5, CC=1000
3363 016544 100401          BMI      .+4
3364 016546 104000          HLT
3365
3366 016550 005002          CLR      R2
3367 016552 000277          SCC          ;SET CC'S
3368 016554 050002          BIS      R0,R2      ;R0=PC (NON-ZERIO DATA), CC=X001
3369 016556 103002          BCC      BISO
3370 016560 102401          BVS      BISO
3371 016562 001001          BNE      .+4
3372 016564 104000          BISO:  HLT
3373
3374 016566 010003          MOV      R0,R3
3375 016570 000277          SCC          ;CC=1111
3376 016572 000244          CLZ
3377 016574 040003          BIC      R0,R3      ;R0=R3, CC=0101
3378 016576 103003          BCC      BICO
3379 016600 102402          BVS      BICO
3380 016602 001001          BNE      BICO
  
```

```
3381 016604 100001
3382 016606 104000
3383
3384 016610 010004
3385 016612 005104
3386 016614 040004
3387 016616 005104
3388 016620 020004
3389 016622 001401
3390 016624 104000
3391
3392 016626 010004
3393 016630 005104
3394 016632 010403
3395 016634 050003
3396 016636 103001
3397 016640 100401
3398 016642 104000
3399 016644 005203
3400 016646 001401
3401 016650 104000
3402 016652 010304
3403 016654 005103
3404 016656 000261
3405 016660 006004
3406 016662 060304
3407 016664 103003
3408 016666 102002
3409 016670 001401
3410 016672 100001
3411 016674 104000
3412 016676 010700
3413 016700 022020
3414 016702 020007
3415 016704 001401
3416 016706 104000
3417
3418 016710 010700
3419 016712 062700 000010
3420 016716 010002
3421 016720 020700
3422 016722 001002
3423 016724 020200
3424 016726 001401
3425 016730 104000
3426
3427
3428
3429 016732 000004
3430 016734 112737 000020 001202
3431 016742 013737 001202 177570
3432 016750 000402
3433 016752 000000
3434 016754 000000
3435 016756 010704
3436 016760 005744

BICO: BPL .+4
      HLT
      MOV R0,R4
      COM R4
      BIC R0,R4 ;R0-COMPLEMENT OF R4, R4 REMAINS UNCHANGED
      COM R4
      CMP R0,R4 ;R0=R4
      BEQ .+4
      HLT

BISOA: MOV R0,R4
        COM R4
        MOV R4,R3 ;R3=R4
        BIS R0,R3 ;R3=COMPLEMENT OF R0, CC=1001
        BCC BISOA
        BMI .+4
        HLT
        INC R3 ;R3=0 AFTER INC
        BEQ .+4
        HLT
        MOV R3,R4 ;R3=R4=0
        COM R3 ;R3=177777
        SEC ;SET C
        ROR R4 ;R4=100000
        ADD R3,R4 ;R3=177777,R4=077777, CC=0011
        BCC ADD0
        BVC ADD0
        BEQ ADD0
        BPL .+4
        HLT
        MOV PC,R0 ;R0=PC
        CMP (R0)+,(R0)+ ;R0=R0+4
        CMP R0,PC ;PC=PC+4=R0
        BEQ .+4
        HLT

ADDO: HLT
        MOV PC,R0 ;R0=PC
        ADD #10,R0 ;R0=PC+10(8)
        MOV R0,R2 ;R2=R0
        CMP PC,R0 ;R0=PC
        BNE CMPOA
        CMP R2,R0 ;R2=R0
        BEQ .+4
        HLT

CMPOA: HLT

*****
:*TEST 20 CHECK BINARY OPS USING ADDRESS MODE 1
*****
TST20: SCOPE
        MOV #20,@$TSTNM ;LOAD TEST NUMBER
        MOV @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
        BR .+6 ;RESERVE TWO WORDS
        .WORD 0 ;RESERVED FOR SOURCE DATA
        .WORD 0 ;RESERVED FOR DESTINATION DATA
        MOV PC,R4
        TST -(R4)
```


3437	016762	005044	CLR	-(R4)	;R4 POINTS TO DESTINATION DATA
3438	016764	010403	MOV	R4,R3	
3439	016766	005043	CLR	-(R3)	;R3 POINTS TO SOURCE DATA
3440					
3441	016770	005113	COM	(R3)	;(R3)=177777
3442	016772	005214	INC	(R4)	;(R4)=000001
3443	016774	000262	SEV		;SET V
3444	016776	061314	ADD	(R3),(R4)	;(R3)=177777,(R4)=000000, CC=0101
3445	017000	103002	BCC	ADD1	
3446	017002	102401	BVS	ADD1	
3447	017004	001401	BEQ	+.4	
3448	017006	104000	ADD1:	HLT	
3449					
3450	017010	000277	SCC		
3451	017012	000250	CLN		
3452	017014	021314	CMP	(R3),(R4)	;(R3)=177777,(R4)=000000, CC=1000
3453	017016	103403	BCS	CMP1	
3454	017020	102402	BVS	CMP1	
3455	017022	001401	BEQ	CMP1	
3456	017024	100401	BMI	+.4	
3457	017026	104000	CMP1:	HLT	
3458					
3459	017030	000277	SCC		
3460	017032	000244	CLZ		
3461	017034	031314	BIT	(R3),(R4)	;(R3)=177777,(R4)=000000, CC=0101
3462	017036	103002	BCC	BITT1	
3463	017040	102401	BVS	BITT1	
3464	017042	001401	BEQ	+.4	
3465	017044	104000	BITT1:	HLT	
3466					
3467	017046	000277	SCC		
3468	017050	000245	+CLC!CLZ		
3469	017052	005114	COM	(R4)	;(R4)=177777
3470	017054	161314	SUB	(R3),(R4)	;(R3)=177777,(R4)=000000, CC=0100
3471	017056	103402	BCS	SUB1	
3472	017060	102401	BVS	SUB1	
3473	017062	001401	BEQ	+.4	
3474	017064	104000	SUB1:	HLT	
3475					
3476	017066	105013	CLRB	(R3)	;(R3)=177400
3477	017070	000313	SWAB	(R3)	;(R3)=000377
3478	017072	000270	SEN		
3479	017074	011314	MOV	(R3),(R4)	;(R3)=(R4)=000377
3480	017076	100001	BPL	+.4	
3481	017100	104000	HLT		
3482	017102	000314	SWAB	(R4)	;(R3)=000377,(R4)=177400
3483	017104	000263	+SEC!SEV		;SET C & V
3484	017106	051314	BIS	(R3),(R4)	;(R3)=000377,(R4)=177777, CC=1001
3485	017110	103002	BCC	BIS1	
3486	017112	102401	BVS	BIS1	
3487	017114	100401	BMI	+.4	
3488	017116	104000	BIS1:	HLT	
3489					
3490	017120	041314	BIC	(R3),(R4)	;(R3)=000377,(R4)=177400, CC=1001
3491	017122	103002	BCC	BIC1	
3492	017124	102401	BVS	BIC1	

```
3493 017126 100401
3494 017130 104000
3495
3496 017132 000262
3497 017134 021314
3498 017136 103003
3499 017140 102402
3500 017142 001401
3501 017144 100001
3502 017146 104000
3503
3504 017150 005013
3505 017152 000261
3506 017154 006013
3507 017156 011314
3508 017160 005114
3509 017162 161314
3510 017164 103002
3511 017166 102001
3512 017170 100401
3513 017172 104000
3514
3515 017174 000277
3516 017176 161314
3517 017200 101402
3518 017202 102401
3519 017204 100001
3520 017206 104000
3521
3522 017210 011314
3523 017212 001401
3524 017214 100401
3525 017216 104000
3526
3527 017220 061314
3528 017222 103003
3529 017224 102002
3530 017226 001001
3531 017230 100001
3532 017232 104000
3533
3534 017234 005113
3535 017236 011314
3536 017240 061314
3537 017242 103402
3538 017244 102001
3539 017246 100401
3540 017250 104000
3541
3542 017252 062714 000002
3543 017256 005714
3544 017260 001401
3545 017262 104000
3546
3547
3548
```

BIC1: BMI .+4
HLT

SEV ;SET V
CMP (R3),(R4) ;(R3)=000377,(R4)=177400,CC=0001
BCC CMP1A
BVS CMP1A
BEQ CMP1A
BPL .+4
CMP1A: HLT

CLR (R3) ;(R3)=000000
SEC
ROR (R3) ;(R3)=100000
MOV (R3),(R4) ;(R3)=(R4)=100000
COM (R4) ;(R4)=077777
SUB (R3),(R4) ;(R3)=100000,(R4)=177777,CC=1011
BCC SUB1A
BVC SUB1A
BMI .+4
SUB1A: HLT

SCC
SUB (R3),(R4) ;(R3)=100000,(R4)=077777,CC=0000
BLOS SUB1B ;BRANCH IF C OR Z IS SET
BVS SUB1B
BPL .+4
SUB1B: HLT

MOV (R3),(R4) ;(R3)=100000,(R4)=100000,CC=1000
BEQ MOV1
BMI .+4
MOV1: HLT

ADD (R3),(R4) ;(R3)=100000,(R4)=000000,CC=0111
BCC ADD1A
BVC ADD1A
BNE ADD1A
BPL .+4
ADD1A: HLT

COM (R3) ;(R3)=077777
MOV (R3),(R4) ;(R4)=077777
ADD (R3),(R4) ;(R3)=077777,(R4)=177776,CC=1010
BCS ADD1B
BVC ADD1B
BMI .+4
ADD1B: HLT

ADD #2,(R4)
TST (R4) ;CHECK FINAL RESULT
BEQ .+4
HLT

*TEST 21 CHECK BINARY BYTE OPS USING ADDRESS MODE 1

```
3549 017264 000004  
3550 017266 112737 000021 001202  
3551 017274 013737 001202 177570  
3552 017302 000402  
3553 017304 000000  
3554 017306 000000  
3555 017310 010705  
3556 017312 005745  
3557 017314 005045  
3558 017316 010502  
3559 017320 005042  
3560 017322 005202  
3561 017324 105112  
3562  
3563 017326 000277  
3564 017330 111215  
3565 017332 103005  
3566 017334 102404  
3567 017336 001403  
3568 017340 100002  
3569 017342 105215  
3570 017344 001401  
3571 017346 104000  
3572  
3573 017350 106312  
3574 017352 102376  
3575 017354 106012  
3576 017356 105315  
3577 017360 106015  
3578 017362 000257  
3579 017364 121512  
3580 017366 102001  
3581 017370 100401  
3582 017372 104000  
3583  
3584 017374 005003  
3585 017376 000261  
3586 017400 006003  
3587 017402 050315  
3588 017404 000273  
3589 017406 131215  
3590 017410 103002  
3591 017412 102401  
3592 017414 001401  
3593 017416 104000  
3594  
3595 017420 151215  
3596 017422 103001  
3597 017424 100401  
3598 017426 104000  
3599  
3600 017430 141215  
3601 017432 103002  
3602 017434 001401  
3603 017436 100001  
3604 017440 104000
```

TST21: SCOPE
MOVB #21,@\$TSTNM ;LOAD TEST NUMBER
MOV @\$TSTNM,@\$DISPLAY ;:DISPLAY TEST NUMBER
BR .+6
.WORD 0
.WORD 0
MOV PC,R5
TST -(R5)
CLR -(R5) ;(R5)=000000
MOV R5,R2
CLR -(R2) ;(R2)=000000
INC R2 ;R2 POINTS TO ODD BYTE
COMB (R2) ;(R2)=177400

SCC
MOVB (R2),(R5) ;(R2)=177400,(R5)=000377,CC=1001
BCC MOVB1
BVS MOVB1
BEQ MOVB1
BPL MOVB1
INCB (R5) ;CHECK RESULT
BEQ .+4
MOVB1: HLT

ASLB (R2) ;SHIFT (R2) UNTIL
BVC .-2 ;(R2)=000000
RORB (R2) ;(R2)=100000
DECB (R5) ;(R5)=00377
RORB (R5) ;(R5)=000177
CCC
CMPB (R5),(R2) ;(R5)=000177,(R2)=100000,CC=1010
BVC CMPB1
BMI .+4
CMPB1: HLT

CLR R3
SEC
ROR R3 ;R3=100000
BIS R3,(R5) ;(R5)=100177
+SEC!SEV.SEN ;SET C,V,&N
BITB (R2),(R5) ;(R2)=100000,(R5)=100177,CC=0101
BCC BITB1
BVS BITB1
BEQ .+4
BITB1: HLT

BISB (R2),(R5) ;(R2)=100000,(R5)=100377,CC=1001
BCC BISB1
BMI .+4
BISB1: HLT

BICB (R2),(R5) ;(R2)=100000,(R5)=100177,CC=0001
BCC BICB1
BEQ BICB1
BPL .+4
BICB1: HLT

3605					
3606	017442	105112	COMB	(R2)	;(R2)=077400,(R5)=100177
3607	017444	121215	CMPB	(R2),(R5)	
3608	017446	001401	BEQ	+.4	
3609	017450	104000	HLT		
3610					
3611	017452	141512	BICB	(R5),(R2)	;(R5)=100177,(R2)=000000,CC=0100
3612	017454	001002	BNE	BICB1A	
3613	017456	105712	TSTB	(R2)	
3614	017460	001401	BEQ	+.4	
3615	017462	104000	BICB1A:	HLT	
3616					
3617	017464	000402	BR	+.6	;RESERVE TWO WORDS FOR DATA
3618	017466	000000	.WORD	0	;SOURCE DATA
3619	017470	000000	.WORD	0	;DEST DATA
3620	017472	010705	MOV	PC,R5	
3621	017474	005745	TST	-(R5)	
3622	017476	105045	CLRB	-(R5)	;R5 POINTS TO DEST ODD BYTE
3623	017500	010504	MOV	R5,R4	
3624	017502	105044	CLRB	-(R4)	;R4 POINTS TO DEST EVEN BYTE
3625	017504	010403	MOV	R4,R3	
3626	017506	105043	CLRB	-(R3)	;R3 POINTS TO SOURCE ODD BYTE
3627	017510	010302	MOV	R3,R2	
3628	017512	105042	CLRB	-(R2)	;R2 POINTS TO SOURCE EVEN BYTE
3629					
3630					
3631					
3632	017514	000261	SEC		;COMMENTS ARE LEAST SIGNIFICANT 4 BITS OF BYTES POINTED TO BY R2,R3 ;R4, AND R5 RESPECTIVELY AND THE REMAINING BITS ARE 0'S.
3633					;SET CARRY
3634	017516	106112	ROLB	(R2)	;(R2),(R3),(R4),(R5)
3635	017520	111214	MOVB	(R2),(R4)	;0001,0000,0000,0000
3636	017522	106112	ROLB	(R2)	;0001,0000,0001,0000
3637	017524	111213	MOVB	(R2),(R3)	;0010,0000,0001,0000
3638	017526	106112	ROLB	(R2)	;0010,0010,0001,0000
3639	017530	111315	MOVB	(R3),(R5)	;0100,0010,0001,0010
3640	017532	106112	ROLB	(R2)	;1000,0010,0001,0010
3641	017534	106113	ROLB	(R3)	;1000,0100,0001,0010
3642	017536	151215	BISB	(R2),(R5)	;1000,0100,0001,1010
3643	017540	131512	BITB	(R5),(R2)	;1000,0100,0001,1010
3644	017542	001426	BEQ	BIN1	
3645	017544	151314	BISB	(R3),(R4)	;1000,0100,0101,1010
3646	017546	131413	BITB	(R4),(R3)	;1000,0100,0101,1010
3647	017550	001423	BEQ	BIN1	
3648	017552	105213	INCB	(R3)	;1000,0101,0101,1010
3649	017554	121314	CMPB	(R3),(R4)	;1000,0101,0101,1010
3650	017556	001020	BNE	BIN1	
3651	017560	106113	ROLB	(R3)	;1000,1010,0101,1010
3652	017562	121315	CMPB	(R3),(R5)	;1000,1010,0101,1010
3653	017564	001015	BNE	BIN1	
3654	017566	106212	ASRB	(R2)	;0100,1010,0101,1010
3655	017570	131214	BITB	(R2),(R4)	;0100,1010,0101,1010
3656	017572	001412	BEQ	BIN1	
3657	017574	106015	RORB	(R5)	;0100,1010,0101,0101
3658	017576	121415	CMPB	(R4),(R5)	;0100,1010,0101,0101
3659	017600	001007	BNE	BIN1	
3660	017602	105314	DECB	(R4)	;0100,1010,0100,0101

```
3661 017604 141214          BICB   (R2),(R4)      ;0100,1010,0000,0101
3662 017606 001004          BNE    BIN1
3663 017610 111314          MOVB   (R3),(R4)      ;0100,1010,1010,0101
3664 017612 106213          ASRB   (R3)           ;0100,0101,1010,0101
3665 017614 141315          BICB   (R3),(R5)      ;0100,0101,1010,0101
3666 017616 001401          BEQ    .+4
3667 017620 104000          BIN1:  HLT
3668
3669          ;*****
3670          ;*TEST 22 CHECK BINARY WORD OPS USING ADDRESS MODE 2 & 4
3671          ;*****
3671 017622 000004          TST22: SCOPE
3672 017624 112737 000022 001202  MOVB   #22,@#STSTNM    ;LOAD TEST NUMBER
3673 017632 013737 001202 177570  MOV    @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
3674 017640 012704 017470          MOV    #BICB1A+6,R4
3675 017644 012702 017466          MOV    #BICB1A+4,R2
3676 017650 063702 001534          ADD    @#FACTOR,R2
3677 017654 063704 001534          ADD    @#FACTOR,R4
3678 017660 010405          MOV    R4,R5           ;SET DESTINATION REGISTER
3679 017662 012715 000001          MOV    #1,(R5)
3680 017666 012712 177777          MOV    #-1,(R2)
3681 017672 000257          CCC
3682 017674 000262          SEV
3683 017676 062225          ADD    (R2)+,(R5)+    ;(R2)=177777,(R5)=000000,CC=0101
3684 017700 103002          BCC    ADD2
3685 017702 102401          BVS    ADD2
3686 017704 001401          BEQ    .+4
3687 017706 104000          ADD2:  HLT
3688
3689 017710 000262          SEV           ;SET V
3690 017712 024527 000001          CMP    -(R5),#1      ;(R5)=000000,CC=1001
3691 017716 103002          BCC    CMP2
3692 017720 102401          BVS    CMP2
3693 017722 100401          BMI    .+4
3694 017724 104000          CMP2:  HLT
3695
3696 017726 054225          BIS    -(R2),(R5)+    ;(R2)=177777,(R5)=177777,CC=1001
3697 017730 103001          BCC    BIS2
3698 017732 100401          BMI    .+4
3699 017734 104000          BIS2:  HLT
3700 017736 000277          SCC
3701 017740 000244          CLZ
3702 017742 162245          SUB    (R2)+,-(R5)    ;(R2)=177777,(R5)=000000,CC=0100
3703 017744 103402          BCS    SUB2
3704 017746 102401          BVS    SUB2
3705 017750 001401          BEQ    .+4
3706 017752 104000          SUB2:  HLT
3707
3708 017754 005442          NEG    -(R2)          ;(R2)=000001
3709 017756 005115          COM    (R5)           ;(R5)=177777
3710 017760 000277          SCC
3711 017762 000250          CLN
3712 017764 042225          BIC    (R2)+,(R5)+    ;(R2)=000001,(R5)=177776,CC=1001
3713 017766 103003          BCC    BIC2
3714 017770 102402          BVS    BIC2
3715 017772 001401          BEQ    BIC2
3716 017774 100401          BMI    .+4
```

```
3717 017776 104000 BIC2: HLT
3718
3719 020000 012742 125252 MOV #125252,-(R2)
3720 020004 012245 MOV (R2)+,-(R5)
3721 020006 005125 COM (R5)+ ;(R5)=052525
3722 020010 000262 SEV
3723 020012 034245 BIT -(R2),-(R5) ;(R2)=125252,(R5)=052525,CC=0101
3724 020014 103002 BCC BITT2
3725 020016 102401 BVS BITT2
3726 020020 001401 BEQ .+4
3727 020022 104000 BITT2: HLT
3728
3729 020024 000262 SEV
3730 020026 052225 BIS (R2)+,(R5)+ ;(R2)=125252,(R5)=177777,CC=1001
3731 020030 103002 BCC BIS2A
3732 020032 102401 BVS BIS2A
3733 020034 100401 BMI .+4
3734 020036 104000 BIS2A: HLT
3735
3736 020040 042745 125252 BIC #125252,-(R5) ;(R5)=052525
3737 020044 005125 COM (R5)+ ;(R5)=125252
3738 020046 024245 CMP -(R2),-(R5)
3739 020050 001401 BEQ .+4
3740 020052 104000 HLT
3741
3742 020054 005012 CLR (R2)
3743 020056 005122 COM (R2)+ ;(R2)=177777
3744 020060 162742 000001 SUB #1,-(R2) ;(R2)=177776,CC=1000
3745 020064 103402 BCS SUB2A
3746 020066 102401 BVS SUB2A
3747 020070 100401 BMI .+4
3748 020072 104000 SUB2A: HLT
3749 020074 010702 MOV PC,R2 ;GET CURRENT PC
3750 020076 010205 MOV R2,R5 ;MOVE TO R5
3751 020100 124245 1$: CMPB -(R2),-(R5) ;COMPARE ALL PREVIOUS MEMORY ADDRESSES
3752 020102 001401 BEQ .+4
3753 020104 104000 HLT ;ERROR!
3754 020106 020237 001540 CMP R2,@#FRSTAD ;CHECK FOR LOW LIMIT
3755 020112 001372 BNE 1$
3756
3757 ;*****
3758 ;*TEST 23 CHECK BINARY BYTE OPS USING ADDRESS MODE 2 & 4
3759 ;*****
3759 020114 000004 TST23: SCOPE
3760 020116 112737 000023 001202 MOVB #23,@#STSTNM ;LOAD TEST NUMBER
3761 020124 013737 001202 177570 MOV @#STSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER
3762 020132 000402 BR .+6 ;RESERVE TWO WORDS
3763 020134 000000 .WORD 0 ;SOURCE DATA
3764 020136 000000 .WORD 0 ;DESTINATION DATA
3765 020140 010703 MOV PC,R3
3766 020142 005743 TST -(R3)
3767
3768 ;FIRST CHECK AUTO INCREMENT/DECREMENT
3769 020144 010300 MOV R3,R0 ;R0=ADDRESS OF MOV ABOVE
3770 020146 010002 MOV R0,R2 ;R2 R0
3771 020150 005302 DEC R2 ;R2-R0-1
3772 020152 010604 MOV SP,R4
```

3773	020154	010605	MOV	SP,R5	
3774	020156	005745	TST	-(R5)	;R5=SP-2
3775					
3776	020160	114046	MOVB	-(R0),-(SP)	;R0=R0-1, SP=SP-2
3777	020162	020506	CMP	R5,SP	;R5=SP
3778	020164	001021	BNE	BINB	
3779	020166	020200	CMP	R2,R0	;R2=R0
3780	020170	001017	BNE	BINB	
3781	020172	122026	CMPB	(R0)+,(SP)+	;R0=R0+1, SP=SP+2
3782	020174	020406	CMP	R4,SP	;R4=SP (SP BACK TO ORIGINAL)
3783	020176	001014	BNE	BINB	

```

3784 020200 020003      CMP      R0,R3      ;R0=R3 (R0 BACK TO ORIGINAL)
3785 020202 001012      BNE     BINB
3786 020204 154640      BISB   -(SP),-(R0)  ;SP=SP-2, R0-R0-1
3787 020206 020506      CMP     R5,SP      ;R5=SP
3788 020210 001007      BNE     BINB
3789 020212 020200      CMP     R2,R0      ;R2=R0
3790 020214 001005      BNE     BINB
3791 020216 142620      BICB   (SP)+,(R0)+ ;SP=SP+2,R0=R0+1 (SP BACK TO ORIGINAL)
3792 020220 020406      CMP     R4,SP      ;R4=SP
3793 020222 001002      BNE     BINB
3794 020224 020003      CMP     R0,R3      ;R0-R3
3795 020226 001401      BEQ    .+4
3796 020230 104000      HLT
3797 020232 010003      MOV     R0,R3      ;R0=R3
3798 020234 112743 000200      MOVB   #200,-(R3)  ;R3=ODD BYTE (UPPER BYTE)
3799 020240 112743 000377      MOVB   #377,-(R3)  ;(R3)=100377, R3=EVEN BYTE (LOWER BYTE)
3800 020244 010304      MOV     R3,R4
3801 020246 112744 000177      MOVB   #177,-(R4)  ;R4= ODD BYTE (UPPER BYTE)
3802 020252 112744 000000      MOVB   #0,-(R4)   ;(R4)=077400, R4=EVEN BYTE (LOWER BYTE)
3803 020256 001401      BEQ    .+4
3804 020260 104000      HLT
3805
3806 020262 152324      BISB   (R3)+,(R4)+ ;(R3)=100377,(R4)=077777
3807 020264 100401      BMI    .+4
3808 020266 104000      HLT
3809
3810 020270 122324      CMPB   (R3)+,(R4)+ ;CC=0X10
3811 020272 103402      BCS    CMPB2
3812 020274 102001      BVC    CMPB2
3813 020276 100001      BPL    .+4
3814 020300 104000      HLT
3815
3816 020302 000261      SEC
3817 020304 134344      BITB   -(R3),-(R4) ;SET C BIT, CC=0X11
3818 020306 103002      BCC    BITB2      ;CC=X101
3819 020310 102401      BVS    BITB2
3820 020312 001401      BEQ    .+4
3821 020314 104000      HLT
3822
3823 020316 000244      CLZ
3824 020320 144344      BICB   -(R3),-(R4) ;(R3)=100377,(R4)=077400
3825 020322 001401      BEQ    .+4
3826 020324 104000      HLT
3827
3828
3829
3830 020326 000004      TST24: SCOPE
3831 020330 112737 000024 001202      MOVB   #24,@$TSTNM ;LOAD TEST NUMBER
3832 020336 013737 001202 177570      MOV    @$TSTNM,@$DISPLAY ;DISPLAY TEST NUMBER
3833 020344 000404      BR     2$          ;RESERVE SPACE FOR DATA AND ADDRESSES
3834 020346 000000      .WORD 0           ;CONTAINS ADDRESS OF SOURCE DATA
3835 020350 000000      .WORD 0           ;CONTAINS ADDRESS OF DEST DATA
3836 020352 000000      .WORD 0           ;CONTAINS SOURCE DATA
3837 020354 000000      .WORD 0           ;CONTAINS DEST DATA
3838 020356 010701      2$:  MOV     PC,R1
3839 020360 010100      MOV     R1,R0      ;SET SCOPE PTR
    
```



```
3840 020362 024040      CMP      -(R0),-(R0)      ;ADJUST R0
3841 020364 010005      MOV      R0,R5           ;R5 POINTS TO DEST DATA
3842 020366 024545      CMP      -(R5),-(R5)     ;SUB 4 FROM R5
3843 020370 010015      MOV      R0,(R5)        ;R5 POINTS TO ADDRESS OF DEST DATA
3844 020372 010502      MOV      R5,R2
3845 020374 010004      MOV      R0,R4           ;R4 POINTS TO DEST DATA
3846 020376 005740      TST      -(R0)
3847 020400 010003      MOV      R0,R3           ;R3 POINTS TO SOURCE DATA
3848 020402 010042      MOV      R0,-(R2)       ;R2 POINTS TO ADDRESS OF SOURCE DATA
3849 020404 005013      CLR      (R3)           ;PRESET SOURCE DATA
3850 020406 005014      CLR      (R4)           ;PRESET DEST DATA
3851
3852 020410 000277      SCC
3853 020412 000244      CLZ
3854 020414 163235      SUB      @ (R2)+,@ (R5)+ ; (R3)=000000,(R4)=000000, CC=0100
3855 020416 103402      BCS     SUB3
3856 020420 102401      BVS     SUB3
3857 020422 001401      BEQ     .+4
3858 020424 104000      SUB3:  HLT
3859
3860 020426 052752 100000      BIS     #100000,@-(R2)   ; (R3)=100000
3861 020432 062755 000001      ADD     #1,@-(R5)       ; (R4)=000001
3862 020436 163235      SUB     @ (R2)+,@ (R5)+ ; (R3)=100000,(R4)=100001, CC=1011
3863 020440 103002      BCC     SUB3A
3864 020442 102001      BVC     SUB3A
3865 020444 100401      BMI     .+4
3866 020446 104000      SUB3A: HLT
3867
3868 020450 005414      NEG     (R4)            ; (R4)=077777
3869 020452 035255      BIT     @-(R2),@-(R5)  ; (R3)=100000,(R4)=077777
3870 020454 001401      BEQ     .+4
3871 020456 104000      HLT
3872 020460 023235      CMP     @ (R2)+,@ (R5)+
3873 020462 102401      BVS     .+4
3874 020464 104000      HLT
3875 020466 005152      COM     @-(R2)
3876 020470 000257      CCC
3877 020472 063255      ADD     @ (R2)+,@-(R5)
3878 020474 102001      BVC     ADD3
3879 020476 100401      BMI     .+4
3880 020500 104000      ADD3:  HLT
3881 020502 000261      SEC
3882 020504 045235      BIC     @-(R2),@ (R5)+ ; (R3)=077777,(R4)=100000
3883 020506 103001      BCC     BIC3
3884 020510 100401      BMI     .+4
3885 020512 104000      BIC3:  HLT
3886
3887 020514 005155      COM     @-(R5)          ; (R4)=077777
3888 020516 023235      CMP     @ (R2)+,@ (R5)+ ; (R3)=077777,(R4)=077777
3889 020520 001401      BEQ     .+4
3890 020522 104000      HLT
3891
3892
3893
3894 020524 000004      TST25: SCOPE
3895 020526 112737 000025 001202      MOV     #25,@#STSTNM ;LOAD TEST NUMBER
```

```
3896 020534 013737 001202 177570      MOV    @#1STSTM,@#DISPLAY      ;;DISPLAY TEST NUMBER
3897 020542 000406                      BR     1$                      ;RESERVE SPACE FOR ADDRESS AND DATA
3898 020544 000000                      .WORD 0                        ;CONTAINS ADDRESS OF SOURCE DATA (EVEN BYTE)
3899 020546 000000                      .WORD 0                        ;CONTAINS ADDRESS OF SOURCE DATA (ODD BYTE)
3900 020550 000000                      .WORD 0                        ;CONTAINS ADDRESS OF DEST DATA (EVEN BYTE)
3901 020552 000000                      .WORD 0                        ;CONTAINS ADDRESS OF DEST DATA (ODD BYTE)
3902 020554 000000                      .WORD 0                        ;CONTAINS SOURCE DATA
3903 020556 000000                      .WORD 0                        ;CONTAINS DEST DATA
3904
3905 020560 010700                      1$:  MOV    PC,R0
3906 020562 024040                      CMP    -(R0),-(R0)            ;R0=ADDRESS OF DEST DATA
3907 020564 010003                      MOV    R0,R3                  ;R3
3908 020566 010305                      MOV    R3,R5                  ;R5
3909 020570 005743                      TST   -(R3)                   ;SUB 2 FROM R3
3910 020572 010043                      MOV    R0,-(R3)              ;R3 POINTS TO ADDRESS OF DEST DATA
3911 020574 005213                      INC   (R3)                   ;ODD BYTE
3912 020576 010043                      MOV    R0,-(R3)              ;EVEN BYTE
3913 020600 010304                      MOV    R3,R4
3914 020602 005740                      TST   -(R0)                   ;R0=ADDRESS OF SOURCE DATA
3915 020604 010044                      MOV    R0,-(R4)              ;R4 POINTS TO ADDRESS OF SOURCE DATA
3916 020606 005214                      INC   (R4)                   ;ODD BYTE
3917 020610 010044                      MOV    R0,-(R4)              ;EVEN BYTE
3918
3919 020612 000261                      SEC                           ;SET CARRY
3920 020614 012734 177001              MOV    #177001,@(R4)+
3921 020620 112734 000200              MOV    #200,@(R4)+           ;SOURCE DATA=100001
3922 020624 115433                      MOV    @-(R4),@(R3)+
3923 020626 115433                      MOV    @-(R4),@(R3)+         ;DEST DATA=000600
3924 020630 103401                      BCS   .+4
3925 020632 104000                      HLT
3926 020634 022715 000600              CMP    #600,(R5)             ;ERROR! MOV DOES AFFECT C BIT IN PSW
3927 020640 001401                      BEQ   .+4                     ;CHECK DEST DATA
3928 020642 104000                      HLT
3929 020644 024343                      CMP    -(R3),-(R3)           ;ERROR! INCORRECT RESULT
3930 020646 153433                      BISB  @(R4)+,@(R3)+           ;POINT R4 BACK TO EVEN BYTE
3931 020650 153433                      BISB  @(R4)+,@(R3)+
3932 020652 022715 100601              CMP    #100601,(R5)         ;DEST DATA=100601
3933 020656 001401                      BEQ   .+4                     ;CHECK RESULT
3934 020660 104000                      HLT
3935 020662 145453                      BICB  @-(R4),@-(R3)          ;ERROR! INCORRECT DEST DATA AFTER BISB
3936 020664 145453                      BICB  @-(R4),@-(R3)
3937 020666 133433                      BITB  @(R4)+,@(R3)+
3938 020670 001002                      BNE   BITB3
3939 020672 135433                      BITB  @-(R4),@(R3)+
3940 020674 001001                      BNE   .+4
3941 020676 104000                      BITB3: HLT
3942
3943 020700 123453                      CMPB  @(R4)+,@-(R3)
3944 020702 001002                      BNE   CMPB3
3945 020704 123453                      CMPB  @(R4)+,@-(R3)
3946 020706 001401                      BEQ   .+4
3947 020710 104000                      CMPB3: HLT
3948
3949 *****
3950 *****
3951 *****
TST26: SCOPE
```

```

3952 020714 112737 000026 001202      MOVB  #26,@%STSTNM      ;LOAD TEST NUMBER
3953 020722 013737 001202 177570      MOV  @%STSTNM,@%DISPLAY ;:DISPLAY TEST NUMBER
3954 020730 000402                BR    .+6                ;RESERVE TWO LOCATIONS
3955 020732 000000      SDATA: .WORD 0          ;RESERVED FOR SOURCE DATA
3956 020734 000000      DDATA: .WORD 0          ;RESERVED FOR DESTINATION DATA
3957
3958 020736 013702 001534      MOV  @%FACTOR,R2        ;GET RELOCATION FACTOR AND USE AS AN
3959 020742 010205                MOV  R2,R5              ;INDEX VALUE TO POINT TO DATA
3960 020744 005065 020734      CLR  DDATA(5)          ;PRESET DESTINATION DATA
3961 020750 012762 000001 020732      MOV  #1,SDATA(2)       ;THIS ROUTINE PUT A 1 BIT INTO EVERY
3962 020756 056265 020732 020734 1$:  BIS  SDATA(2),DDATA(5) ;OTHER BIT POSITION IN THE DEST-
3963 020764 006362 020732                ASL  SDATA(2)          ;INATION ADDRESS (52525)
3964 020770 006362 020732                ASL  SDATA(2)
3965 020774 103370                BCC  1$
3966 020776 022765 052525 020734      CMP  #52525,DDATA(5) ;CHECK RESULT
3967 021004 001401                BEQ  .+4
3968 021006 104000                HLT                          ;ERROR! INCORRECT RESULT
3969 021010 012762 177777 020732      MOV  #-1,SDATA(2)
3970 021016 046562 020734 020732      BIC  DDATA(5),SDATA(2) ;SOURCE DATA=125252
3971 021024 036265 020732 020734      BIT  SDATA(2),DDATA(5)
3972 021032 001401                BEQ  .+4
3973 021034 104000                HLT                          ;ERROR! BIT INST FAILED
3974 021036 006365 020734      ASL  DDATA(5)          ;DDATA=125252
3975 021042 026265 020732 020734      CMP  SDATA(2),DDATA(5)
3976 021050 001401                BEQ  .+4
3977
3978 021052 104000                HLT                          ;ERROR! CMP INST FAILED
3979 021054 000257                CCC
3980 021056 066265 020732 020734      ADD  SDATA(2),DDATA(5)
3981 021064 103002                BCC  ADD6
3982 021066 102001                BVC  ADD6
3983 021070 100001                BPL  .+4
3984 021072 104000      ADD6: HLT
3985
3986 021074 006362 020732      ASL  SDATA(2)          ;SDATA=52524
3987 021100 166265 020732 020734      SUB  SDATA(2),DDATA(5)
3988 021106 103401                BCS  SUB6
3989 021110 001401                BEQ  .+4
3990 021112 104000      SUB6: HLT
3991
3992 021114 112700 000377                MOVB #377,R0            ;R0=177777 (MOVB %R EXTENDS SIGN)
3993 021120 010062 020732      MOV  R0,SDATA(2)
3994 021124 012765 177777 020734      MOV  #-1,DDATA(5)
3995 021132 166500 020734      SUB  DDATA(5),P0
3996 021136 001401                BEQ  .+4
3997 021140 104000                HLT
3998 021142 066265 020732 020734 1$:  ADD  SDATA(2),DDATA(5)
3999 021150 006362 020732      ASL  SDATA(2)
4000 021154 005162 020732      COM  SDATA(2)
4001 021160 036265 020732 020734      BIT  SDATA(2),DDATA(5)
4002 021166 001401                BEQ  .+4
4003 021170 104000                HLT
4004 021172 005162 020732      COM  SDATA(2)
4005 021176 026265 020732 020734      CMP  SDATA(2),DDATA(5)
4006 021204 001401                BEQ  .+4
4007 021206 104000                HLT
    
```

```
4008 021210 026200 020732          CMP      SDATA(2),R0
4009 021214 001352          BNE      1$
4010                               ;:*****
4011                               ;:*TEST 27      CHECK BINARY BYTE OPS USING ADDRESS MODE 6
4012                               ;:*****
4013 021216 000004          TST27:  SCOPE
4014 021220 112737 000027 001202      MOV      #27,@#STSTNM          ;LOAD TEST NUMBER
4015 021226 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY  ;;DISPLAY TEST NUMBER
4016                               ;NOTE: SDATA(2), AND DDATA(4) REFERENCE EVEN BYTE OF SOURCE & DEST DATA
4017                               ;AND SDATA(3), AND DDATA(5) REFERENCE ODD BYTE OF SOURCE & DEST DATA
4018
4019 021234 013702 001534          MOV      @#FACTOR,R2          ;GET INDEX VALUE
4020 021240 010204          MOV      R2,R4                ;R2 FOR SOURCE EVEN BYTE INDEX, R4 FOR
4021 021242 010403          MOV      R4,R3                ;DEST ODD BYTE, R3 FOR SOURCE EVEN
4022 021244 005203          INC      R3                    ;AND R5 FOR DEST ODD BYTE
4023 021246 010305          MOV      R3,R5
4024 021250 000261          SEC                               ;SET CARRY
4025 021252 012762 125252 021374      MOV      #125252,SDATAB(2)
4026 021260 112763 177125 021374      MCVB    #177125,SDATAB(3)    ;SOURCE DATA = 052652
4027 021266 016264 021374 021376      MOV      SDATA(2),DDATAB(4)
4028 021274 052764 125125 021376      BIS      #125125,DDATAB(4)  ;DEST DATA = 177777
4029 021302 136263 021374 021374      BITB    SDATA(2),SDATAB(3)
4030 021310 001401          BEQ      .+4
4031 021312 104000          BITB6: HLT
4032
4033 021314 146264 021374 021376      BICB    SDATA(2),DDATAB(4)
4034 021322 103401          BCS      .+4
4035 021324 104000          HLT                               ;ERROR MOV,BIS,BIT;BIC DO NOT AFFECT 'C'
4036 021326 126364 021374 021376      CMPB    SDATA(3),DDATAB(4)
4037 021334 001401          BEQ      .+4
4038 021336 104000          HLT
4039
4040 021340 146365 021374 021376      BITB    SDATA(3),DDATAB(5)
4041 021346 126265 021374 021376      CMPB    SDATA(2),DDATAB(5)
4042 021354 001401          BEQ      .+4
4043 021356 104000          HLT
4044
4045 021360 136564 021376 021376      BITB    DDATA(5),DDATAB(4)
4046 021366 001401          BEQ      .+4
4047 021370 104000          HLT
4048 021372 000415          BR      UB7                    ;RESERVE TWO WORDS
4049 021374 000000          SDATA: .WORD 0                ;RESERVED FOR SOURCE DATA
4050 021376 000000          DDATA: .WORD 0                ;RESERVED FOR DEST DATA
4051
4052                               ;:*****
4053                               ;:*TEST 30      CHECK BINARY WORD OPS USING ADDRESS MODE 7
4054                               ;:*      R2=ADDRESS OF SOURCE DATA, AND R3= ADDRESS OF DEST DATA
4055                               ;:*****
4056 021400 000004          TST30:  SCOPE
4057 021402 112737 000030 001202      MOV      #30,@#STSTNM          ;LOAD TEST NUMBER
4058 021410 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY  ;;DISPLAY TEST NUMBER
4059 021416 000000          SBIN7: .WORD 0                ;CONTAINS ADDRESS OF SOURCE DATA
4060 021420 000000          DBIN7: .WORD 0                ;CONTAINS ADDRESS OF DEST DATA
4061 021422 000000          .WORD 0                        ;CONTAINS SOURCE DATA
4062 021424 000000          .WORD 0                        ;CONTAINS DEST DATA
4063
```

```
4064 021426 010700      UB7:  MOV    PC,R0
4065 021430 024040      CMP    -(R0),-(R0)
4066 021432 010002      MOV    R0,R2
4067 021434 024242      CMP    -(R2),-(R2)
4068 021436 010012      MOV    R0,(R2)
4069 021440 010203      MOV    R2,R3
4070 021442 024043      CMP    -(R0),-(R3)
4071 021444 010013      MOV    R0,(R3)
4072
4073 021446 000261      SEC
4074 021450 012777 100000 177740      MOV    #100000,@SBIN7 ;SOURCE DATA = 100000
4075 021456 017777 177734 177734      MOV    @SBIN7,@DBIN7 ;DEST DATA = 100000
4076 021464 103001      BCC   MOV7
4077 021466 100401      BMI  .+4
4078 021470 104000      MOV7: HLT
4079 021472 006377 177722      ASL   @DBIN7 ;DEST DATA = 000000
4080 021476 102001      BVC  .+4
4081 021500 001401      BEQ  .+4
4082 021502 104000      HLT
4083
4084 021504 027777 177706 177706      CMP   @SBIN7,@DBIN7 ;(R2)=100000,(R3)=000000
4085 021512 103402      BCS  CMP7
4086 021514 102401      BVS  CMP7
4087 021516 100401      BMI  .+4
4088 021520 104000      CMP7: HLT
4089
4090 021522 167777 177670 177670      SUB   @SBIN7,@DBIN7 ;(R2)-100000,(R3)=100000
4091 021530 103003      BCC  SUB7
4092 021532 102002      BVC  SUB7
4093 021534 001401      BEQ  SUB7
4094 021536 100401      BMI  .+4
4095 021540 104000      SUB7: HLT
4096
4097 021542 006277 177650      ASR   @SBIN7 ;(R2)=140000
4098 021546 067777 177644 177644      ADD   @SBIN7,@DBIN7 ;(R2)=140000,(R3)=040000
4099 021554 103003      BCC  ADD7
4100 021556 102002      BVC  ADD7
4101 021560 001401      BEQ  ADD7
4102 021562 100001      BPL  .+4
4103 021564 104000      ADD7: HLT
4104
4105 021566 047777 177624 177624      BIC   @SBIN7,@DBIN7 ;(R2)=140000,(R3)=000000
4106 021574 001401      BEQ  .+4
4107 021576 104000      HLT
4108
4109 021600 057777 177612 177612      BIS   @SBIN7,@DBIN7 ;(R2)=140000,(R3)=140000
4110 021606 100401      BMI  .+4
4111 021610 104000      HLT
4112
4113 021612 027777 177600 177600      CMP   @SBIN7,@DBIN7
4114 021620 001401      BEQ  .+4
4115 021622 104000      HLT
4116
4117
4118
4119
:*****
:*TEST 31 SOME MISCELLANEOUS OPERATIONS INVOLVING THE PC
:* NOTE: NONE OF THESE OPERATIONS SHOULD AFFECT THE PC
:*****
```

4120 021624 000004
4121 021626 112737 000031 001202
4122 021634 013737 001202 177570
4123 021642 005000
4124 021644 005067 000072
4125 021650 010707
4126 021652 120707
4127 021654 030707
4128 021656 060007
4129 021660 105707
4130 021662 005507
4131 021664 021007
4132 021666 131007
4133 021670 062707 000000
4134 021674 023707 001534
4135 021700 133707 001534
4136 021704 000240
4137
4138
4139 021706 163707 001534
4140 021712 063707 001534
4141 021716 000240
4142 021720 024607
4143 021722 132607
4144 021724 026707 000012
4145 021730 166707 000006
4146 021734 046707 000002
4147 021740 000401
4148 021742 000000
4149 021744 000004
4150 021746 010702
4151 021750 062702 000012
4152 021754 012707 045764
4153 021760 000000
4154
4155
4156
4157
4158
4159 021762 012767 000001 157536
4160 021770 000004
4161 021772 112737 000032 001202
4162 022000 013737 001202 177570
4163
4164
4165
4166 022006 010700
4167 022010 005740
4168 022012 010037 001540
4169 022016 010700
4170 022020 162700 022020
4171 022024 010037 001534
4172 022030 010737 001212
4173 022034 062737 000030 001212
4174 022042 013737 001212 001210
4175 022050 105737 001530

TST31: SCOPE
MOV #31,@\$TSTNM ;LOAD TEST NUMBER
MOV @\$TSTNM,@\$DISPLAY ;;DISPLAY TEST NUMBER
CLR R0
CLR 1\$
MOV PC,PC
CMPB PC,PC
BIT PC,PC
ADD R0,PC
TSTB PC
ADC PC
CMP (R0),PC
BITB (R0),PC
ADD #0,PC
CMP @\$FACTOR,PC
BITB @\$FACTOR,PC
NOP
;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)
SUB @\$FACTOR,PC ;JUMPS TO UNRELOCATED CODE
ADD @\$FACTOR,PC ;RETURNS
NOP
CMP -(SP),PC
BITB (SP)+,PC
CMP 1\$,PC
SUB 1\$,PC
BIC 1\$,PC
BR .+4 ;BRANCH OVER 1\$
1\$: 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
;*****
MOV #1,\$TIMES ;;DO 1 ITERATION
TST32: SCOPE
MOV #32,@\$TSTNM ;LOAD TEST NUMBER
MOV @\$TSTNM,@\$DISPLAY ;;DISPLAY TEST NUMBER
;333333333333 FIRST ADDRESS TO BE RELOCATED 333333333
RFL3: 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 @ANEXC ;BR IF TEST CODE TO BE EXECUTED

```
4176 022054 001402 BEQ .+6
4177 022056 000167 002314 JMP RELE3
4178 022062 012703 125252 MOV #125252,R3
4179 022066 010304 MOV R3,R4 ;R3=R4=125252
4180 022070 140304 BICB R3,R4 ;R3=125252,R4=125000
4181 022072 022704 125000 CMP #125000,R4 ;CHECK RESULT
4182 022076 001401 BEQ .+4
4183 022100 104000 HLT
4184
4185 022102 005004 CLR R4 ;R3=125252,R4=0
4186 022104 150304 BISB R3,R4 ;R3=125252,R4=000252
4187 022106 022704 000252 CMP #252,R4 ;CHECK RESULT
4188 022112 001401 BEQ .+4
4189 022114 104000 HLT
4190
4191 022116 110404 MOVB R4,R4 ;R4=177652
4192 022120 022704 177652 CMP #177652,R4 ;CHECK RESULT
4193 022124 001401 BEQ .+4
4194 022126 104000 HLT
4195
4196 022130 132704 177525 BITB #177525,R4
4197 022134 001401 BEQ .+4
4198 022136 104000 HLT
4199
4200 022140 105104 COMB R4 ;R4=177525
4201 022142 110404 MOVB R4,R4 ;R4=000125
4202 022144 022704 000125 CMP #125,R4 ;CHECK RESULT
4203 022150 001401 BEQ .+4
4204 022152 104000 HLT
4205
4206 022154 150304 BISB R3,R4 ;R3=125252,R4=000377
4207 022156 105204 INCB R4
4208 022160 001401 BEQ .+4
4209 022162 104000 HLT
4210
4211 *****
4212 : *TEST 33 CHECK BINARY BYTE OPS USING ADDRESS MODE 7
4213 : *****
4214 TST33: SCOPE
4215 MOVB #33,@$TSTNM ;LOAD TEST NUMBER
4216 MOV @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
4217 BR BINB7 ;RESERVE SPACE FOR ADDRESSES & DATA
4218 SBINB7: .WORD 0 ;CONTAINS ADDRESS OF SOURCE EVEN BYTE
4219 .WORD 0 ;CONTAINS ADDRESS OF SOURCE ODD BYTE
4220 .WORD 0 ;CONTAINS ADDRESS OF DEST EVEN BYTE
4221 .WORD 0 ;CONTAINS ADDRESS OF DEST ODD BYTE
4222 DBINB7: .WORD 0 ;CONTAINS SOURCE DATA
4223 .WORD 0 ;CONTAINS DEST DATA
4224 BINB7: MOV PC,R0
4225 CMP -(R0),-(R0) ;R0 - ADDRESS OF DEST DATA
4226 MOV R0,-6(R0) ;LOAD ADDRESS OF DEST EVEN BYTE DATA
4227 MOV R0,-4(R0)
4228 INC -4(R0) ;LOAD ADDRESS OF DEST ODD BYTE DATA
4229 TST -(R0) ;R0=ADDRESS OF SOURCE DATA
4230 MOV R0,-10(R0) ;LOAD ADDRESS OF SOURCE EVEN BYTE DATA
4231 MOV R0,-6(R0)
```

```
4232 022252 005260 177772      INC      -6(R0)          ;LOAD ADDRESS OF SOURCE ODD BYTE DATA
4233
4234 022256 005002      CLR      R2              ;SET INDEX REGISTERS
4235 022260 012703 000002      MOV      #2,R3           ;@SBINB7(2);@SBINB7(3) REFERENCE EVFN 8
4236 022264 012704 177774      MOV      #-4,R4          ;ODD BYTE SOURCE DATA; @DBINB7(4);@DBINB7(5)
4237 022270 012705 177776      MOV      #-2,R5          ;REFERENCE DEST EVEN& ODD BYTE DATA
4238
4239
4240 022274 005020      CLR      (R0)+           ;PRESET SOURCE DATA
4241 022276 005010      CLR      (R0)            ;PRESET DEST DATA
4242 022300 013746 001534      MOV      @#FACTOR,-(SP)  ;GET RELOCATION FACTOR
4243 022304 061602      ADD      (SP),R2         ;AND ADD TO INDEX VALUES
4244 022306 061603      ADD      (SP),R3
4245 022310 061604      ADD      (SP),R4
4246 022312 062605      ADD      (SP)+,R5
4247
4248 022314 112773 177777 022204      MOV      #-1,@SBINB7(3)  ;SRC DATA = 177400
4249 022322 132772 000377 022204      BIT      #377,@SBINB7(2) ;CHECK THAT EVEN BYTE WAS NOT AFFECTED
4250 022330 001401      BEQ      .+4             ;BY MOV B INSTRUCTION
4251 022332 104000      HLT
4252
4253 022334 157374 022204 022214      BIS      @SBINB7(3),@DBINB7(4)
4254 022342 105274 022214      INCB    @DBINB7(4)      ;CHECK THAT BIS SET ALL BITS
4255 022346 001401      BEQ      .+4
4256 022350 104000      HLT
4257
4258 022352 105375 022214      DECB    @DBINB7(5)      ;DEST DATA = 177400
4259 022356 005274 022214      INC     @DBINB7(4)      ;DEST DATA = 177401
4260 022362 127375 022204 022214      CMP     @SBINB7(3),@DBINB7(5)
4261 022370 001401      BEQ      .+4
4262 022372 104000      HLT
4263
4264 022374 147375 022204 022214      BIC     @SBINB7(3),@DBINB7(5)
4265 022402 001401      BEQ      .+4
4266 022404 104000      HLT
4267
4268 022406 105073 022204      CLRB    @SBINB7(3)      ;SRC DATA = 000000
4269
4270
4271 022412 157473 022214 022204      ;THIS ROUTINE SETS ALL BITS IN THE SOURCE ODD BYTE BY BISING A BIT FROM
4272 022420 106174 022214      ;THE DEST EVEN BYTE INTO THE SOURCE ODD BYTE
4273 022424 103372      BIS7:  BIS      @DBINB7(4),@SBINB7(3)
4274 022426 022772 177400 022204      ROL     @DBINB7(4)
4275 022434 001401      BCC     BIS7
4276 022436 104000      CMP     #177400,@SBINB7(2) ;CHECK RESULT
4277
4278 022440 000372 022204      BEQ     .+4
4279 022444 112775 000200 022214      SWAB   @SBINB7(2)      ;SRC DATA = 000377
4280
4281 022452 147572 022214 022204      MOV     #200,@DBINB7(5) ;DEST DATA = 100000
4282 022460 106075 022214      BIC7:  BIC     @DBINB7(5),@SBINB7(2)
4283 022464 103372      RORB   @DBINB7(5)
4284 022466 005772 022204      BCC     BIC7
4285 022472 001401      TST    @SBINB7(2)
4286 022474 104000      BEQ     .+4
4287      HLT
```



```

4288 022476 012702 000001      OAERR:  MOV    #1,R2          ;LOAD R2 WITH ODD #
4289 022502 010703             MOV    PC,R3
4290 022504 000401             BR     .+4              ;RESERVE SPACE FOR A WORD
4291 022506 000000             .WORD  0                ;WILL CONTAIN AN ODD ADDRESS
4292 022510 005723             TST   (R3)+            ;STEP R3 TO POINT TO WORD ABOVE
4293 022512 010313             MOV    R3,(R3)
4294 022514 005213             INC   (R3)             ;AND MAKE ODD
4295 022516 012737 022644 000004  MOV    #1$,@ERRVEC     ;SET ODD ADDRESS & RESERVED INSTRUCTION
4296 022524 063737 001534 000004  ADD   @FACTOR,@ERRVEC
4297 022532 013737 000004 000010  MOV   @ERRVEC,@RESVEC ;TO TRAP TO 1$ BELOW
4298
4299 022540 000277             SCC                    ;SET ALL CC'S
4300 022542 160212             SUB   R2,(R2)
4301 022544 104000             HLT
4302 022546 060222             ADD   R2,(R2)+
4303 022550 104000             HLT
4304 022552 006342             ASI   -(R2)
4305 022554 104000             HLT
4306 022556 106512             MFPD (R2)
4307 022560 104000             HLT
4308 022562 170412             CLRF (R2)
4309 022564 104000             HLT
4310 022566 042202             BIC   (R2)+,R2
4311 022570 104000             HLT
4312 022572 164202             SUB   -(R2),R2
4313 022574 104000             HLT
4314 022576 155202             BISB @-(R2),R2
4315 022600 104000             HLT
4316 022602 105532             ADCB @R2)+
4317 022604 104000             HLT
4318 022606 163302             SUB   @R3)+,R2
4319 022610 104000             HLT
4320 022612 005733             TST   @R3)+
4321 022614 104000             HLT
4322 022616 106533             MFPD @R3)+
4323 022620 104000             HLT
4324 022622 170453             CLRD @-(R3)
4325 022624 104000             HLT
4326 022626 137702 177775       BITB @.+1,R2
4327 022632 104000             HLT
4328 022634 105477 177773       NEGB @.-1
4329 022640 104000             HLT
4330 022642 000406             BR    2$
4331
4332 022644 062716 000002 1$:  ADD   #2,(SP)          ;ADJUST RETURN PC
4333 022650 052766 000017 000002  BIS   #17,2(SP)        ;SET CONDITION CODES ON RETURN
4334 022656 000002             RTI
4335
4336 022660 012706 000700 2$:  MOV   #SUPSTK,SP       ;RESET STACK PTR
4337 022664 012737 064270 000004  MOV   #ERPRT,@ERRVEC  ;RESET TIME OUT VECTOR
4338 022672 012737 064216 000010  MOV   #RESERR,@RESVEC
4339
4340 :*****
4341 :*TEST 34 CHECK JUMP INSTRUCTIONS
4342 :*****
4342 022700 000004  TST34: SCOPE
4343 022702 112737 000034 001202  MOVB  #34,@$TSTNM     ;LOAD TEST NUMBER

```

4344	022710	013737	001202	177570	MOV	@#STSTNM,@#DISPLAY	::DISPLAY TEST NUMBER
4345	022716	010700			MOV	PC,R0	
4346	022720	062700	000012		ADD	#12,R0	:SET ADDRESS FOR JMP INST
4347	022724	000277			SCC		:SET CC'S
4348	022726	000110			JMP	(R0)	
4349	022730	000402			BR	+.6	
4350	022732	000250			CLN		:JMP INST JUMPS HERE
4351	022734	000775			BR	-.4	
4352							
4353	022736	103003			BCC	JMP1	
4354	022740	102002			BVC	JMP1	
4355	022742	001001			BNE	JMP1	
4356	022744	100001			BPL	+.4	
4357	022746	104000			JMP1:	HLT	:ERROR. INCORRECT CC'S AFTER JMP
4358							
4359	022750	005002			CLR	R2	:SET INDICATOR
4360	022752	010703			MOV	PC,R3	
4361	022754	000401			BR	+.4	:RESERVE WORD FOR JMP ADDRESS
4362	022756	000000			.WORD	0	:CONTAINS ADDRESS FOR JMP INST
4363	022760	005723			TST	(R3)+	
4364	022762	010313			MOV	R3,(R3)	
4365	022764	010300			MOV	R3,R0	
4366	022766	062713	000022		ADD	#22,(R3)	:(R3) IS JMP ADDRESS
4367	022772	010300			MOV	R3,R0	
4368	022774	000133			JMP	@(R3)+	:JUMP TO ADDRESS CONTAINED IN R3
4369	022776	000402			BR	+.6	
4370	023000	005102			COM	R2	:COMPLEMENT INDICATOR
4371	023002	000775			BR	-.4	
4372	023004	005202			INC	R2	:CHECK INDICATOR
4373	023006	001003			BNE	JMP3	
4374	023010	005720			TST	(R0)+	
4375	023012	020003			CMP	R0,R3	:CHECK AUTO-INC R3
4376	023014	001401			BEQ	+.4	
4377	023016	104000			JMP3:	HLT	
4378							
4379	023020	005002			CLR	R2	:SET INDICATOR
4380	023022	010704			MOV	PC,R4	:SET UP JMP REGISTER
4381	023024	010400			MOV	R4,R0	:SET UP CHECK REGISTER
4382	023026	000402			BR	1\$	
4383	023030	005102			COM	R2	:COMPLEMENT INDICATOR
4384	023032	000403			BR	2\$	
4385	023034	022424			1\$:	CMP (R4)+,(R4)+	
4386	023036	005724			TST	(R4)+	:R4=JMP ADDRESS
4387	023040	000144			JMP	-(R4)	:USE R4 AS ADDRESS
4388	023042	005202			2\$:	INC R2	:CHECK INDICATOR
4389	023044	001003			BNE	JMP4	
4390	023046	022020			CMP	(R0)+,(R0)+	
4391	023050	020004			CMP	R0,R4	:CHECK AUTO-DEC R4
4392	023052	001401			BEQ	+.4	
4393	023054	104000			JMP4:	HLT	
4394							
4395	023056	010703			MOV	PC,R3	
4396	023060	000401			BR	+.4	:RESERVE WORD FOR JMP ADDRESS
4397	023062	000000			1\$:	.WORD 0	:CONTAINS JUMP ADDRESS
4398	023064	005723			TST	(R3)+	
4399	023066	010313			MOV	R3,(R3)	

```
4400 023070 062723 000016      ADD    #16,(R3)+
4401 023074 010300              MOV    R3,R0          ;LOAD CHECK REGISTER
4402 023076 000402              BR     3$
4403 023100 005102      2$:   COM    R2
4404 023102 000401              BR     4$
4405 023104 000153      3$:   JMP    @-(R3)      ;JUMP TO 2$ VIA 1$ ABOVE
4406 023106 005202      4$:   INC    R2          ;CHECK INDICATOR
4407 023110 001003              BNE   JMP5
4408 023112 005740              TST   -(R0)
4409 023114 020003              CMP   R0,R3          ;CHECK AUTO-DEC R3
4410 023116 001401              BEQ   .+4
4411 023120 104000      JMP5:  HLT
4412
4413 023122 000402              BR     2$
4414 023124 005102      1$:   COM    R2          ;COMPLEMENT INDICATOR
4415 023126 000402              BR     3$
4416 023130 000167 17777C      2$:   JMP    1$
4417 023134 005202      3$:   INC    R2
4418 023136 001401              BEQ   .+4
4419 023140 104000      JMP6:  HLT
4420
4421 023142 012767 023160 000020      MOV    #1$,7$        ;SET UP JMP ADDRESS
4422 023150 063767 001534 000012      ADD    @#FACTOR,7$   ;ADD RELOCATION FACTOR
4423 023156 000402              BR     2$            ;GO TO JMP @7$ INST
4424 023160 005102      1$:   COM    R2          ;COMPLEMENT INDICATOR
4425 023162 000403              BR     3$            ;GO TO CHECK ROUTINE
4426 023164 000177 000000      2$:   JMP    @7$        ;JMP TO 1$ ABOVE VIA 7$
4427 023170 000000      7$:   .WORD 0          ;CONTAINS JMP ADDRESS
4428 023172 005202      3$:   INC    R2          ;CHECK INDICATOR
4429 023174 001401              BEQ   .+4
4430 023176 104000      JMP7:  HLT
4431
4432      ;*****
4433      ;*TEST 35      CHECK JSR INSTRUCTIONS
4434      ;*****
4434 023200 000004      TST35: SCOPE
4435 023202 112737 000035 001202      MOV    #35,@#STSTNM ;LOAD TEST NUMBER
4436 023210 013737 001202 177570      MOV    @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
4437 023216 013705 001534      JSR1:  MOV    @#FACTOR,R5 ;GET RELOCATION FACTOR
4438 023222 012702 023254      MOV    #3$,R2        ;FORM DEST ADRS
4439 023226 060502      ADD    R5,R2          ;ADD RELOCATION FACTOR
4440 023230 000277      SCC
4441 023232 000242      CLV
4442 023234 004512      JSR    R5,(R2)        ;GO TO 3$ VIA R2
4443 023236 005702      1$:   TST   R2          ;CHECK INDICATOR
4444 023240 001017              BNE   4$            ;R2 SHOULD=0
4445 023242 023705 001534      CMP    @#FACTOR,R5   ;CHECK THAT RTS R5 RESTORED R5
4446 023246 001014              BNE   4$
4447 023250 000414              BR    JSR3          ;GO TO NEXT TEST
4448 023252 000205      2$:   RTS   R5          ;RETURN FROM SUBROUTINE
4449 023254 103011      3$:   BCC   4$          ;CHECK THAT JSR DID NOT
4450 023256 102410              BVS   4$
4451 023260 001007              BNE   4$            ;AFFECT CC'S
4452 023262 100006              BPL   4$
4453 023264 005002              CLR   R2            ;CLEAR INDICATOR
4454 023266 012704 023236      MOV    #1$,R4        ;GET UNRELOCATED RETURN ADDRESS
4455 023272 061604              ADD    (SP),R4       ;ADD RELOCATION FACTOR (OLD R5)
```

```

4456 023274 020405          CMP      R4,R5          ;CHECK THAT OLD R5 WAS PLACED ON THE
4457 023276 001765          BEQ      2$              ;STACK, & THAT NEW R5 CONTAINS RETURN PC
4458 023300 104000          4$:      HLT              ;ERROR! ABOVE
4459
4460          ;CHECK JSR INSTRUCTION ADDRESS MODE 3
4461 023302 013704 001534      JSR3:    MOV      @#FACTOR,R4 ;GET RELOCATION FACTOR
4462 023306 005000          CLR      R0              ;SET INDICATOR
4463 023310 012705 023330      MOV      #1$,R5
4464 023314 060405          ADD      R4,R5          ;SET UP JSR DEFERRED ADRS
4465 023316 010502          MOV      R5,R2
4466 023320 012715 023346      MOV      #5$, (R5)
4467 023324 060415          ADD      R4, (R5)       ;(R5)=DEST ADRS
4468 023326 000401          BR       2$              ;RESERVE WORD FOR ADDRESS
4469 023330 000000          1$:      .WORD    0        ;CONTAINS DEST ADRS FOR JSR
4470 023332 004435          2$:      JSR      R4,@(R5)+ ;JSR TO 5$ VIA 1$ ABOVE
4471 023334 005200          3$:      INC      R0        ;CHECK INDICATOR
4472 023336 001013          BNE      6$
4473 023340 000413          BR       JSR4
4474 023342 005100          4$:      COM      R0        ;COMPLEMENT INDICATOR
4475 023344 000204          RTS      4              ;RETURN FROM SUBROUTINE
4476 023346 012703 023334          5$:      MOV      #3$,R3    ;GET UNRELOCATED RETURN ADDRESS
4477 023352 061603          ADD      (SP),R3       ;ADD RELOCATION FACTOR (OLD R4)
4478 023354 020403          CMP      R4,R3
4479 023356 001003          BNE      6$
4480 023360 005722          TST      (R2)+
4481 023362 020205          CMP      R2,R5          ;CHECK AUTO-INC R5
4482 023364 001766          BEQ      4$            ;GO TO RTS
4483 023366 104000          6$:      HLT              ;ERROR ABOVE
4484
4485          ;CHECK JSR INST ADDRESS MODE 4
4486 023370 013704 001534      JSR4:    MOV      @#FACTOR,R4
4487 023374 010405          MOV      R4,R5
4488 023376 010703          MOV      PC,R3
4489 023400 000401          BR       2$
4490 023402 000405          1$:      BR       1$
4491 023404 022323          2$:      CMP      (R3)+,(R3)+
4492 023406 000277          SCC
4493 023410 004443          JSR      R4,-(R3)       ;GO TO 2$
4494 023412 104000          3$:      HLT
4495 023414 000414          BR       JSR6           ;GO TO NEXT TEST
4496 023416 103012          4$:      BCC      5$
4497 023420 102011          BVC      5$
4498 023422 001010          BNE      5$
4499 023424 100007          BPL      5$
4500 023426 012702 023412      MOV      #3$,R2        ;GET UNRELOCATED RETURN ADDRESS
4501 023432 061602          ADD      (SP),R2       ;ADD RELOCATION FACTOR (OLD R4)
4502 023434 020204          CMP      R2,R4        ;CHECK THAT CALCULATED RETURN
4503 023436 001002          BNE      5$           ;PC = NEW R4
4504 023440 005724          TST      (R4)+
4505 023442 000204          RTS      R4
4506 023444 104000          5$:      HLT
4507
4508          ;TEST JSR INST ADDRESS MODE 6
4509 023446 000401          JSR6:    BR       2$
4510 023450 000405          1$:      BR       3$
4511 023452 010700          2$:      MOV      PC,R0
    
```

```
4512 023454 004767 177770      JSR    PC,1$
4513 023460 100407      BMI    JSR7      ;GO TO NEXT TEST
4514 023462 104000      HLT                    ;ERROR ON CC'S
4515 023464 022020      3$:  CMP    (R0)+,(R0)+
4516 023466 020016      CMP    R0,(SP)      ;CHECK THAT RETURN ADDRESS IS ON THE
4517 023470 001401      BEQ    .+4          ;STACK
4518 023472 104000      HLT
4519 023474 000270      SEN                    ;SET N
4520 023476 000207      RTS    PC
4521
4522      ;TEST JSR INST ADDRESS MODE 7
4523 023500 013746 001534      JSR7:  MOV    @#FACTOR,-(SP) ;GET RELOCATION FACTOR
4524 023504 062716 023524      ADD    #1$,(SP) ;FORM ADDRESS OF 1$ BELOW
4525 023510 000277      SCC                    ;SET ALL CC'S
4526 023512 004076 000000      JSR    R0,@(SP) ;JSR TO 1$
4527 023516 003003      BGT    3$
4528 023520 102002      BVC    3$
4529 023522 000402      BR     4$
4530
4531 023524 000200      1$:  RTS    R0      ;RETURN
4532 023526 104000      3$:  HLT                    ;ERROR!! INCORRECT CC'S
4533 023530      4$:
4534      ;*****
4535      ;*TEST 36 CHECK IOT TRAP (AND ROLB/ASLB)
4536      ;* THIS TEST CHECKS THAT THE PSW IS CORRECT AFTER THE IOT AND THAT THE
4537      ;* 'NEW'PSW (FROM IOTVEC+2) IS CORRECT.
4538      ;*****
4539 023530 000004      TST36: SCOPE
4540 023532 112737 000036 001202      MOV    #36,@#TSTNM ;LOAD TEST NUMBER
4541 023540 013737 001202 177570      MOV    @#TSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
4542 023546 012705 000022      IOTTST: MOV    #IOTVEC+2,R5
4543 023552 005000      CLR    R0
4544 023554 052740 000200      BIS    #PR4,-(R0) ;SET PRIORITY LEVEL 4 IN PSW
4545 023560 011015      MOV    (R0),(R5) ;SET IOTVEC+2 = PSW
4546 023562 011504      MOV    (R5),R4 ;SAVE IN R4
4547 023564 010746      MOV    PC,-(SP)
4548 023566 062716 000036      ADD    #1$-.,(SP)
4549 023572 012645      MOV    (SP)+,-(R5) ;LOAD IOT TRAP VECTOR
4550 023574 042710 000357      BIC    #PR7+17,(R0)
4551 023600 052710 000244      BIS    #PR5+4,(R0) ;PSW=X XXX X00 101 1x1 000
4552 023604 012003      MOV    (R0)+,R3 ;R3 = PSW ABOVE
4553 023606 010340      MOV    R3,-(R0) ;RESTORE PSW (MOV CHANGED IT)
4554 023610 000004      IOT
4555 023612 012737 054370 000020 10$:  MOV    #$$SCOPE,@#IOTVEC ;RESTORE IOT VECTOR
4556 023620 104000      HLT                    ;ERROR! IOT FAILED TO TRAP
4557 023622 000457      BR     TST37 ;:GO TO NEXT TEST
4558
4559 023624 012002      1$:  MOV    (R0)+,R2 ;GET PSW AFTER IOT TRAP
4560      ;NOTE: R0=0
4561 023626 012725 054370      MOV    #$$SCOPE,(R5)+ ;RESTORE IOTVEC
4562 023632 012715 000200      MOV    #PR4,(R5) ;AND IOTVEC+2
4563 023636 010746      MOV    PC,-(SP) ;FORM PC OF 10$ ABOVE
4564 023640 062716 177752      ADD    #10$-.,(SP)
4565 023644 022626      CMP    (SP)+,(SP)+ ;CHECK RETURN PC ON STACK
4566 023646 001036      BNE    99$
4567 023650 022603      CMP    (SP)+,R3 ;CHECK SAVED PSW
```

```
4568 023652 001034          BNE      99$
4569 023654 032703 140000    BIT      #UM,R3          ;BRANCH TO 3$ IF IN USER MODE
4570 023660 100413          BMI      3$
4571 023662 001003          BNE      2$          ;BRANCH TO 2$ IF IN SUPER MODE
4572 023664 020204          CMP      R2,R4          ;CHECK PSW AFTER IOT
4573 023666 001026          BNE      99$
4574 023670 000413          BR       4$
4575
4576 023672 042704 030000    2$:     BIC      #PUM,R4          ;CLEAR PREV MODE BITS
4577 023676 052704 010000    BIS      #PSM,R4          ;SET PREV SUPER MODE
4578 023702 020204          CMP      R2,R4          ;CHECK PSW AFTER IOT
4579 023704 001017          BNE      99$
4580 023706 000404          BR       4$
4581
4582 023710 052704 030000    3$:     BIS      #PUM,R4          ;SET PREV USER MODE
4583 023714 020204          CMP      R2,R4          ;CHECK PSW AFTER IOT
4584 023716 001012          BNE
4585
4586 023720 005002          4$:     CLR      R2
4587 023722 000261          SEC
4588 023724 106100          ROLB     R0          ;ROTATE R0
4589 023726 102376          BVC     .-2          ;UNTIL V SETS (R0 200)
4590
4591 023730 106300          ASLB     R0          ;SHIFT SHOULD SET CARRY
4592 023732 103004          BCC      99$
4593 023734 102003          BVC      99$
4594 023736 001002          BNE      99$
4595 023740 005700          TST      R0
4596 023742 001401          BEQ      .+4
4597 023744 104000          99$:    HLT
4598
4599
4600 023746 042704 000340          BIC      #PR7,R4
4601 023752 010437 177776    MOV      R4,@#PSW          ;RESTORE PSW
4602 023756 012706 000700    MOV      #SUPSTK,SP          ;RESTORE STACK PTR
4603
4604
4605
4606 023762 000004          ;*****
4607 023764 112737 000037 001202    ;*TEST 37 CHECK EMT TRAP SEQUENCE
4608 023772 013737 001202 177570    ;*****
4609
4610 024000 012737 054622 000020    TST37:  SCOPE
4611 024006 012737 000340 000022    MOV      #37,@#STSTNM          ;LOAD TEST NUMBER
4612 024014 005000          MOV      @#STSTNM,@#DISPLAY          ;;DISPLAY TEST NUMBER
4613 024016 010746          .EQUIV   IOT,HLT          ;REDEFINE HLT CALL
4614 024020 062716 000030          MOV      #ERROR,@#IOTVEC          ;SETUP VECTOR
4615 024024 012637 000030          MOV      #PR7,@#IOTVEC+2
4616 024030 000262          CLR      R0
4617 024032 013737 177776 000032    MOV      PC,-(SP)
4618 024040 000265          MOV      #EMT1-,(SP)
4619 024042 104000          ADD      (SP)+,@#EMTVEC
4620 024044 001433          SEV
4621 024046 000004          MOV      @#PSW,@#EMTVEC+2          ;SET V
4622 024050 102027          +SEZ!SEC          ;RETAIN CURRENT PSW ON TRAP
4623 024052 105100          EMT
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
```

```
4624 024054 105500 ADCB R0 ;R0=000000,CC'S=0101
4625 024056 106000 RORB R0 ;R0=000200,CC'S-1010
4626 024060 102023 BVC EMT1B
4627 024062 100022 BPL EMT1B
4628 024064 000257 CCC
4629 024066 105400 NEGB R0 ;R0=000200,CC'S=1010
4630 024070 102017 BVC EMT1B
4631 024072 100016 BPL EMT1B
4632 024074 000242 CLV ;CLEAR 'V'
4633 024076 000261 SEC ;AND SET 'C'
4634 024100 105300 DECB R0 ;R0=000177,CC'S=0011
4635 024102 102012 BVC EMT1B
4636 024104 100411 BMI EMT1B
4637 024106 000242 CLV ;CLEAR 'V'
4638 024110 105200 INCB R0 ;R0=000200,CC'S-1011
4639 024112 103006 BCC EMT1B
4640 024114 102005 BVC EMT1B
4641 024116 100004 BPL EMT1B
4642 024120 000242 CLV ;CLEAR 'V'
4643 024122 106200 ASRB R0 ;SHIFT R0 UNTIL 'V' CLEARS
4644 024124 102776 BVS .-2
4645 024126 000401 BR .+4
4646 024130 000004 EMT1B: HLT ;ERROR!
4647 024132 000002 RTI ;EXIT WITH R0=000377
4648 024134 105500 EMT1C: ADCB R0 ;R0=000000
4649 024136 103003 BCC EMT1D
4650 024140 001002 BNE EMT1D
4651 024142 005700 TST R0
4652 024144 001401 BEQ .+4
4653 024146 000004 EMT1D: HLT
4654 024150 012737 054622 000030 MOV #ERROR,@EMTVEC ;RESTORE EMT TO ERROR
4655 024156 012737 000340 000032 MOV #PR7,@EMTVEC+2 ;SET PRIORITY 7 ON ERROR
4656 024164 012737 054370 000020 MOV #SCOPE,@IOTVEC ;RESTORE IOT VECTOR
4657 024172 005037 000022 CLR @IOTVEC+2
4658 .EQUIV ERROR,HLT ;REDEFINE HLT CALL
4659 *****
4660 ;*TEST 40 CHECK TRAP INSTRUCTION TRAP SEQUENCE
4661 *****
4662 024176 000004 TST40: SCOPE
4663 024200 112737 000040 001202 MOVB #40,@STSTNM ;LOAD TEST NUMBER
4664 024206 013737 001202 177570 MOV @STSTNM,@DISPLAY ;:DISPLAY TEST NUMBER
4665 024214 052737 000340 177776 BIS #PR7,@PSW ;LOCK OUT LINE CLOCK
4666 024222 052737 000340 000016 BIS #PR7,@TBITVEC+2
4667 024230 010746 MOV PC,-(SP)
4668 024232 062716 000056 ADD #TRAP1-,(SP)
4669 024236 012637 000034 MOV (SP)+,@TRAPVEC
4670 024242 000270 SEN ;SET N
4671 024244 013737 177776 000036 MOV @PSW,@TRAPVEC+2 ;RETAIN CURRENT PSW ON TRAP
4672 024252 000261 SEC ;SET CARRY
4673 024254 010700 MOV PC,R0
4674 024256 000264 SEZ ;SET Z BIT
4675 024260 104400 TRAP ;TRAP TO TRAP1
4676 024262 103404 BCS .+12
4677 024264 012737 062144 000034 MOV #TRAP,@TRAPVEC ;RESTORE TRAP VECTOR
4678 024272 104000 HLT
4679 024274 001404 BEQ .+12
```

```
4680 024276 012737 062144 000034      MOV    #STRAP,@TRAPVEC      ;RESTORE TRAP VECTOR
4681 024304 104000                      HLT
4682 024306 000420                      BR    TRAP1C
4683 024310 100404                      TRAP1: BMI    .+12          ;N BIT GOT SET ON TRAP
4684 024312 012737 062144 000034      MOV    #STRAP,@TRAPVEC      ;RESTORE TRAP VECTOR
4685 024320 104000                      HLT
4686 024322 062700 000004              ADD    #4,R0
4687 024326 020016                      CMP    R0,(SP)              ;CHECK LOW BYTE OF RETURN PC ON
4688 024330 001404                      BEQ    .+12                  ;STACK
4689 024332 012737 062144 000034      MOV    #STRAP,@TRAPVEC      ;RESTORE TRAP VECTOR
4690 024340 104000                      HLT
4691 024342 124646                      CMPB   -(SP),-(SP)
4692 024344 032626                      BIT    (SP)+,(SP)+
4693 024346 000002                      RTI                          ;RETURN TO INST FOLLOWING TRAP (1$)
4694
4695 024350 012702 000036              TRAP1C: MOV   #TRAPVEC+2,R2   ;RESTORE VECTORS
4696 024354 012712 000340              MOV   #PR7,(R2)
4697 024360 012742 062144              MOV   #STRAP,-(R2)
4698 024364 042737 000340 000016      BIC   #PR7,@TBITVEC+2
4699 024372 105037 177776              CLRB  @PSW                  ;GO BACK TO PRIORITY 0
4700
4701 024376 000004              RELE3: SCOPE
4702 024400 010702              MOV   PC,R2
4703 024402 062702 000012              ADD   #12,R2
4704 024406 012707 043764              MOV   #RELOC,PC           ;GO RELOCATE PROGRAM CODE
4705 024412 000000              REL33: .WORD 0
4706                      ;33333333333333 LAST ADDRESS OF CODE TO BE RELOCATED 333333333333
4707
4708                      ;*****
4709                      ;*TEST 41      CHECK STACK OVERFLOW
4710                      ;*****
4711 024414 012767 000001 154704      TST41: MOV   #1,$TIMES      ;;DO 1 ITERATION
4712 024422 000004              SCOPE
4713 024424 112737 000041 001202      MOV   #41,@$STSTM         ;LOAD TEST NUMBER
4714 024432 013737 001202 177570      MOV   @$STSTM,@$DISPLAY   ;;DISPLAY TEST NUMBER
4715
4716                      .SBTTL START OF SECTION 4
4717                      ;4444444444444444 FIRST ADDRESS TO BE RELOCATED 4444444444
4718 024440 010700              REL4: MOV   PC,R0           ;GET PC
4719 024442 005740              TST   -(R0)              ;R0 CONTAINS THE ADDRESS OF REL4
4720 024444 010037 001540              MOV   R0,@$FRSTAD        ;SAVE
4721 024450 010700              MOV   PC,R0           ;GET CURRENT PC
4722 024452 162700 024452              SUB   #.,R0             ;SUBTRACT RELOCATION FACTOR
4723 024456 010037 001534              MOV   R0,@$FACTOR        ;SAVE RELOCATION FACTOR
4724 024462 010737 001212              MOV   PC,@$SLPERR        ;SET LOOP ADDRESS
4725 024466 062737 000030 001212      ADD   #30,@$SLPERR       ;ADJUST
4726 024474 013737 001212 001210      MOV   @$SLPERR,@$SLPADR
4727 024502 105737 001530              TSTB  @NEXEC            ;BR IF TEST CODE TO BE EXECUTED
4728 024506 001402                      BEQ   .+6
4729 024510 000167 001512              JMP   RELE4
4730
4731 024514 013767 177776 000334      OVFLW: MOV   @PSW,$S       ;SAVE STATUS IN 7$ BELOW
4732 024522 005037 177776              CLR   @PSW              ;SET KERNEL MODE
4733 024526 004737 063162              JSR   PC,@CLRTBIT        ;GO CLEAR 'T' BIT IF SET
4734 024532 052737 000340 177776      BIS   #PR7,@PSW         ;SET PRIORITY LEVEL 7 TO BLOCK CLOCK
4735 024540 010746              MOV   PC,-(SP)          ;PUSH CURRENT PC ONTO STACK
```


4736	024542	062716	000152		ADD	#2\$-,(SP)	:FORM ADDRESS OF 2\$ BELOW
4737	024546	011637	000004		MOV	(SP),@#ERRVEC	:SET ERROR VECTOR
4738	024552	012737	000340	000006	MOV	#340,@#ERRVEC+2	:SET PRIORITY LEVEL 7 ON TRAP
4739	024560	013727	000014		MOV	@#BPTVEC,(PC)+	:SAVE BPT VECTOR ADRS
4740	024564	000000			.WORD	0	
4741	024566	062716	000100		ADD	#41\$-2\$,(SP)	:FORM ADDRESS OF 41\$ BELOW
4742	024572	012637	000014		MOV	(SP)+,@#BPTVEC	:SET BPT TRAP VECTOR TO 41\$
4743	024576	012737	000340	000016	MOV	#340,@#BPTVEC+2	
4744							
4745	024604	012703	000376		MOV	#376,R3	
4746	024610	010313			MOV	R3,(R3)	:LOAD 376 INTO ADDRESS 376
4747	024612	010306			MOV	R3,SP	:SET STACK PTR AT BOUNDARY
4748	024614	032767	140000	000234	BIT	#UM,7\$:CHECK IF ENTERED TEST IN KERNEL
4749	024622	001015			BNE	1\$:MODE. BRANCH IF NOT IN KERNEL
4750							
4751							:THE BELOW INSTRUCTIONS SHOULD NOT CAUSE AN OVERFLOW TRAP
4752	024624	005716			TST	(SP)	:BECAUSE TST IS A NON MODIFYING INST
4753	024626	021666	177776		CMF	(SP),-2(SP)	:SO IS COMPARE
4754	024632	012656			MOV	(SP)+,@-(SP)	:BECAUSE OF ADDRESS MODE 5
4755	024634	057636	000000		BIS	@(SP),@(SP)+	:BECAUSE OF ADDRESS MODE 3
4756	024640	054676	000000		BIS	-(SP),@(SP)	:BECAUSE OF ADDRESS MODE 7
4757	024644	005006			CLR	SP	
4758	024646	013766	020000	020000	MOV	@#20000,20000(SP)	
4759	024654	000425			BR	3\$:BRANCH OVER NON KERNEL MODE TESTS
4760							
4761							:NOTE: NO OVEFLOW TRAP WILL OCCUR IF NOT IN KERNEL MODE!...
4762	024656	156737	000175	177777	BISB	7\$+1,@#PSW+1	:RESTORE MODE BITS IN PSW
4763	024664	012706	000376		MOV	#376,SP	:SET STACK PTR
4764	024670	016646	177776		MOV	-2(SP),-(SP)	:SHOULD NOT TRAP
4765	024674	051616			BIS	(SP),(SP)	
4766	024676	061666	177776		ADD	(SP),-2(SP)	
4767	024702	105037	177777		CLRB	@#PSW+1	:SET KERNEL MODE
4768	024706	012706	000700		MOV	#SUPSTK,SP	:RESTORE THE STACK
4769	024712	000451			BR	6\$:EXIT TEST
4770							
4771							:ERROR SERVICE ROUTINE
4772	024714	012600			MOV	(SP)+,R0	:SAVE PC OF INSTRUCTION THAT TRAPPED
4773	024716	012602			MOV	(SP)+,R2	:SAVE PSW
4774	024720	012706	000700		MOV	#SUPSTK,SP	:SET STACK PTR
4775	024724	104000			HLT		:ERROR! AN INSTRUCTION THAT WAS NOT
4776							:SUPPOSED TO TRAP TRAPPED
4777							:R0 CONTAINS PC, R2 CONTAINS PSW
4778	024726	000443			BR	6\$:EXIT TEST
4779							:THE BELOW INSTRUCTIONS WILL CAUSE A STACK OVERFLOW
4780							:STACK PTR IS AT 376
4781	024730	062737	000066	000004	ADD	#4\$-2\$,@#ERRVEC	:SET ERROR VECTOR TO 4\$
4782	024736	010306			MOV	R3,SP	:SET STACK PTR AT 376
4783	024740	112702	000001		MOVB	#1,R2	
4784	024744	005000			CLR	R0	
4785	024746	005016			CLR	(SP)	:SETS BIT 0 IN R0
4786	024750	006302			ASL	R2	:SHIFT INDICATOR BIT
4787	024752	105226			INCB	(SP)+	:SETS BIT 1 IN R0
4788	024754	006302			ASL	R2	
4789	024756	060746			ADD	PC,-(SP)	:SETS BIT 2 IN R0
4790	024760	006302			ASL	R2	
4791	024762	000003			BPT		:SETS BIT 3 IN R0

```
4792 024764 006302      ASL      R2
4793 024766 004767 000014  JSR      PC,40$      ;SETS BIT 4 IN R0
4794 024772 006302      ASL      R2
4795 024774 050666 177776  BIS     SP,-2(SP)    ;SETS BIT 5 IN R0
4796 025000 000410      BR       5$
4797
4798 ;PROGRAM WILL TRAP HERE ON OVERFLOW TRAP
4799 025002 050200 4$:  BIS     R2,R0      ;SET APPROPRIATE BIT IN R0
4800 025004 000002      RTI      ;RETURN FROM TRAP
4801
4802 025006 052700 001000 40$:  BIS     #1000,R0      ;SET IND THAT JSR WAS EXECUTED
4803 025012 000207      RTS     PC
4804
4805 025014 052700 000400 41$:  BIS     #400,R0      ;SET IND THAT BPT WAS EXECUTED
4806 025020 000002      RTI
4807
4808 ;CHECK THAT ABOVE INSTRUCTIONS DID TRAP
4809 025022 012706 000700 5$:  MOV     #SUPSTK,SP    ;SET STACK PTR
4810 025026 022700 001477      CMP     #1477,R0      ;EACH INSTRUCTION SET A BIT IN R0
4811 025032 001401      BEQ     .+4           ;R0= 1477
4812 025034 104000      HLT
4813
4814 ;EXIT ROUTINE
4815 025036 012706 001200 6$:  MOV     #KERSTK,SP    ;SET KERNEL STACK PTR
4816 025042 016737 177516 000014  MOV     43$,@BPTVEC    ;RESTORE BPT VECTOR
4817 025050 005037 000016      CLP     @BPTVEC+2
4818 025054 012746      MOV     (PC)+,-(SP)    ;PUSH OLD PSW ONTO STACK
4819 025056 000000 7$:  .WORD   0             ;CONTAINS SAVED PSW
4820 025060 010746      MOV     PC,-(SP)      ;PUSH CURRENT PC ONTO STACK
4821 025062 062716 000006      ADD     #6,(SP)       ;ADD OFFSET
4822 025066 000002      RTI
4823 025070 012706 000700      MOV     #SUPSTK,SP    ;SET STACK PTR
4824 025074 012737 064270 000004  MOV     #ERPRT,@ERRVEC ;RESET TIME OUT VECTOR
4825 025102 013737 177776 000006  MOV     @PSW,@ERRVEC+2
4826 025110 052737 000340 000006  BIS     #PR7,@ERRVEC+2
4827 025116 042737 000020 000006  BIC     #BIT4,@ERRVEC+2
4828 025124 005037 177766      CLR     @CPUERR
4829
4830 ;*****
4831 ;*TEST 42 CHECK THAT ALL RESERVED INSTRUCTIONS TRAP
4832 ;*****
4832 025130 000004 TST42: SCOPE
4833 025132 112737 000042 001202  MOV     #42,@$TSTNM    ;LOAD TEST NUMBER
4834 025140 013737 001202 177570  MOV     @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
4835 025146 005737 001502  RESTRP: TST     @KB11E    ;IS THIS A KB11-E OR KB11-EM?
4836 025152 001403      BEQ     10$           ;BR IF NOT
4837 025154 012767 000010 000122  MOV     #10,5$        ;KB11-E AND KB11-EM USES OPCODE 7, START WITH OPCODE 10
4838 025162 012702 025304 10$:  MOV     #5$,R2        ;GET ADDRESS OF RESERVED INSTRUCTION TABLE
4839 025166 105737 001505      TSTB   @CISP         ;IS CISP OPTION PRESENT?
4840 025172 001402      BEQ     8$           ;BR IF NOT
4841 025174 012702 025342      MOV     #6$,R2        ;ADDRESS OF RESERVED INSTRUCTION TABLE WITH CIS
4842 025200 063702 001534 8$:  ADD     @FACTOR,R2
4843 025204 132737 000040 001501  BITB   #40,@$OPT.CP+1 ;CHECK IF 11/45 FLOATING POINT IS AVAIL.
4844 025212 001404      BEQ     9$           ;BRANCH IF NOT AVAILABLE
4845 025214 005067 000212      CLR     51$          ;SET CIS TABLE TERMINATOR AT GROUP 7
4846 025220 005067 000110      CLR     50$          ;SET TABLE TERMINATOR AT GROUP 7
4847 025224 012737 025262 000010 9$:  MOV     #4$,@$RESVEC  ;SET RESERVED INSTRUCTION TRAP
```

```
4848 025232 063737 001534 000010      ADD    @#FACTOR,@#RESVEC
4849 025240 012203      1$:   MOV    (R2)+,R3      ;GET FIRST RESERVED INSTRUCTION
4850 025242 001476      BEQ    7$            ;0 TERMINATES THE TABLE
4851 025244 012204      MOV    (R2)+,R4      ;GET LAST RESERVED INSTRUCTION IN GROUP
4852 025246 010317      2$:   MOV    R3,(PC)    ;EXECUTE RESERVED INSTRUCTION
4853 025250 000000      3$:   .WORD  0         ;CONTAINS RESERVED INSTRUCTION
4854 025252 000240      NOP                    ;ERROR! INSTRUCTION IN R3
4855 025254 000240      NOP                    ;(2$) ABOVE FAILED TO CAUSE A
4856 025256 104000      HLT                    ;RESERVED INSTRUCTION TRAP
4857 025260 000405      BR     41$
4858 025262 012716 025274 4$:   MOV    #41$, (SP)    ;ADJUST RETURN PC
4859 025266 063716 001534  ADD    @#FACTOR,(SP) ;TO RETURN TO 41$
4860 025272 000002      RTI                    ;RETURN TO 41$
4861 025274 020304      41$:  CMP    R3,R4         ;HAS GROUP OF RESERVED INSTRUCTIONS
4862 025276 001760      BEQ    1$            ;BEEN EXECUTED
4863 025300 005203      INC    R3            ;INCREMENT THIS RESERVED INSTRUCTION
4864 025302 000761      BR     2$            ;TO NEXT ONE AND EXECUTE
4865      ;TABLE OF 1170 RESERVED INSTRUCTIONS (0 TERMINATES THE TABLE)
4866 025304 000007      5$:   7              ;GROUP 1 (GETS A 10 IF KB11-E OR KB11-EM)
4867 025306 000077      77
4868 025310 000210      210          ;GROUP 2
4869 025312 000227      227
4870 025314 007000      7000        ;GROUP 3
4871 025316 007777      7777
4872 025320 075040      75040       ;GROUP 4
4873 025322 076777      76777
4874 025324 106400      106400      ;GROUP 5
4875 025326 106477      106477
4876 025330 106700      106700      ;GROUP 6
4877 025332 107777      107777
4878 025334 170000      50$:  170000      ;GROUP 7      FLOATING POINT
4879 025336 177777      177777      INSTRUCTIONS
4880 025340 000000      C            ;0 TERMINATES THE TABLE
4881
4882
4883      ;TABLE OF KB11-E/EM WITH CIS RESERVED INSTRUCTIONS (0 TERMINATES THE TABLE)
4884 025342 000010      6$:   10          ;GROUP 1
4885 025344 000077      77
4886 025346 000210      210        ;GROUP 2
4887 025350 000227      227
4888 025352 007000      7000       ;GROUP 3
4889 025354 007777      7777
4890 025356 075040      75040      ;GROUP 4A
4891 025360 076017      76017
4892 025362 076033      76033      ;GROUP 4B
4893 025364 076037      76037
4894 025366 076046      76046      ;GROUP 4C
```

4895	025370	076047			76047					
4896	025372	076100			76100					
4897	025374	076127			76127					
4898	025376	076133			76133					
4899	025400	076137			76137					
4900	025402	076146			76146					
4901	025404	076147			76147					
4902	025406	076160			76160					
4903	025410	076167			76167					
4904	025412	076200			76200					
4905	025414	076600			76600					
4906	025416	076602			76602					
4907	025420	076777			76777					
4908	025422	106400			106400					
4909	025424	106477			106477					
4910	025426	106700			106700					
4911	025430	107777			107777					
4912	025432	170000			170000					
4913	025434	177777			177777					
4914	025436	000000			0					
4915										
4916	025440	012737	064216	000010	7\$: MOV	#RESERR,@#RESVEC				;RESTORE RESERVED TRAP
4917										
4918										
4919										
4920	025446	000004			TST43:	SCOPE				
4921	025450	112737	000043	001202	MOVB	#43,@#TSTNM				;LOAD TEST NUMBER
4922	025456	013737	001202	177570	MOV	@#TSTNM,@#DISPLAY				;:DISPLAY TEST NUMBER
4923	025464	105737	001531		PSWCHK:	TSTB	@#MMON			;IF MEM MGMT IS ON SKIP THIS TEST
4924	025470	001070			BNE	4\$				
4925	025472	013767	177776	000140	MOV	@#PSW,3\$;SAVE STATUS
4926	025500	005037	177776		CLR	@#PSW				;CLEAR MODE BITS IN PSW
4927	025504	004737	063162		JSR	PC,@#CLRTRBIT				;GO CLEAR 'T' BIT IF SET
4928	025510	013746	000016		MOV	@#TBITVEC+2,-(SP)				
4929	025514	012704	177776		MOV	#PSW,R4				;LOAD ADDRESS OF PSW INTO R4
4930	025520	000250			CLN					
4931	025522	005714			TST	(R4)				;CHECK THAT PSW WAS CLEARED
4932	025524	001401			BEQ	+.4				
4933	025526	104000			HLT					;ERROR! PSW FAILED TO CLEAR
4934	025530	012700	170357		MOV	#170357,R0				
4935	025534	005737	001502		TST	@#KB11E				;IS THIS A KB11-E OR KB11-EM PROCESSOR?
4936	025540	001402			BEQ	10\$;BR IF NOT
4937	025542	052700	000400		BIS	#400,R0				;ALSO TEST PS08 IF KB11-E
4938	025546	012702	000001		10\$: MOV	#1,R2				;R2 = TEST BIT
4939	025552	030200			1\$: BIT	R2,R0				;CHECK IF BIT CAN BE SET/CLEARED
4940	025554	001423			BEQ	2\$				
4941	025556	005037	000016		CLR	@#TBITVEC+2				
4942	025562	030227	000020		BIT	R2,#20				;CHECK IF TEST WILL SET 'T' BIT
4943	025566	001403			BEQ	20\$				
4944	025570	012737	000002	000016	20\$: MOV	#RTI,@#TBITVEC+2				;SET RTI INTO RETURN
4945	025576	005014			CLR	(R4)				;CLEAR PSW
4946	025600	050214			BIS	R2,(R4)				;SET R2 INTO PSW
4947	025602	011403			MOV	(R4),R3				;GET BIT
4948	025604	020203			CMP	R2,R3				;CHECK THAT BIT WAS SET IN PSW
4949	025606	001401			BEQ	+.4				
4950	025610	104000			HLT					;ERROR! BIT IN R2 FAILED TO SET IN PSW

```

4951 025612 000244          CLZ          ;CLEAR Z BIT
4952 025614 040214          BIC          R2,(R4)      ;CLEAR BIT IN PSW
4953 025616 011403          MOV          (R4),R3     ;GET PSW RESULT
4954 025620 001401          BEQ          2$          ;BRANCH IF BIC ABOVE CLEARED BIT IN PSW
4955 025622 104000          HLT          ;ERROR! BIT IN R2 FAILED TO CLEAR IN PSW
4956 025624 006302          2$: ASL          R2          ;SHIFT TEST BIT
4957 025626 103351          BCC          1$          ;BRANCH IF ALL BITS NOT TESTED
4958 025630 005014          CLR          (R4)        ;CLEAR STATUS
4959 025632 012637 000016    MOV          (SP)+,@#TBITVEC+2 ;RESTORE T BIT RETURN
4960 025636 012746          MOV          (PC)+,-(SP) ;PUSH ORIGINAL STATUS ON STACK
4961 025640 000000          3$: .WORD          0          ;CONTAINS ORIGINAL PSW
4962 025642 010746          MOV          PC,-(SP)    ;SET RETURN PC
4963 025644 062716 000006    ADD          #6,(SP)
4964 025650 000002          RTI          ;RETURN
4965 025652 013704 177776    4$: MOV          @#PSW,R4    ;SAVE PSW IN R4
4966 025656 112737 000340 177776  MOVB         #340,@#PSW    ;SET PRIORITY LEVEL 7
4967 025664 004737 065162    JSR          PC,@#CLRTRBIT ;GO CLEAR 'T' BIT IF SET
4968
4969          ;*****
4970          ;*TEST 44 CHECK THAT ALL BITS IN THE CURRENT STACK PTR CAN BE SET CLEARED
4971          ;*****
4971 025670 000004          TST44: SCOPE
4972 025672 112737 000044 001202  MOVB         #44,@#TSTNM    ;LOAD TEST NUMBER
4973 025700 013737 001202 177570  MOV          @#TSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
4974 025706 010603          CHKSP: MOV          SP,R3    ;SAVE STACK PTR
4975 025710 000257          CCC
4976 025712 112706 000377    MOVB         #377,SP       ;SET STACK PTR = -1
4977 025716 006006          1$: ROR          SP          ;ROTATE 0 BIT THROUGH ALL BIT
4978 025720 103776          BCS          1$          ;BIT POSITIONS
4979 025722 005206          INC          SP          ;SHOULD INCREMENT SP TO 0
4980 025724 001403          BEQ          2$
4981 025726 010602          MOV          SP,R2       ;SAVE ERROR STACK PTR
4982 025730 010306          MOV          R3,SP       ;SET STACK PTR FOR TRAP
4983 025732 104000          HLT          ;ERROR.
4984
4985 025734 010306          2$: MOV          R3,SP       ;RESTORE ORIGINAL STACK PTR
4986
4987          ;CHECK BYTE OPERATIONS USING THE STACK
4988 025736 010600          SPCHK: MOV          SP,R0    ;SAVE STACK PTR
4989 025740 010003          MOV          R0,R3
4990
4991 025742 005043          CLR          -(R3)
4992 025744 112746 177777    MOVB         #-1,-(SP)    ;(SP) = 377
4993 025750 022713 000377    CMP          #377,(R3)    ;CHECK THAT ONLY EVEN BYTE WAS AFFECTED
4994 025754 001002          BNE          1$
4995 025756 020306          CMP          R3,SP       ;CHECK AUTO-DEC
4996 025760 001401          BEQ          .+4
4997 025762 104000          1$: HLT
4998
4999 025764 105226          INCB         (SP)+
5000 025766 005723          TST          (R3)+       ;CHECK RESULT
5001 025770 001002          BNE          2$
5002 025772 020006          CMP          R0,SP       ;CHECK AUTO-INC
5003 025774 001401          BEQ          .+4
5004 025776 104000          2$: HLT
5005
5006 026000 005143          COM          -(R3)       ;(R3)=177777

```

```
5007 026002 144613          BICB    -(SP), (R3)
5008 026004 022713 177400    CMP     #177400, (R3) ;CHECK RESULT
5009 026010 001002          BNE     3$
5010 026012 020603          CMP     SP, R3
5011 026014 001401          BEQ     .+4
5012 026016 104000          3$:    HLT
5013
5014 026020 132627 000377          BITB    (SP)+, #377
5015 026024 001002          BNE     4$
5016 026026 020600          CMP     SP, R0
5017 026030 001401          BEQ     .+4
5018 026032 104000          4$:    HLT
5019
5020 026034 012746 000001          MOV     #1, -(SP)
5021 026040 062706 000002          ADD     #2, SP
5022 026044 012702 177401          MOV     #177401, R2
5023 026050 120246          CMPB   R2, -(SP)
5024 026052 001004          BNE     5$
5025 026054 122602          CMPB   (SP)+, R2
5026 026056 001002          BNE     5$
5027 026060 020006          CMP     R0, SP
5028 026062 001401          BEQ     .+4
5029 026064 104000          5$:    HLT
5030 026066 105037 177776          CLRB   @PSW
5031 026072 010446          MOV     R4, -(SP) ;RESTORE ORIGINAL PSW TO STACK
5032 026074 010746          MOV     PC, -(SP)
5033 026076 062716 000006          ADD     #6, (SP)
5034 026102 000002          RTI
5035
5036          ;*****
5037          ;*TEST 45 CHECK THAT 'C' BIT SETS/CLEARs PROPERLY
5038          ;*****
5038 026104 000004          TST45: SCOPE
5039 026106 112737 000045 001202          MOVB   #45, @STSTNM ;LOAD TEST NUMBER
5040 026114 013737 001202 177570          MOV    @STSTNM, @DISPLAY ;:DISPLAY TEST NUMBER
5041 026122 012727 177776          (BIT:  MOV    #177776, (PC)+ ;LOAD CONSTANT
5042 026126 000000          1$:    .WORD  0
5043 026130 010700          MOV    PC, R0 ;GET CURRENT PC
5044 026132 162700 000004          SUB    #4, R0 ;POINT R0 TO 1$ ABOVE
5045 026136 005520          2$:    ADC    (R0)+ ;ADD 'C' BIT TO 1$ ABOVE
5046 026140 006340          ASL    -(R0) ;SHIFT 1$
5047 026142 102375          BVC    2$ ;UNTIL 'V' BIT SETS
5048 026144 022767 077776 177754          CMP    #077776, 1$ ;CHECK RESULT
5049 026152 001401          BEQ    .+4
5050 026154 104000          HLT ;ERROR! INCORRECT RESULT IN 1$ ABOVE
5051          ;R0=ADDRESS OF DATA
5052
5053          ;CHECK THAT CONDITION CODES ARE SET PROPERLY WHEN A NUMBER (CURRENT PC)
5054          ;AND THAT NUMBER +1 ARE COMPARED, AND VICE VERSA.
5055 026156 010700          (MPNUM: MOV    PC, R0 ;GET CURRENT PC
5056 026160 010002          MOV    R0, R2 ;SAVE IN R2
5057 026162 005202          INC    R2 ;MAKE R2 = R0+1
5058 026164 000277          SCC
5059 026166 000251          +CLC!CLN ;CLEAR C & N BITS
5060 026170 020002          CMP    R0, R2 ;COMPARE # WITH #+1
5061 026172 103003          BCC    1$ ;CARRY BIT SHOULD SET
5062 026174 102402          BVS    1$ ;V BIT SHOULD CLEAR
```

```
5063 026176 001401          BEQ      1$          ;Z BIT SHOULD CLEAR
5064 026200 100401          BMI      .+4         ;N BIT SHOULD SET
5065 026202 104000          1$:      HLT          ;ERROR! COMPARE # WITH #+1 FAILED TO
5066                                     ;SET CONDITION CODES IN PSW CORRECTLY
5067
5068 026204 000277          SCC          ;SET CONDITION CODES IN PSW
5069 026206 120200          CMPB     R2,R0       ;COMPARE #+1 WITH #
5070 026210 103403          BCS      2$          ;C BIT SHOULD CLEAR
5071 026212 102402          BVS      2$          ;V BIT SHOULD CLEAR
5072 026214 001401          BEQ      2$          ;Z BIT SHOULD CLEAR
5073 026216 100001          BPL      .+4         ;N BIT SHOULD CLEAR
5074 026220 104000          2$:      HLT          ;ERROR! COMPARE #+1 WITH # FAILED TO SET
5075                                     ;CONDITION CODES IN PSW CORRECTLY
5076 026222 105037 177776          CLRB     @#PSW       ;ENSURE PRIORITY 0
5077 026226 000004          RELE4:   SCOPE
5078 026230 010702          MOV      PC,R2
5079 026232 062702 000012          ADD      #12,R2
5080 026236 012707 043764          MOV      #RELOC,PC  ;GO RELOCATE PROGRAM CODE
5081 026242 000000          REL44:  .WORD      0
5082                                     ;44444444444444 LAST ADDRESS OF CODE TO BE RELOCATED 444444444444
5083
5084
5085
5086
5087 026244 012767 000001 153054          *TEST 46          CHECK EXTENDED INSTRUCTION SET
5088 026252 000004          ;*****
5089 026254 112737 000046 001202          ;*****
5090 026262 013737 001202 177570          TST46:  MOV      #1,$TIMES          ;;DO 1 ITERATION
5091                                     ;LOAD TEST NUMBER
5092                                     ;:DISPLAY TEST NUMBER
5093                                     ;*****
5094                                     ;*****
5095                                     ;*****
5096                                     ;*****
5097                                     ;*****
5098                                     ;*****
5099                                     ;*****
5100                                     ;*****
5101                                     ;*****
5102                                     ;*****
5103                                     ;*****
5104                                     ;*****
5105                                     ;*****
5106                                     ;*****
5107                                     ;*****
5108                                     ;*****
5109                                     ;*****
5110                                     ;*****
5111                                     ;*****
5112                                     ;*****
5113                                     ;*****
5114                                     ;*****
5115                                     ;*****
5116                                     ;*****
5117                                     ;*****
5118                                     ;*****

;SBTTL START OF SECTION 5
;5555555555555555 FIRST ADDRESS TO BE RELOCATED 5555555555
REL5:  MOV      PC,R0          ;GET PC
      TST      -(R0)         ;R0 CONTAINS THE ADDRESS OF REL5
      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      RELE5
EXTINST: CLR      R0
      SCC          ;PRESET CC'S
      SXT      R0          ;EXTEND SIGN (1) INTO R0
      BCC     SXT0        ;CHECK RESULT CC'S
      BVS     SXT0
      BEQ     SXT0
      BPL     SXT0
      INC     R0          ;CHECK RESULT
      BEQ     .+4
SXT0:  HLT
      MOV     PC,R0
      MOV     R0,R2
```

```

5119 026374 012703 177777      MOV    #-1,R3
5120 026400 005102      COM    R2
5121 026402 000243      +CLV:CLC      ;CLEAR C AND V BITS
5122 026404 074003      XOR    R0,R3      ;R3 SHOULD CONTAIN COMPLEMENT OF R0
5123 026406 103404      BCS    XOR0      ;CHECK THAT C WAS NOT AFFECTED
5124 026410 102403      BVS    XOR0      ;AND THAT V WAS CLEARED
5125 026412 001402      BEQ    XOR0
5126 026414 020203      CMP    R2,R3      ;CHECK RESULT
5127 026416 001401      BEQ    .+4
5128 026420 104000      XOR0:  HLT      ;ERROR.  XOR FAILED
5129
5130 026422 010700      MOV    PC,R0
5131 026424 022020      CMP    (R0)+,(R0)+ ;SET ADDRESS REGISTER
5132 026426 000401      BR     1$        ;RESERVE WORD FOR TEST DATA
5133 026430 000000      .WORD 0         ;CONTAINS TEST DATA
5134 026432 005700      1$:  TST    R0      ;EXTEND SIGN OF ADDRESS INTO
5135 026434 006710      SXT    (R0)      ;ADDRESS (R0)=-1 IF MSB R0-1
5136 026436 005002      CLR    R2        ;OTHERWISE, (R0)=0
5137 026440 005700      TST    R0        ;CHECK SIGN OF ADDRESS
5138 026442 100001      BPL    .+4
5139 026444 005102      COM    R2        ;COMPLEMENT CHECK REG IF NEG
5140 026446 021002      CMP    (R0),R2   ;CHECK RESULT OF SXT
5141 026450 001401      BEQ    .+4
5142 026452 104000      SXT1:  HLT      ;ERROR!  SXT FAILED TO EXTEND SIGN PROPERLY
5143
5144 026454 012710 100000      MOV    #100000,(R0) ;PRESET DATA
5145 026460 011002      MOV    (R0),R2
5146 026462 000277      SCC    ;PRESET CC'S
5147 026464 074210      XOR    R2,(R0)   ;XOR 100000 WITH 100000 RESULT = 0
5148 026466 103007      BCC    XOR1      ;CHECK CC'S AFTER XOR
5149 026470 102406      BVS    XOR1
5150 026472 001005      BNE    XOR1
5151 026474 100404      BMI    XOR1
5152 026476 005710      TST    (R0)      ;CHECK RESULT (0)
5153 026500 001002      BNE    XOR1
5154 026502 005402      NEG    R2        ;CHECK THAT REG WAS NOT AFFECTED
5155 026504 102401      BVS    .+4
5156 026506 104000      XOR1:  HLT
5157
5158 026510 010702      MOV    PC,R2
5159 026512 022222      CMP    (R2)+,(R2)+ ;PRESERVE WORD FOR DATA
5160 026514 000401      BR     SXT4      ;RESERVED FOR DATA
5161 026516 000000      .WORD 0         ;PRESET DATA
5162 026520 012722 125252      SXT4:  MOV    #125252,(R2)+ ;EXTEND SIGN
5163 026524 006742      SXT    -(R2)
5164 026526 074722      XOR    PC,(R2)+
5165 026530 010700      MOV    PC,R0      ;GET PC
5166 026532 005740      TST    -(R0)     ;SUBTRACT 2 FROM PC
5167 026534 005100      COM    R0        ;R0=RESULT OF XOR PC-1 ABOVE
5168 026536 074042      XOR    R0,-(R2)  ;CHECK RESULT OF SXT AND XOR ABOVE
5169 026540 001401      BEQ    .+4
5170 026542 104000      XOR24: HLT      ;ERROR!  SXT & XOR ABOVE INCORRECT
5171
5172 026544 012704 000001      MOV    #1,R4      ;SET R4
5173 026550 006767 000060      SXT    XOR6A     ;PRESET DATA 0
5174 026554 074467 000054      2$:  XOR    R4,XOR6A
    
```



```

5175 026560 100423          BMI    XOR6
5176 026562 006304          ASL   R4           ;SHIFT R4
5177 026564 102373          BVC  2$           ;UNTIL V SETS (R4=100000)
5178 026566 100020          BPL  XOR6         ;BRANCH IF 'N' IS CLEAR
5179 026570 074467 000040      XOR   R4,XOR6A    ;XOR6A=177777
5180 026574 100015          BPL  XOR6
5181 026576 074767 000032      XOR   PC,XOR6A    ;XOR PC WITH XOR6A (177777)
5182 026602 010767 000030      MOV   PC,XOR6B    ;FORM PC AS USED IN XOR ABOVE
5183 026606 162767 000004 000022  SUB   #4,XOR6B
5184 026614 005167 000016      COM  XOR6B
5185 026620 026767 000012 000006  CMP   XOR6B,XOR6A ;XOR6A SHOULD = COMPLEMENT OF PC
5186 026626 001401          BEQ  .+4
5187 026630 104000          XOR6: HLT         ;ERROR. XOR TESTS ABOVE FAILED
5188
5189 026632 000402          BR   .+6
5190
5191 026634 000000          XOR6A: .WORD 0    ;CONTAINS DATA USED BY TEST ABOVE
5192 026636 000000          XOR6B: .WORD 0
5193
5194
5195 026640 012700 077777      MOV   #077777,R0 ;SET SOURCE OPERAND FOR ADD
5196 026644 006767 177764      SXT  XOR6A        ;CLEAR XOR6A
5197 026650 001004          BNE  SXT6         ;CHECK CC'S AFTER EXTENDING ZERO'S
5198 026652 100403          BMI  SXT6
5199 026654 103402          BCS  SXT6
5200 026656 102401          BVS  SXT6
5201 026660 000401          BR   .+4
5202 026662 104000          SXT6: HLT        ;ERROR! SXT FAILED
5203
5204 026664 012702 000001      MOV   #1,R2       ;SET DEST OPERAND FOR ADD
5205 026670 013703 001534      MOV   @#FACTOR,R3 ;LOAD INDEX REGISTER
5206 026674 060002          ADD  R0,R2        ;RESULT OF ADD=100000
5207 026676 006763 026634      SXT  XOR6A(3)     ;EXTEND SIGN OF ADD ABOVE
5208 026702 001403          BEQ  SXT6A
5209 026704 005267 177724      INC  XOR6A
5210 026710 001401          BEQ  .+4
5211 026712 104000          SXT6A: HLT       ;ERROR! SXT ABOVE FAILED TO EXTEND
5212
5213 026714 010703          MOV   PC,R3
5214 026716 000402          BR   .+6         ;PRESERVE 2 WORDS FOR DATA
5215 026720 000000          SXRA: .WORD 0    ;RESERVED WORD FOR DATA
5216 026722 000000          SXRB: .WORD 0    ;RESERVED WORD FOR DATA
5217 026724 005723          TST  (R3)+
5218 026726 010304          MOV   R3,R4      ;R3 = ADDRESS OF SXRA
5219 026730 000250          CLN  ;CLEAR N BIT
5220 026732 006724          SXT  (R4)+       ;EXTEND ZEROS INTO SXRA
5221 026734 001401          BEQ  .+4
5222 026736 104000          SXT2: HLT       ;ERROR! SXT FAILED
5223
5224 026740 010467 177754      MOV   R4,SXRA    ;SXRA = ADDRESS OF SXRB
5225 026744 000257          CCC  ;CLEAR CONDITION CODES
5226 026746 006733          SXT  @ (R3)+     ;EXTEND ZEROS INTO SXRB
5227 026750 001401          BEQ  .+4
5228 026752 104000          SXT3: HLT       ;ERROR!
5229
5230 026754 000270          SEN  ;SET N BIT

```

```

5231 026756 006753          SXT    @-(R3)          ;EXTEND ONES INTO SXR8
5232 026760 100401          BMI    .+4              ;
5233 026762 104000          SXT5: HLT              ;ERROR.
5234
5235 026764 012704 025252          MOV    #025252,R4       ;R4 = 025252
5236 026770 074433          XOR    R4,@(R3)+       ;SXR8 = 152525 (COMPLEMENT OF R4)
5237 026772 005002          CLR    R2              ;
5238 026774 074253          XOR    R2,@-(R3)       ;SXR8 REMAINS UNCHANGED
5239 026776 001405          BEQ    XOR35           ;CHECK CONDITION CODES
5240 027000 100004          BPL    XOR35           ;
5241 027002 005104          COM    R4              ;R4 - 152525
5242 027004 020467 177712          CMP    R4,SXR8        ;CHECK XOR
5243 027010 001401          BEQ    .+4              ;
5244 027012 104000          XOR35: HLT            ;ERROR: XOR FAILED
5245
5246 027014 005743          TST    -(R3)          ;R3 = ADDRESS OF SXRA-2
5247 027016 000250          CLN    .+4              ;CLEAR N BIT
5248 027020 006773 000002          SXT    @2(R3)         ;SXR8 = 0
5249 027024 001401          BEQ    .+4              ;
5250 027026 104000          SXT7: HLT            ;ERROR! SXT FAILED
5251
5252 027030 074473 000002          XOR    R4,@2(R3)       ;SXR8 = R4
5253 027034 020473 000002          CMP    R4,@2(R3)       ;CHECK XOR
5254 027040 001401          BEQ    .+4              ;
5255 027042 104000          XOR7:  HLT            ;ERROR! XOR FAILED
5256
5257
5258
5259
5260
5261 027044 000004          TST47: SCOPE
5262 027046 112737 000047 001202          MOV    #47,@$TSTNM     ;LOAD TEST NUMBER
5263 027054 013737 001202 177570          MOV    @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
5264
5265 027062 005005          CLR    R5              ;CLEAR ERROR INDICATOR
5266 027064 000407          BR    SOB0             ;BRANCH TO SOB TEST
5267
5268 027066 005004          SOB10: CLR    R4         ;R4 = 0
5269 027070 005705          TST    R5              ;CHECK ERROR INDICATOR
5270 027072 001401          BEQ    .+4              ;SOB BRANCHED CORRECTLY
5271 027074 104000          HLT                    ;ERROR!
5272
5273 027076 005005          SOB9:  CLR    R5         ;CLEAR INDICATOR (R5)
5274 027100 006004          ROR    R4              ;ROTATE RIGHT R4
5275 027102 000467          BR    SOB8             ;
5276
5277 027104 012700 000010          SOB0:  MOV    #10,R0     ;R0=10
5278 027110 000277          SCC                    ;SET CONDITION CODES
5279 027112 001012          SOB1:  BNE    SOB2       ;CHECK CONDITION CODES AFTER SOB
5280 027114 100011          BPL    SOB2             ;SOB SHOULD NOT EFFECT THE
5281 027116 102010          BVC    SOB2             ;CONDITION CODES.
5282 027120 103007          BCC    SOB2             ;
5283 027122 077005          SOB    R0,SOB1         ;
5284 027124 001005          BNE    SOB2             ;CHECK CONDITION CODES AFTER
5285 027126 100004          BPL    SOB2             ;SOB FALLS THROUGH,
5286 027130 102003          BVC    SOB2             ;SOB SHOULD NOT EFFECT
    
```

```

5287 027132 103002          BCC SOB2          ;CONDITION CODES.
5288 027134 005700          TST R0            ;CHECK IF R0=0
5289 027136 001401          BEQ .+4
5290 027140 104000 SOB2: HLT          ;ERROR!
5291
5292 027142 012702 000100    MOV #100,R2       ;R2=100
5293 027146 012700 000101    MOV #101,R0       ;SET CHECK REGISTER, R0=101
5294 027152 001414 SOB3: BEQ SOB4     ;CHECK CONDITION CODES AFTER
5295 027154 100413          BMI SOB4          ;SOB BRANCH.
5296 027156 102412          BVS SOB4          ;SOB SHOULD NOT EFFECT
5297 027160 103411          BCS SOB4          ;CONDITION CODES.
5298 027162 005300          DEC R0            ;DECREMENT CHECK REGISTER
5299 027164 020002          CMP R0,R2         ;CHECK THAT SOB DECREMENTS
5300 027166 001006          BNE SOB4
5301 027170 000257          CCC
5302 027172 077211          SOB R2,SOB3       ;SET CONDITION CODES BEFORE SOB
5303 027174 001403          BEQ SOB4          ;BRANCH TO SOB3 UNTIL R2=0
5304 027176 100402          BMI SOB4          ;CHECK CONDITION CODES AFTER
5305 027200 005702          TST R2            ;SOB FALLS THROUGH
5306 027202 001401          BEQ .+4           ;CHECK IF R2=0
5307 027204 104000 SOB4: HLT          ;ERROR.
5308
5309 027206 012700 000001    SOB5: MOV #1,R0   ;R0=1
5310 027212 000401          BR .+4
5311 027214 104000          HLT
5312 027216 077002          SOB R0,-2         ;SOB SHOULD NOT BRANCH
5313
5314 027220 005700          TST R0            ;CHECK IF R0=0 AFTER SOB
5315 027222 001401          BEQ .+4
5316 027224 104000          HLT
5317
5318 027226 012704 100000    SOB5A: MOV #100000,R4 ;R4=100000
5319 027232 000403          BR $
5320 027234 005204 3$: INC R4 ;R4=100000
5321 027236 100403          BMI 2$           ;N BIT SHOULD BE SET
5322 027240 104000          HLT              ;ERROR! SOB DID NOT
5323
5324
5325 027242 077404 1$: SOB R4,3$ ;SOB SHOULD BRANCH
5326 027244 104000          HLT              ;ERROR! SOB DID NOT BRANCH
5327
5328 027246 012703 000100    SOB6: MOV #100,R3 ;R3=100
5329 027252 077301          SOB R3,SOB6      ;USE SOB TO BRANCH TO ITSELF
5330 027254 005703          TST R3           ;CHECK IF R3=0
5331 027256 001703          BEQ SOB10
5332 027260 104000 SOB7: HLT          ;ERROR!
5333
5334 027262 005705 SOB8: TST R5       ;CHECK INDICATOR (R5)
5335
5336
5337
5338
5339
5340 027264 001401          BEQ .+4           ;BRANCH IF SOB BRANCHES CORRECTLY
5341 027266 104000          HLT              ;ERROR!
5342

```

```
5343 027270 005205      INC      R5          ;SET INDICATOR (R5)
5344 027272 077477      SOB      R4,S0B9    ;TEST MAX. BRANCH OF SOB
5345 027274 005704      TST      R4          ;CHECK IF R4=0
5346 027276 001401      BEQ      .+4
5347 027300 104000      HLT                      ;ERROR!
5348
5349      ;*****
5350      ;*TEST 50      CHECK THE MARK INSTRUCTION
5351      ;*****
5351 027302 000004      TST50: SCOPE
5352 027304 112737 000050 001202      MOV      #50,@$STSTNM ;LOAD TEST NUMBER
5353 027312 013737 001202 177570      MOV      @$STSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
5354 027320 010602      MRKTST: MOV      SP,R2
5355 027322 010705      MOV      PC,R5          ;THE STACK LOOKS LIKE THIS AFTER
5356 027324 010500      MOV      R5,R0          ;THE JSR INSTRUCTION
5357 027326 010546      MOV      R5,-(SP)       ; -2(SP)= R0      THIS IS A
5358 027330 010746      MOV      PC,-(SP)       ; -4(SP)= PC      STRING
5359 027332 010746      MOV      PC,-(SP)       ; -6(SP)= PC+2    OF
5360 027334 010746      MOV      PC,-(SP)       ; -10(SP)= PC+4   FIVE
5361 027336 010746      MOV      PC,-(SP)       ; -12(SP)= PC+6   DUMMY
5362 027340 010746      MOV      PC,-(SP)       ; -14(SP)= PC+10  ARGUMENTS
5363 027342 012746 006405      MOV      #MARK+5,-(SP)  ; -16(SP)= MARK 5
5364 027346 010605      MOV      SP,R5          ; -20(SP) PC PUSHED BY .JSR
5365 027350 004767 000002      JSR      PC,MARK1
5366 027354 000403      BR      .+10
5367 027356 000205      MARK1: RTS      R5
5368 027360 104000      HLT                      ;ERROR! SHOULD BE DOING MARK 5 INST.
5369 027362 000407      BR      MARKEX
5370 027364 020602      CMP      SP,R2
5371 027366 001402      BEQ      .+6
5372 027370 104000      HLT                      ;ERROR! SP NOT RETURNED TO PROPER
5373 027372 000403      BR      MARKEX          ;VALUE BY MARK INSTRUCTION
5374 027374 020005      CMP      R0,R5
5375 027376 001401      BEQ      .+4
5376 027400 104000      HLT                      ;ERROR! DID NOT RESTORE R5 FROM STACK
5377 027402 010206      MARKEX: MOV     R2,SP   ;RESTORE SP
5378
5379      ;*****
5380      ;*TEST 51      RTT/RTI TEST
5381      ;*      RTT/RTI TEST INSURES THAT CP DOES THE INSTRUCTION FOLLOWING
5382      ;*      AN RTT IF THE 'T' BIT IS SET IN THE PSW,BUT DOES HONOR
5383      ;*      THE TRAP IMMEDIATELY IF IT EXECUTES AN RTI
5384      ;*      INSTRUCTION SEQUENCE-RTT
5385      ;*      2$:      RTT          ;NO 'T' TRAP AFTER RTT
5386      ;*              INC      R0          ;R0=000001
5387      ;*      5$:      COM      R0          ;'T' TRAP TO 5$ AFTER INC
5388      ;*              MOV      SAVPSW,2(SP) ;CLEAR 'T' BIT IN RETURN PSW
5389      ;*              RTI          ;RETURN TO INSTRUCTION FOLLOWING INC
5390      ;*              CMP      #RTT,2$    ;CHECK
5391      ;*              ETC
5392
5393      ;*      INSTRUCTION SEQUENCE-RTI
5394      ;*      2$:      RTI          ;'T' TRAP AFTER RTI
5395      ;*      5$:      COM      R0          ;R0=177777
5396      ;*              MOV      SAVPSW,2(SP) ;CLEAR 'T' BIT IN RETURN PSW
5397      ;*              RTI          ;RETURN TO INC INSTRUCTION
5398      ;*              INC      R0          ;R0=000000
```

```
5399          : *          CMP      #RTT,2$          :CHECK
5400          : *          ETC
5401          : *-----*
5402 027404 000004 T5T51: SCOPE
5403 027406 112737 000051 001202 MOV      #51,@$TSTNM          :LOAD TEST NUMBER
5404 027414 013737 001202 177570 MOV      @$TSTNM,@$DISPLAY    :DISPLAY TEST NUMBER
5405 027422 013767 177776 000214 RTT1: MOV      @$PSW,SAVPSW      :SAVE PSW
5406 027430 032767 000020 000206 BIT      #20,SAVPSW          :CHECK IF 'T' BIT SET
5407 027436 001402 BEQ      1$                  :CONTINUE IF NOT
5408 027440 000167 000402 JMP      RTT2EX              :BRANCH TO EXIT
5409 027444 010746 1$: MOV      PC,-(SP)          :GET CURRENT PC
5410 027446 062716 ADD      #5$-,(SP)          :FORM RELOCATED PC
5411 027452 012637 000014 MOV      (SP)+,@$TBITVEC      :LOAD INTO TRAP VECTOR
5412 027456 016746 000162 MOV      SAVPSW,-(SP)        :GET CURRENT PSW
5413 027462 011637 000016 MOV      (SP),@$TBITVEC+2
5414 027466 052737 000340 177776 BIS      #PR7,@$PSW          :SET PRIORITY LEVEL 7
5415 027474 005000 CLR      R0
5416 027476 052716 000360 BIS      #PR7+20,(SP)        :SET 'T' BIT IN PSW ON STACK
5417 027502 010746 MOV      PC,-(SP)          :PUT THE PC ON THE STACK
5418 027504 062716 000006 ADD      #6,(SP)            :ADJUST PC FOR NEXT INSTRUCTION
5419 027510 000006 2$: RTT
5420 027512 005200 INC      R0                  :DONE TO SEE IF INSTR. FOLLOWING
5421          :RTT IS EXECUTED IF T-BIT SET
5422 027514 042737 000340 177776 BIC      #PR7,@$PSW          :SET PRIORITY LEVEL 0
5423 027522 022767 000006 177760 CMP      #RTT,2$
5424 027530 001005 BNE      3$
5425 027532 022700 177776 CMP      #177776,R0          :CHECK IF INC WAS EXECUTED
5426 027536 001406 BEQ      4$                  :CHECK IF COM-R0 EXECUTED
5427 027540 104000 HLT
5428 027542 000415 BR      6$                  :ERROR!R0 NOT COMPLIMENTED
5429 027544 005700 3$: TST      R0              :EXIT TEST
5430          :TEST IF TRAPED BEFORE INC INST.
5431          :WAS EXECUTED
5432 027546 001413 BEQ      6$
5433 027550 104000 HLT
5434 027552 000411 BR      6$                  :ERROR!
5435 027554 012767 000002 177726 4$: MOV      #RTI,2$          :EXIT TEST
5436 027562 000730 BR      1$
5437 027564 005100 5$: COM      R0              :RTT CHECK
5438 027574 000002 MOV      SAVPSW,2(SP)
5439 027576 012767 000006 177704 6$: MOV      #RTT,2$
5440 027604 012737 001520 000014 MOV      #SRTN,@$TBITVEC      :RESTORE 'T' TRAP VECTOR
5441 027612 005037 000016 CLR      @$TBITVEC+2
5442 027616 042737 000360 000016 BIC      #PR7+BIT4,@$TBITVEC+2
5443 027624 RTT1EX:
5444          :*****
5445          :*TEST 52 SECOND RTT TEST
5446          :*****
5447 027624 000004 T5T52: SCOPE
5448 027626 112737 000052 001202 MOV      #52,@$TSTNM          :LOAD TEST NUMBER
5449 027634 013737 001202 177570 MOV      @$TSTNM,@$DISPLAY    :DISPLAY TEST NUMBER
5450 027642 000401 BR      RTT2A
5451 027644 000000 SAVPSW: .WORD 0
5452 027646 016700 177772 RTT2A: MOV      SAVPSW,R0      :GET SAVED PSW
5453 027652 105000 CLRB    R0                  :CLEAR PRIORITY LEVEL,T, AND COND CODES
5454 027654 012702 144000 MOV      #UM+REG,R2
```

```
5455 027660 074002      XOR    R0,R2
5456 027662 001435      BEQ    2$                ;USER MODE REG. SET #1 ON
5457 027664 012702 044000  MOV    #SM+REG,R2
5458 027670 074002      XOR    R0,R2
5459 027672 001447      BEQ    3$                ;SUPER MODE REG. SET #1 ON
5460 027674 032700 140000  BIT    #UM,R0
5461 027700 001062      BNE    RTT2EX
5462
5463                      ;TEST THAT RTT CLEARS BITS 11,12,13 & PRIORITY LEVEL BITS IN KERNEL MODE
5464 027702 012702 177777  MOV    #-1,R2            ;KERNEL MODE REG. SET 0 ON
5465 027706 012737 034240 177776  MOV    #PUM+REG+PR5,@PSW ;SELECT REG. SET #1
5466 027714 005002      CLR    R12              ;SHOULD CLEAR REG #12
5467 027716 012746 000100  MOV    #PR2,-(SP)
5468 027722 010746  MOV    PC,-(SP)
5469 027724 062716 000006  ADD    #1$-.,(SP)       ;FORM NEW PC
5470 027730 000006  RTT
5471 027732 013700 177776  1$:   MOV    @PSW,R0          ;NOW USING REG SET 0
5472 027736 005702      TST    R2              ;SHOULD TEST R2 NOT R12
5473 027740 001001      BNE    4$
5474 027742 104000      HLT
5475 027744 022700 000100  4$:   CMP    #PR2,R0          ;TESTS THE PSW AFTER THE RTT
5476 027750 001436      BEQ    RTT2EX
5477 027752 104000      HLT
5478 027754 000434      BR     RTT2EX           ;ERROR. INCORRECT PSW AFTER THE RTT
5479
5480                      ;TEST TO INSURE THAT RTI DOES NOT CLEAR BITS 11-15 IN USER MODE
5481 027756 052737 030340 177776  2$:   BIS    #PUM+PR7,@PSW   ;PSW<15-5>=144X
5482 027764 005046      CLR    -(SP)
5483 027766 010746  MOV    PC,-(SP)
5484 027770 062716 000006  ADD    #5$-.,(SP)
5485 027774 000002  RTI
5486 027776 022737 174340 177776  5$:   CMP    #UM+PUM+REG+PR7,@PSW ;ATTEMPS TO INSERT A PSW OF 0
5487 030004 001420      BEQ    RTT2EX          ;SHOULD CHECK AGAINST REG #0
5488 030006 104000      HLT
5489 030010 000416      BR     RTT2EX          ;ERROR. RTI CLEARED BITS IN PSW
5490
5491                      ;TEST THAT BITS 11-15 AND PRIORITY BITS ARE NOT ALTERED IN SUPFR MODE
5492 030012 052737 030200 177776  3$:   BIS    #PUM+PR4,@PSW   ;PSW<15-5>=044X
5493 030020 012746 000340  MOV    #PR7,-(SP)
5494 030024 010746  MOV    PC,-(SP)
5495 030026 062716 000006  ADD    #6$-.,(SP)
5496 030032 000006  RTT
5497                      ;ATTEMPS TO CLEAR 11-15 AND ALTER PR
5498 030034 022737 074200 177776  6$:   CMP    #SM+PUM+REG+PR4,@PSW
5499 030042 001401      BEQ    RTT2EX
5500 030044 104000      HLT
5501                      ;ERROR! RTT ALTERED PR IN
5502 030046 016737 177572 177776  RTT2EX: MOV    SAVPSW,@PSW   ;SUPER MODE OR BITS 11-15.
5503 030054 000004  RELE5: SCOPE
5504 030056 010702  MOV    PC,R2
5505 030060 062702 000012  ADD    #12,R2
5506 030064 012707 043764  MOV    #RELOC,PC       ;GO RELOCATE PROGRAM CODE
5507 030070 000000  REL55: .WORD 0
5508                      ;55555555555555 LAST ADDRESS OF CODE TO BE RELOCATED 5555555555
5509
5510                      ;:*****
```

```

5511      :*TEST 53      CHECK ASH, ASHC, MUL, AND DIV INSTRUCTIONS
5512      :*****
5513 030072 012767 000001 151226      :MOV      #1,$TIMES      ;;DO 1 ITERATION
5514 030100 000004                        TST53: SCOPE
5515 030102 112737 000053 001202      :MOVB     #53,@$STSTNM      ;LOAD TEST NUMBER
5516 030110 013737 001202 177570      :MOV      @$STSTNM,@$DISPLAY ;DISPLAY TEST NUMBER
5517
5518      .SBTTL      START OF SECTION 6
5519      ;6666666666666666 FIRST ADDRESS TO BE RELOCATED 6666666666
5520 030116 010700      REL6: MOV      PC,R0      ;GET PC
5521 030120 005740      TST      -(R0)      ;R0 CONTAINS THE ADDRESS OF REL6
5522 030122 010037 001540      MOV      R0,@$FRSTAD     ;SAVE
5523 030126 010700      MOV      PC,R0      ;GET CURRENT PC
5524 030130 162700 030130      SUB      #.,R0      ;SUBTRACT RELOCATION FACTOR
5525 030134 010037 001534      MOV      R0,@$FACTOR     ;SAVE RELOCATION FACTOR
5526 030140 010737 001212      MOV      PC,@$SLPERR     ;SET LOOP ADDRESS
5527 030144 062737 000030 001212      ADD      #30,@$SLPERR    ;ADJUST
5528 030152 013737 001212 001210      MOV      @$SLPERR,@$SLPADR
5529 030160 105737 001530      TSTB    @ANEXEC      ;BR IF TEST CODE TO BE EXECUTED
5530 030164 001402      BEQ     .+6
5531 030166 000167 002120      JMP     REL6
5532 030172 012700 000001      ASHL0: MOV      #1,R0      ;R0 WILL BE THE SHIFT COUNT
5533 030176 012703 000021      MOV      #17.,R3      ;MAX SHIFT COUNT
5534 030202 005067 000014      1$:    CLR     2$      ;PRESET SAVED CC'S LOCATION-0
5535 030206 010002      MOV      R0,R2      ;GET SHIFT COUNT FOR PASS
5536 030210 010705      MOV      PC,R5      ;R5 & R4 WILL BE DATA SHIFTED BY
5537 030212 010504      MOV      R5,R4      ;ASH & ASL INSTRUCTIONS
5538 030214 072502      ASH     R2,R5      ;SHIFT R5
5539 030216 113727 177776      MOVB    @$PSW,(PC)+   ;SAVE CC'S
5540 030222 000000      2$:    .WORD  0      ;CONTAINS ASH CC'S IN EVEN BYTE
5541                                     ;ASL CC'S IN ODD BYTE
5542 030224 006304      3$:    ASL     R4      ;SHIFT R4
5543 030226 113746 177776      MOVB    @$PSW,-(SP)   ;SAVE PSW ON STACK
5544 030232 132716 000002      BITB    #2,(SP)      ;CHECK IF ASL SET V BIT
5545 030236 001403      BEQ     30$
5546 030240 152767 000002 177755      BISB    #2,2$+1      ;IF ASL SET V THEN SET V IN 2$+1
5547 030246 112637 177776      30$:   MOVB    (SP)+,@$PSW   ;RESTORE ORIGINAL PSW
5548 030252 077214      SOB     R2,3$        ;SHIFT R4 R2 TIMES
5549 030254 153767 177776 177741      BISB    @$PSW,2$+1   ;SAVE CC'S AFTER ASL
5550 030262 020504      CMP     R5,R4      ;CHECK ASH & ASL RESULTS
5551 030264 001004      BNE     4$
5552 030266 126767 177730 177727      CMPB    2$,2$+1      ;CHECK ASH & ASL CC'S
5553 030274 001401      BEQ     .+4
5554 030276 104000      4$:    HLT
5555 030300 005200      INC     R0      ;ERROR! INCORRECT RESULT OR CC'S
5556 030302 020003      CMP     R0,R3      ;INCREMENT PASS SHIFT COUNT
5557 030304 001336      BNE     1$
5558
5559 030306 012700 177777      ASHR0: MOV      #-1,R0     ;R0 = RIGHT SHIFT COUNT FOR PASS
5560 030312 012703 177757      MOV      #-17.,R3    ;MAX SHIFT COUNT
5561 030316 010002      1$:    MOV      R0,R2     ;GET SHIFT COUNT FOR PASS
5562 030320 010705      MOV      PC,R5     ;R5 & R4 = DATA TO BE SHIFTED
5563 030322 010504      MOV      R5,R4     ;BY ASH & ASR INSTRUCTIONS
5564 030324 072502      ASH     R2,R5     ;SHIFT R5 R2 TIMES
5565 030326 113727 177776      MOVB    @$PSW,(PC)+   ;SAVE CC'S IN EVEN BYTE
5566 030332 000000      2$:    .WORD  0      ;CONTAINS ASH CC'S IN EVEN BYTE
  
```

```

5567                                     :ASR CC'S IN ODD BYTE
5568 030334 005402                     NEG      R2
5569 030336 006204                     3$: ASR      R4           :SHIFT R4
5570 030340 077202                     SOB      R2,3$         :SHIFT R4 R2 TIMES
5571 030342 113767 177776 177763      MOVB    @PSW,2$+1     :SAVE CC'S AFTER ASR
5572 030350 142767 000002 177755      BICB    #2,2$+1      :ASH RIGHT WILL NOT SET V ASR MAY SET V
5573 030356 020504                     CMP      R5,R4        :CHECK ASH & ASR RESULTS
5574 030360 001004                     BNE     4$
5575 030362 126767 177744 177743      CMPB    2$,2$+1      :CHECK ASH & ASR CC'S
5576 030370 001401                     BEQ     .+4
5577 030372 104000                     4$: HLT
5578 030374 005300                     DEC      R0           :DECREMENT PASS SHIFT COUNT
5579 030376 020003                     CMP      R0,R3
5580 030400 001346                     BNE     1$
5581
5582 030402 012746 000037      ASHCLO: MOV    #-31,-(SP) :PUT MAX SHIFT COUNT ON STACK
5583 030406 012746 000001      MOV     #1,-(SP)       :PUT LEFT SHIFT COUNT ON STACK
5584 030412 011600      1$: MOV     (SP),R0     :GET PASS SHIFT COUNT
5585 030414 010705      MOV     PC,R5          :CURRENT PC IS DATA TO BE SHIFTED
5586 030416 010503      MOV     R5,R3         :ASHC SHIFTS R4,R5;ASL,ROL SHIFTS R2,R3
5587 030420 005004      CLR     R4
5588 030422 005002      CLR     R2
5589 030424 073400      ASHC    R0,R4         :SHIFT R4 LEFT AS SPECIFIED BY R0
5590 030426 006303      2$: ASL     R3         :SHIFT R2,R3 LEFT
5591 030430 006102      ROL     R2           :AS SPECIFIED BY R0
5592 030432 077003      SOB     R0,2$
5593 030434 020402      CMP     R4,R2        :CHECK RESULTS
5594 030436 001002      BNE     3$
5595 030440 020503      CMP     R5,R3
5596 030442 001401      BEQ     .+4
5597 030444 104000      3$: HLT
5598 030446 005216      INC     (SP)         :INCREMENT NEXT PASS SHIFT COUNT
5599 030450 021666 000002      CMP     (SP),2(SP)   :REACHED MAX COUNT (31.)
5600 030454 001356      BNE     1$
5601 030456 022626      CMP     (SP)+,(SP)+  :RESTORE STACK PTR
5602
5603 030460 012746 177740      ASHCRO: MOV    #-32,-(SP) :PUT MAX RIGHT SHIFT COUNT ON STACK
5604 030464 012746 177777      MOV     #-1,-(SP)    :PUT PASS SHIFT COUNT ON STACK
5605 030470 011600      1$: MOV     (SP),R0     :GET PASS SHIFT COUNT
5606 030472 010702      MOV     PC,R2        :R2,R3 & R4,R5 ARE THE DATA REGISTERS
5607 030474 010204      MOV     R2,R4        :TO BE SHIFTED BY TEST
5608 030476 005003      CLR     R3
5609 030500 005005      CLR     R5
5610 030502 000262      SEV
5611 030504 073200      ASHC    R0,R2        :SET V BIT IN PSW
5612 030506 102410      BVS    3$           :SHIFT R2,R3 RIGHT R0 TIMES
5613 030510 005400      NEG     R0           :SHIFT RIGHT CLEARS V
5614 030512 006204      2$: ASR     R4         :NEGATE SHIFT COUNT FOR SOB
5615 030514 006005      ROR     R5           :SHIFT R4,R5 RIGHT R0 TIMES
5616 030516 077003      SOB     R0,2$
5617 030520 020204      CMP     R2,R4        :CHECK RESULT
5618 030522 001002      BNE     3$
5619 030524 020305      CMP     R3,R5
5620 030526 001401      BEQ     .+4
5621 030530 104000      3$: HLT
5622 030532 005316      DEC     (SP)         :SET SHIFT COUNT FOR NEXT PASS
    
```


5623 030534 021666 000002
 5624 030540 001353
 5625 030542 022626
 5626
 5627
 5628
 5629
 5630
 5631
 5632 030544 000004
 5633 030546 112737 000054 001202
 5634 030554 013737 001202 177570
 5635 030562 012700 000001
 5636 030566 012706 000700
 5637 030572 005016
 5638 030574 010702
 5639 030576 010227
 5640 030600 000000
 5641 030602 005003
 5642 030604 005004
 5643 030606 010205
 5644 030610 100001
 5645 030612 005104
 5646 030614 000277
 5647 030616 070200
 5648
 5649 030620 102406
 5650 030622 001405
 5651 030624 073416
 5652
 5653 030626 020204
 5654 030630 001002
 5655 030632 020305
 5656 030634 001401
 5657 030636 104000
 5658 030640 005216
 5659 030642 006300
 5660 030644 102353
 5661
 5662 030646 010702
 5663 030650 005202
 5664 030652 010227
 5665 030654 000000
 5666 030656 005103
 5667 030660 010204
 5668 030662 006204
 5669 030664 005104
 5670 030666 070200
 5671
 5672 030670 020204
 5673 030672 001002
 5674 030674 020003
 5675 030676 001401
 5676 030700 104000
 5677
 5678

```

CMP      (SP),2(SP)      ;CHECK IF MAX SHIFT COUNT
BNE      1$
CMP      (SP)+,(SP)+    ;RESTORE STACK PIR
:*****
:TEST 54      CHECK MUL
:*          THE BELOW TEST OF THE MUL INSTRUCTION MULTIPLIES THE CURRENT PC
:*          BY 1,2,4,8 ETC AND SHIFTS THE SAME PC VALUE USING AN ASHC LEFT BY
:*          0,1,2,3,ETC AND COMPARES THE RESULTS. CONDITION CODE RESULTS ARE NOT CHECKED.
:*****
TST54:  SCOPE
        MOV      #54,2$TSTNM      ;LOAD TEST NUMBER
        MOV      2$TSTNM,2$DISPLAY ;:DISPLAY TEST NUMBER
MULO:   MOV      #1,R0            ;R0 CONTAINS MULTIPLIER FOR MUL
        MOV      #SUPSTK,SP      ;SETUP THE STACK
        CLR      (SP)            ;(SP) CONTAINS SHIFT VALUE FOR ASHC
1$:      MOV      PC,R2           ;R3,R2 & R5,R4 ARE DATA REGISTERS
        MOV      R2,(PC)+        ;SAVE MULTIPLICAND
        .WORD    0               ;CONTAINS ORIGINAL MULTIPLICAND
        CLR      R3
        CLR      R4
        MOV      R2,R5          ;FOR MUL AND ASHC
        BPL      .+4            ;IF MULTIPLICAND IS NEG THEN SET R4  -1
        COM      R4            ;FOR ASHC
        SCC      ;PRESET CC'S
        MUL      R0,R2          ;MULTIPLY R2 BY R0 LEAVE PRODUCT
        ;IN R2,R3 MSH IN R2,LSH IN R3
        BVS      2$
        BEQ      2$            ;PRODUCT WILL NEVER BE = 0
        ASHC    (SP),R4        ;'MULTIPLY' R4,R5 BY (SP) LEAVE PRODUCT
        ;IN R4,R5 MSH IN R4,LSH IN R5
        CMP      R2,R4          ;CHECK MSH RESULT
        BNE      2$
        CMP      R3,R5          ;CHECK LSH RESULT
        BEQ      .+4
2$:      HLT
        INC      (SP)          ;INCREMENT ASHC SHIFT COUNT
        ASL      R0            ;SHIFT MUL MULTIPLIER
        BVC      1$
;CHECK MUL INST WITH MULTIPLIER (R0) = 100000
        MOV      PC,R2          ;R2 = MULTIPLICAND
        INC      R2
        MOV      R2,(PC)+        ;SAVE MULTIPLICAND
        .WORD    0               ;CONTAINS ORIGINAL MULTIPLICAND
        COM      R3
        MOV      R2,R4          ;R4 WILL BE MSH 'PRODUCT'
        ASR      R4            ;FORM 'PRODUCT'
        COM      R4            ;COMPLEMENT MSH 'PRODUCT'
        MUL      R0,R2          ;MULTIPLY R2 BY 100000 LEAVING
        ;R2 = MSH, R3 = LSH PRODUCT
        CMP      R2,R4          ;COMPARE MSH PRODUCTS
        BNE      3$
        CMP      R0,R3          ;CHECK LSH PRODUCT
        BEQ      .+4
3$:      HLT
:*****
:TEST 55      CHECK THE DIV INSTRUCTION
    
```

```

5679      :*      THE BELOW TEST OF THE DIV INSTRUCTION DIVIDES THE CURRENT PC BY
5680      :*      1,2,4,8,ETC LEAVING THE QUOTIENT/REMAINDER IN R2/R3. NEXT THE QUOTIENT
5681      :*      IS MULTIPLIED BY 1,2,4,8,ETC AND THE REMAINDER ADDED. THE RESULT IS
5682      :*      THEN COMPARED WITH THE ORIGINAL CURRENT PC.
5683      :*****
5684 030702 000004      TST55: SCOPE
5685 030704 112737 000055 001202      MOVB #55,@#STSTNM ;LOAD TEST NUMBER
5686 030712 013737 001202 177570      MOV @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
5687 030720 012700 000001      DIV0: MOV #1,R0 ;R0=DIVISOR
5688 030724 010716      MOV PC,(SP) ;SAVE DATA ON STACK
5689 030726 011603      1$: MOV (SP),R3 ;GET DATA
5690 030730 005002      CLR R2 ;CLEAR MSH DIVIDEND
5691 030732 000277      SCC
5692 030734 071200      DIV R0,R2 ;DIVIDE R2 BY R0 LEAVING QUOTIENT IN R2
5693      ;AND REMAINDER IN R3
5694 030736 103417      BCS 2$
5695 030740 100416      BMI 2$
5696 030742 102007      BVC 20$ ;BRANCH IF DIVIDE WORKED
5697 030744 022700 000001      CMP #1,R0 ;V BIT SHOULD ONLY SET IF DIVIDING BY 1
5698 030750 001012      BNE 2$ ;AND THE LSH OF DIVIDEND
5699 030752 032716 100000      BIT #100000,(SP) ;IS NEGATIVE
5700 030756 001407      BEQ
5701 030760 000407      BR 3$
5702 030762 010204      20$: MOV R2,R4 ;GET QUOTIENT
5703 030764 070400      MUL R0,R4 ;MULTIPLY QUOTIENT BY DIVISOR
5704 030766 060305      ADD R3,R5 ;ADD REMAINDER TO LSH PRODUCT
5705 030770 103402      BCS 2$ ;SHOULD BE NO CARRY
5706 030772 021605      CMP (SP),R5 ;CHECK RESULT
5707 030774 001401      BEQ
5708 030776 104000      2$: HLT ;ERROR! DIVIDE FAILED
5709      ;QUOTIENT IS IN R2,REMAINDER IN R3
5710      ;ORIGINAL PC IS ON STACK AND FINAL
5711      ;PRODUCT IN R4,R5 [MSH][LSH]
5712 031000 006300      3$: ASL R0 ;GET NEXT DIVISOR
5713 031002 102351      BVC 1$
5714
5715      ;CHECK ASH,ASHC,MUL, AND DIV INSTRUCTIONS USING ADDRESS MODE 1
5716 031004 005016      ASHL1: CLR (SP) ;(SP) = SHIFT COUNT
5717 031006 005000      CLR R0 ;R0 = SHIFT COUNT FOR CHECK ASH
5718 031010 012702 000020      MOV #16.,R2 ;R2 = MAX LEFT SHIFT COUNT
5719 031014 005067 000012      1$: CLR 2$ ;CLEAR CC'S HOLDING ADDRESS
5720 031020 010703      MOV PC,R3 ;R3,R4 = DATA TO BE SHIFTED
5721 031022 010304      MOV R3,R4
5722 031024 072316      ASH (SP),R3 ;SHIFT R3 LEFT (SP) TIMES
5723 031026 013727 177776      MOV @#PSW,(PC)+ ;SAVE CC'S
5724 031032 000000      2$: .WORD 0 ;CONTAINS ASH (SP),R3 (C'S IN EVEN BYTE
5725      ;AND ASH R0,R4 CC'S IN ODD BYTE
5726 031034 072400      ASH R0,R4 ;SHIFT R4 LEFT R0 TIMES
5727 031036 113767 177776 177767      MOVB @#PSW,2$+1 ;SAVE CC'S IN ODD BYTE OF 2$
5728 031044 020304      CMP R3,R4 ;COMPARE RESULTS
5729 031046 001004      BNE 3$ ;BRANCH IF THEY DO NOT COMPARE
5730 031050 126767 177756 177755      CMPB 2$,2$+1 ;CHECK CC'S AFTER ASH INSTRUCTIONS
5731 031056 001401      BEQ
5732 031060 104000      3$: HLT ;ERROR. EITHER RESULTS OF SHIFT OR
5733      ;RESULT CC'S ARE INCORRECT
5734 031062 005200      INC R0 ;INCREMENT SHIFT COUNT FOR ASH R0,R4
    
```

```
5735 031064 005216          INC      (SP)          ;INCREMENT SHIFT COUNT FOR ASH (SP),R3
5736 031066 020200          CMP      R2,R0        ;CHECK FOR MAX SHIFT COUNT
5737 031070 001351          BNE     1$
5738
5739 031072 005016          ASHR1:  CLR      (SP)          ;(SP) = SHIFT COUNT FOR ASH (SP),R4
5740 031074 005000          CLR      R0           ;R0 = SHIFT COUNT FOR ASH R0,R5
5741 031076 005402          NEG      R2           ;R2 = MAX RIGHT SHIFT COUNT (SET BY
5742                          ;ABOVE TEST TO 16. NOW = -16.
5743 031100 005067 000012    1$:     CLR      2$          ;CLEAR CC'S HOLDING ADDRESS
5744 031104 010704          MOV      PC,R4        ;R4,R5 = DATA TO BE SHIFTED RIGHT
5745 031106 010405          MOV      R4,R5
5746 031110 072416          ASH     (SP),R4       ;SHIFT R4 RIGHT (SP) TIMES
5747 031112 013727 177776    MOV      @RPSW,(PC)+  ;SAVE CC'S
5748 031116 000000    2$:     .WORD   0          ;CONTAINS ASH (SP),R4 CC'S IN EVEN BYTE
5749                          ;AND ASH R0,R5 CC'S IN ODD BYTE
5750 031120 072500          ASH     R0,R5         ;SHIFT R5 RIGHT R0 TIMES
5751 031122 113767 177776 177767 MOVB     @RPSW,2$+1    ;SAVE CC'S IN ODD BYTE 2$
5752 031130 020405          CMP      R4,R5        ;CHECK RESULTS
5753 031132 001004          BNE     3$
5754 031134 126767 177756 177755 CMPB     2$,2$+1      ;CHECK RESULT CC'S
5755 031142 001401          BEQ     .+4
5756 031144 104000    3$:     HLT
5757                          ;ERROR! EITHER RESULTS OR RESULT CC'S
5758                          ;DID NOT COMPARE
5758 031146 005300          DEC      R0           ;DECREMENT SHIFT COUNT
5759 031150 005316          DEC     (SP)         ;DECREMENT SHIFT COUNT FOR ASH (SP),R4
5760 031152 020002          CMP      R0,R2        ;CHECK FOR MAX RIGHT SHIFT
5761 031154 001351          BNE     1$
```

```
5762
5763
5764
5765
5766
5767
5768 031156 000004          ;*****
5769 031160 112737 000056 001202 ;*TEST 56      DIVIDE AGAIN
5770 031166 013737 001202 177570 ;*
5771 031174 010703          ;* THE BELOW TEST CHECKS THE DIVIDE INSTRUCTION BY DIVIDING
5772 031176 006702          ;* THE CURRENT PC BY ITSELF+1. THE QUOTIENT (IN R2) ALWAYS = 0,
5773 031200 010304          ;* AND THE REMAINDER (IN R3) ALWAYS = THE CURRENT PC.
5774 031202 010316          ;*****
5775 031204 005216          TST56:  SCOPE
5776 031206 100002          MOVB     #56,@R5TSTM  ;LOAD TEST NUMBER
5777 031210 162716 000002    MOV      @R5TSTM,@R5DISPLAY ;:DISPLAY TEST NUMBER
5778 031214 071216          DIV1:   MOV      PC,R3        ;CURRENT PC IS LSH DIVIDEND
5779 031216 103410          SXT     R2           ;EXTEND SIGN TO R2 (MSH DIVIDEND)
5780 031220 102407          MOV     R3,R4        ;SAVE ORIGINAL DIVIDEND
5781 031222 001006          MOV     R3,(SP)      ;PUT ON STACK
5782 031224 100405          INC     (SP)         ;ADD 1 (WILL BE DIVISOR)
5783 031226 005702          BPL     1$          ;BRANCH IF POSITIVE
5784 031230 001361          SUB     #2,(SP)      ;MAKE DIVISOR 1 LESS THAN DIVIDEND
5785 031232 010416          1$:     DIV     (SP),R2   ;DIVIDE R2 BY (SP)
5786 031234 020316          BCS     2$          ;CHECK CONDITION CODES
5787 031236 001401          BVS     2$
5788 031240 104000          BNE     2$
5789                          BMI     2$
5790                          TST     R2           ;CHECK QUOTIENT (R2 - 0)
5791                          BNE     DIV1
5792                          MOV     R4,(SP)      ;GET ORIGINAL DIVISOR
5793                          CMP     R3,(SP)      ;CHECK REMAINDER
5794                          BEQ     .+4
5795                          2$:     HLT
5796                          ;REPORT ERROR
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
```

```
5791
5792 031242 000004
5793 031244 112737 000057 001202
5794 031252 013737 001202 177570
5795 031260 012702
5796 031262 000237
5797 031264 005004
5798 031266 042744 000340
5799 031272 011403
5800 031274 042703 177757
5801
5802 031300 012767 000230 000010
5803 031306 012767 000237 000050
5804 031314 000257
5805 031316 000230
5806 031320 121403
5807 031322 001401
5808 031324 104000
5809 031326 03274 140000
5810 031332 001002
5811 031334 062703 000040
5812 031340 005267 177752
5813 031344 026702 177746
5814 031350 002761
5815 031352 012702
5816 031354 000230
5817 031356 052703 000017
5818 031362 000277
5819 031364 000237
5820 031366 121403
5821 031370 001401
5822 031372 104000
5823 031374 032714 140000
5824 031400 001002
5825 031402 162703 000040
5826 031406 005367 177752
5827 031412 026702 177746
5828 031416 002361
5829
5830
5831
5832
5833
5834
5835
5836 031420 000004
5837 031422 112737 000060 001202
5838 031430 013737 001202 177570
5839 031436 012700 031602
5840
5841 031442 012702 000400
5842 031446 005003
5843 031450 012704 177772
5844 031454 005014
5845 031456 013737 177776 000242
5846 031464 112737 000340 000242
```

```
*****
TST57: SCOPE
        MOV      #57, @S7STNM          ;LOAD TEST NUMBER
        MOV      @S7STNM, @DISPLAY    ;:DISPLAY TEST NUMBER
SPLTST: MOV      (PC)+, R2             ;R2 CONTAINS OP CODE FOR SPL 7
        SPL      7
        CLR      R4
        BIC      #PR7, -(R4)          ;CLEAR PRIORITY LEVEL BITS IN PSW
        MOV      (R4), R3             ;GET CURRENT PSW
        BIC      #177757, R3         ;R3 CONTAINS CORRECT PSW AFTER SPL
        MOV      #SPL+0, 2$          ;INITIALIZE SPL INSTRUCTIONS
        MOV      #SPL+7, 5$
1$:     CCC
2$:     SPL      0                    ;CLEAR CONDITION CODES
        CMPB     (R4), R3             ;SET PRIORITY LEVEL (NOTE: SPL-NOP IF USER/SUPER MODE)
        BEQ      .+4                 ;CHECK RESULT OF SPL ABOVE
        HLT      .                   ;ERROR! SPL ABOVE FAILED
        BIT      #UM, (R4)           ;IF NOT IN KERNEL MODE THEN SPL
        BNE      3$                  ;ACTS AS A NOP
        ADD      #40, R3             ;SET NEXT CORRECT PSW RESULT
        INC      2$                  ;SET NEXT SPL INSTRUCTION
        CMP      2$, R2              ;CHECK IF DONE
        BLT      1$                  ;LOOP UNTIL DONE CHANGING SPL EACH PASS
        MOV      (PC)+, R2           ;R2 CONTAINS SPL INSTRUCTION BELOW
        SPL      0
        BIS      #17, R3             ;SET CONDITION CODE RESULT INTO R3
4$:     SCC
5$:     SPL      7                    ;SET CONDITION CODES
        CMPB     (R4), R3             ;SET PRIORITY LEVEL
        BEQ      .+4                 ;CHECK RESULT OF SPL ABOVE
        HLT      .                   ;ERROR! SPL ABOVE FAILED
        BIT      #UM, (R4)           ;CHECK IF IN KERNEL MODE
        BNE      6$
        SUB      #40, R3             ;SET NEXT CORRECT PSW RESULT
6$:     DEC      5$                  ;SET NEXT SPL
        CMP      5$, R2              ;CHECK IF DONE ALL SPL'S
        BGE      4$
*****
*TEST 60 CHECK PIRQ LOGIC
* THIS TEST CHECKS THAT WHEN A REQUEST IS MADE AT A LEVEL = TO THE
* CURRENT PROCESSER PRIORITY LEVEL THAT NO INTERRUPT TAKES PLACE, AND
* THAT WHEN A REQUEST IS MADE AT A LEVEL 1 GREATER THAN THE CURRENT PRO-
* CESSER LEVEL THAT AN INTERRUPT OCCURS
*****
TST60: SCOPE
        MOV      #60, @S7STNM        ;LOAD TEST NUMBER
        MOV      @S7STNM, @DISPLAY   ;:DISPLAY TEST NUMBER
PIRQ0:  MOV      #4$, R0              ;R0 POINTS TO A TABLE OF CORRECT PIRQ
        MOV      #400, R2            ;CONTENTS AFTER AN INTERRUPT
        CLR      R3                  ;R2 CONTAINS INTERRUPT REQUEST LEVEL
        MOV      #PIRQ, R4           ;R3 CONTAINS PROCESSER PRIORITY LEVEL
        CLR      (R4)                ;R4 CONTAINS ADDRESS OF PIRQ REGISTER
        MOV      @PSW, @PIRQVEC+2    ;INITIALZE REQUEST LEVEL TO 0
        MOV      #PR7, @PIRQVEC+2    ;RETAIN MODE & REG SET ON TRAP
        MOV      #PR7, @PIRQVEC+2    ;ASSUME LEVEL 7 ON INTERRUPT
```

```

5847 031472 112737 000340 000016      MOVB  #PR7,@TBITVEC+2      ;PRIORITY LEVEL 7 ON TRAP
5848 031500 012737 031540 000240 1$:  MOV  #2$,@PIRQVEC          ;SET PIRQ ERROR INTERRUPT VECTOR
5849 031506 063737 001534 000240      ADD  @FACTOR,@PIRQVEC      ;ADD RELOCATION FACTOR
5850 031514 110337 177776              MOVB  R3,@PSW              ;SET CP PRIORITY LEVEL
5851 031520 050214              BIS  R2,(R4)               ;MAKE REQUEST AT LEVEL = TO CP LEVEL
5852 031522 100436              BMI  $$                    ;BRANCH WHEN DONE
5853 031524 062737 000002 000240      ADD  #3$-2$,@PIRQVEC      ;SET PIRQ INTERRUPT VECTOR TO 3$
5854 031532 006302              ASL  R2                    ;
5855 031534 050214              BIS  R2,(R4)              ;MAKE REQUEST AT LEVEL 1 HIGHER
5856 031536 000240              NOP
5857 031540 104000              2$:  HLT                    ;ERROR! EITHER AN INTERRUPT OCCURED
5858                                ;WHEN RQST LEVEL = CP LEVEL (PIRQVEC) 2$
5859                                ;OR INTERRUPT FAILED (PIRQVEC)-3$
5860 031542 022014              3$:  CMP  (R0)+,(R4)        ;CHECK CONTENTS OF PIRQ REGISTER
5861 031544 001406              BEQ  6$                    ;
5862 031546 013737 177772 001302      MOV  @PIRQ,@STMP0         ;SAVE PIRQ
5863 031554 005037 177772              CLR  @PIRQ
5864 031560 104000              HLT
5865 031562 062703 000040              6$:  ADD  #40,R3            ;ERROR! INCORRECT PIRQ CONTENTS
5866 031566 040214              BIC  R2,(R4)              ;SET NEXT CP PRIORITY LEVEL
5867 031570 012716 031500              MOV  #1$, (SP)            ;LOWER LEVEL BY 1
5868 031574 063716 001534              ADD  @FACTOR,(SP)        ;ADJUST RETURN ADDRESS
5869 031600 000006              30$: RTT                  ;TO RETURN TO 1$
5870
5871 ;TABLE OF CORRECT PIRQ REGISTER CONTENTS ON INTERRUPT
5872 031602 001042              4$:  1042                    ;PIR1+PIA1
5873 031604 003104              ;3104                      ;PIR2+PIR1+PIA2
5874 031606 007146              ;7146                      ;PIR3+PIR2+PIR1+PIA3
5875 031610 017210              ;17210                     ;PIR4+PIR3+PIR2+PIR1+PIA4
5876 031612 037252              ;37252                     ;PIR5+PIR4+PIR3+PIR2+PIR1+PIA5
5877 031614 077314              ;77314                     ;PIR6+PIR5+PIR4+PIR3+PIR2+PIR1+PIA6
5878 031616 177356              ;177356                    ;PIR7+PIR6+PIR5+PIR4+PIR3+PIR2+PIR1+PIA7
5879
5880 031620 005014              5$:  CLR  (R4)              ;CLEAR PIRQ REGISTER
5881 031622 012737 000242 000240      MOV  #PIRQVEC+2,@PIRQVEC ;RESET PIRQVEC TO HALT AT FIRQVEC+2
5882 031630 005037 000242              CLR  @PIRQVEC+2
5883 031634 105037 177776              CLRB @PSW
5884 031640 042737 000340 000016      BIC  #PR7,@TBITVEC+2
5885 ;*****
5886 ;*TEST 61 CHECK MICRO-BREAK REGISTER
5887 ;* THIS TEST SHIFTS A '0' BIT THRU ALL BIT POSITIONS.
5888 ;*****
5889 031646 000004              TST61: SCOPE
5890 031650 112737 000061 001202      MOVB  #61,@STSTNM        ;LOAD TEST NUMBER
5891 031656 013737 001202 177570      MOV  @STSTNM,@DISPLAY    ;:DISPLAY TEST NUMBER
5892 031664 012702 177770              MBRK:  MOV  #UBREAK,R2    ;SET ADDRESS OF MICRO BREAK REGISTER
5893 031670 011246              MOV  (R2),-(SP)         ;SAVE ORIG CONTENTS
5894 031672 012700 177776              MOV  #177776,R0         ;SET DATA PATTERN
5895 031676 010003              1$:  MOV  R0,R3            ;GOING TO COMPARE DATA WITH R3
5896 031700 005737 001502              TST  @KB11E             ;IS THIS A KB11-E OR KB11-EM PROCESSOR?
5897 031704 001002              BNE  $$                 ;BR IF IT IS
5898 031706 042703 177400              BIC  #177400,R3         ;ONLY 8 BITS IN U-BREAK OF KB11-B/C
5899 031712 010012              5$:  MOV  R0,(R2)         ;LOAD REGISTER WITH PATTERN
5900 031714 021203              CMP  (R2),R3            ;AND CHECK
5901 031716 001004              BNE  3$                 ;BRANCH IF INCORRECT
5902 031720 000261              2$:  SEC                    ;SET 'C'
    
```

```
5903 031722 006100          ROL    R0          ;SHIFT DATA
5904 031724 103764          BCS    1$
5905 031726 000402          BR     4$
5906 031730 104000          3$:   HLT          ;ERROR DATA IN R0 NOT IN UBREAK REG
5907 031732 000772          BR     2$          ;CONTINUE TEST
5908 031734 012612          4$:   MOV    (SP)+,(R2) ;RESTORE ORIG UBREAK CONTENTS
5909
5910          ;*****
5911          ;*TEST 62 CHECK MFPI/MTPI INSTRUCTIONS
5912          ;*****
5912 031736 000004          TST62: SCOPE
5913 031740 112737 000062 001202          MOVB   #62,@$TSTNM ;LOAD TEST NUMBER
5914 031746 013737 001202 177570          MOV    @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
5915 031754 032737 140000 177776          MPI:  BIT    #UM,@$PSW ;KERNEL MODE?
5916 031762 001553          BEQ    ENDCP       ;YES EXIT TEST
5917 031764 010746          MOV    PC,-(SP)
5918 031766 062716 000134          ADD    #5$-.,(SP)
5919 031772 012637 000250          MOV    (SP)+,@$MMVEC ;SET MEM MGMT ABORT VECTOR
5920 031776 005046          CLR    -(SP)       ;CLEAR CHECK WORD
5921 032000 010603          MOV    SP,R3
5922 032002 010346          MOV    R3,-(SP)   ;PUT ADDRESS OF CHECK WORD ON THE STACK
5923 032004 105737 001531          TSTB   @$MMON     ;CHECK IF MEM MGMT IS ENABLED
5924 032010 001417          BEQ    1$         ;BRANCH IF OFF
5925 032012 013737 177640 177654          MOV    @$UIPAR0,@$UIPAR6 ;SET UP USER PAGE ADDR. REG.
5926 032020 012737 006006 177614          MOV    #6006,@$UIPDR6 ;SET USER PAGE DESC REG R/W UP 6 PAGES
5927 032026 013737 172240 172254          MOV    @$SIPAR0,@$SIPAR6
5928 032034 012737 006006 172214          MOV    #6006,@$SIPDR6 ;SET SUPER PAGE DESC. REG.
5929 032042 062706 140000          10$:  ADD    #140000,SP ;SET CURRENT MODE'S STACK POINTER
5930 032046 000240          NOP
5931 032050 010746          1$:   MOV    PC,-(SP)
5932 032052 062716 000024          ADD    #3$-.,(SP)
5933 032056 012637 000020          MOV    (SP)+,@$IOTVEC ;SET IOT TRAP VECTOR
5934 032062 000004          IOT
5935 032064 005266 000002          INC    2(SP)      ;INCREMENT CHECK WORD
5936 032070 001417          BEQ    6$
5937 032072 104000          4$:   HLT          ;ERROR! MFPI,MTPI FAILURE-FOR BETTER
5938 032074 000415          BR     6$          ;ISOLATION SUGGEST RUNNING MFPI DIAG. DCKTD/E
5939 032076 000240          3$:   NOP          ;PSW=KERNEL MODE,PREV USER OR SUPER MODE
5940 032100 006506          MFPI   SP         ;GET PREV. MODES STACK POINTER
5941 032102 006536          MFPI   @(SP)+     ;GET DATA (AN ADDRESS) ON PREV MODE'S STACK
5942 032104 006576 000000          MFPI   @(SP)      ;GET DATA (=0) FROM PREV MODES ADDRESS
5943 032110 000240          NOP          ;SPACE AND PUSH ONTO KERNEL STACK
5944 032112 001367          BNE    4$         ;ERROR IF BRANCH TAKEN! SHOULD HAVE A ZERO ON THE STACK
5945 032114 005116          COM    (SP)      ;COMPLEMENT OPERAND
5946 032116 006636          MTPI   @(SP)+    ;POP OPERAND OFF KERNEL STACK AND MOVE
5947          ;IT TO PREV MODE'S SPACE
5948 032120 000002          RTI          ;RETURN TO INST FOLLOWING IOT ABOVE
5949 032122 104000          5$:   HLT          ;ERROR! MEMORY MANG. ABORT
5950 032124 105037 177776          CLRB   @$PSW     ;SET PRIORITY LEVEL BACK TO 0
5951 032130 012737 064122 000250          6$:   MOV    #KTABRT,@$MMVEC ;RESTORE VECTOR
5952 032136 012737 054370 000020          MOV    #$$SCOPE,@$IOTVEC
5953 032144 012706 000700          MOV    #SUPSTK,SP ;RESTORE STACK POINTER
5954          ;*****
5955          ;*TEST 63 CHECK ILLEGAL HALT
5956          ;*****
5957 032150 000004          TST63: SCOPE
5958 032152 112737 000063 001202          MOVB   #63,@$TSTNM ;LOAD TEST NUMBER
```

```

5959 032160 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5960 032166 010746                      HALT1:  MOV      PC,-(SP)          ;GET CURRENT PC
5961 032170 062716 000022              ADD      #2$-.,(SP)
5962 032174 011637 000004              MOV      (SP),@#ERRVEC          ;SET ERROR TRAP VECTOR TO 2$ BELOW
5963 032200 012637 000010              MOV      (SP)+,@#RESVEC         ;LOAD RESERVED INST TRAP VECTOR (11/40)
5964 032204 000000                      HALT   ;SHOULD TRAP TO 4 IN USER/SUPER MODE
5965 032206 104000                      1$:    HLT                       ;ERROR! HALT ABOVE FAILED IN USER/SUPER MODE
5966 032210 000404                      BR      3$
5967 032212 010716                      2$:    MOV      PC,(SP)          ;REPLACE RETURN PC WITH
5968 032214 062716 000006              ADD      #3$-.,(SP)            ;ADDRESS OF 3$ BELOW
5969 032220 000002                      RTI                       ;RETURN (TO 3$)
5970
5971 032222 012737 064270 000004      3$:    MOV      #ERPRT,@#ERRVEC    ;RESTORE ERROR TRAP VECTOR
5972 032230 012737 064216 000010      MOV      #RESERR,@#RESVEC
5973 032236 105037 177776              CLRB    @#PSW
5974 032242 005037 177766              CLR     @#CPUERR
5975
5976                      ;*****
5976                      ;*TEST 64      CHECK RESET IN SUPER/USER MODE
5977                      ;*****
5978 032246 000004                      TST64: SCOPE
5979 032250 112737 000064 001202      MOVVB   #64,@#STSTNM          ;LOAD TEST NUMBER
5980 032256 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY    ;;DISPLAY TEST NUMBER
5981 032264 000277                      RESET1: SCC
5982 032266 013700 177776              MOV      @#PSW,R0             ;GET CURRENT PSW
5983 032272 000277                      SCC
5984 032274 000005                      RESET
5985 032276 023700 177776              CMP      @#PSW,R0             ;CHECK THAT PSW UNCHANGED BY RESET ABOVE
5986 032302 001401                      BEQ     .+4
5987 032304 104000                      HLT                       ;ERROR! RESET CLEARED MODE BITS IN PSW
5988 032306 010037 177776              MOV      R0,@#PSW            ;RESTORE PSW (FOR ERROR)
5989 032312
5990 032312 000004                      ENDCP:
5991 032314 010702                      RELE6: SCOPE
5992 032316 062702 000012              MOV      PC,R2
5993 032322 012707 043764              ADD      #12,R2
5994 032326 000000              MOV      #RELOC,PC          ;GO RELOCATE PROGRAM CODE
5995                      REL66: .WORD      0
5996                      ;6666666666666666 LAST ADDRESS OF CODE TO BE RELOCATED 666666666666
5997
5998                      ;*****
5998                      ;*TEST 65      TEST STACK LIMIT REGISTER
5999                      ;*****
6000 032330 012767 000001 146770      TST65: MOV      #1,$TIMES      ;;DO 1 ITERATION
6001 032336 000004                      SCOPE
6002 032340 112737 000065 001202      MOVVB   #65,@#STSTNM          ;LOAD TEST NUMBER
6003 032346 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY    ;;DISPLAY TEST NUMBER
6004
6005                      .SBTTL  START OF SECTION 7
6006                      ;7777777777777777 FIRST ADDRESS TO BE RELOCATED 7777777777
6007 032354 010700                      REL7:  MOV      PC,R0             ;GET PC
6008 032356 005740                      TST     -(R0)                ;R0 CONTAINS THE ADDRESS OF REL7
6009 032360 010037 001540              MOV      R0,@#FRSTAD         ;SAVE
6010 032364 010700                      MOV      PC,R0             ;GET CURRENT PC
6011 032366 162700 032366              SUB     #.,R0                ;SUBTRACT RELOCATION FACTOR
6012 032372 010037 001534              MOV      R0,@#FACTOR         ;SAVE RELOCATION FACTOR
6013 032376 010737 001212              MOV      PC,@#SLPERR         ;SET LOOP ADDRESS
6014 032402 062737 000030 001212      ADD     #30,@#SLPERR         ;ADJUST
    
```

```
6015 032410 013737 001212 001210      MOV    @#SLPERR,@#SLPADR
6016 032416 105737 001530                TSTB   @#NEXEC          ;BR IF TEST CODF TO BE EXECUTED
6017 032422 001402                        BEQ    .+6
6018 032424 000167 001236                JMP    RELE7
6019                                ;THIS TEST SHIFTS A '1' BIT THROUGH ALL BIT POSITIONS
6020 032430 012702 177774                MOV    #STKLMT,R2      ;GET ADDRESS OF STACK LIM REG
6021 032434 005022                        CLR    (R2)+           ;CLEAR STACK LIMIT REG
6022 032436 032712 000020                BIT    #20,(R2)        ;EXIT TEST IF '1' BIT IS SET
6023 032442 001116                        BNE    101$
6024 032444 052712 000340                BIS    #340,(R2)      ;SET PRIORITY LEVEL 7 TO PREVENT
6025                                ;ANY INTERRUPTS FROM OCCURRING
6026 032450 012700 000400                MOV    #400,R0        ;SET CHECK DATA
6027 032454 010042 1$:                MOV    R0,-(R2)       ;MOVE TO STACK LIMIT REG
6028 032456 022200                        CMP    (R2)+,R0       ;AND CHECK RESULT
6029 032460 001401                        BEQ    2$
6030 032462 104000                        HLT
6031                                ;ERROR! STACK LIMIT DID NOT
6032                                ;LOAD CORRECTLY. CORRECT RESULT
6033 032464 006300 2$:                ASL    R0              ;IS IN R0
6034 032466 103372                        BCC    1$             ;SHIFT '1' BIT LEFT
6035 032470 005042                        CLR    -(R2)         ;LOOP UNTIL 1 BIT SHIFTS OUT
6036                                ;CLEAR STACK LIMIT REG
6037                                ;THIS TEST CHECKS THAT A PROPER 'RED' ZONE VIOLATION OCCURS, NOTE THAT
6038                                ;NO 'RED ZONE' VIOLATION WILL OCCUR IF IN USER/SUPER MODES.
6039                                ;A RED ZONE VIOLATION PUSHES THE CURRENT PSW,PC ON A STACK AT 2 AND 0
```



```
6040 ;AND TAKES THE NEXT INSTRUCTION FROM THE PC IN LOCATION4. THE INST-
6041 ;RUCTION CAUSING THE RED ZONE VIOLATION IS 'ABORTED'.
6042 032472 010746 MOV PC,-(SP) ;GET CURRENT PC
6043 032474 062716 000060 ADD #4$-.,(SP) ;FORM ADDRESS OF 4$ BELOW
6044 032500 012637 000004 MOV (SP)+,@ERRVEC ;SET ERROR TRAP VECTOR TO 4$ BELOW
6045 032504 013737 177776 000006 MOV @PSW,@ERRVEC+2 ;RETAIN CURRENT STATUS ON TRAP
6046 032512 010712 MOV PC,(R2) ;SET STACK LIMIT TO CURRENT PC
6047 ;+400
6048 032514 011206 MOV (R2),SP ;AND STACK PTR = STACK LIMIT REG
6049 032516 010603 MOV SP,R3 ;SAVE STACK PTR
6050 032520 016304 000336 MOV 336(R3),R4 ;SAVE MEMORY LOC CONTENTS
6051 ;AT 'RED ZONE' BOUNDARY
6052 032524 032737 140000 177776 BIT #UM,@PSW ;BRANCH IF IN KERNEL MODE
6053 032532 001403 BEQ 20$
6054 032534 010466 000336 MOV R4,336(SP) ;SHOULD NOT CAUSE TRAP
6055 032540 000432 BR 100$
6056
6057 032542 005066 000336 20$: CLR 336(SP) ;SHOULD CAUSE 'RED ZONE' TRAP
6058 032546 012706 000700 3$: MOV #SUPSTK,SP ;RESTORE THE STACK
6059 032552 104000 HLT ;ERROR. FAILED TO TRAP
6060
6061 032554 032737 140000 000002 4$: BIT #UM,@#2 ;CHECK IF TRAPPED WHEN IN USER
6062 ;/SUPER MODES (2 CONTAINS OLD PSW)
6063 032562 001013 BNE 99$ ;GO TO ERROR CALL
6064 032564 010600 MOV SP,R0 ;STACK PTR SHOULD = 0
6065 032566 001011 BNE 99$ ;GO TO ERROR CALL IF NOT 0
6066 032570 026304 000336 CMP 336(R3),R4 ;CHECK THAT INST WAS ABORTED
6067 032574 001006 BNE 99$ ;GO REPORT EPRPR
6068 032576 005012 5$: CLR (R2) ;CLEAR STACK LIMIT REG
6069 032600 010705 MOV PC,R5 ;GET CURRENT PC
6070 032602 062705 177744 ADD #3$-.,R5 ;FORM ADDRESS OF 3$ ABOVE
6071 032606 020516 CMP R5,(SP) ;CHECK THAT RETURN PC IS ON
6072 ;THE STACK (AT 0)
6073 032610 001406 BEQ 100$ ;EXIT TEST
6074
6075 ;ERROR
6076 032612 005012 99$: CLR (R2) ;CLEAR STACK LIMIT REG
6077 032614 010463 000336 MOV R4,336(R3) ;RESTORE MEM LOCATION
6078 032620 012706 000700 MOV #SUPSTK,SP ;SET STACK PTR
6079 032624 104000 HLT ;ERROR!
6080 032626 010463 000336 100$: MOV R4,336(R3) ;RESTORE MEM LOCATION
6081 032632 005022 CLR (R2)+ ;CLEAR STACK LIM REG
6082 032634 012706 000700 MOV #SUPSTK,SP ;SET STACK PTR
6083 032640 042712 000340 BIC #340,(R2) ;SET PRIORITY LEVEL BACK TO 0
6084 032644 012737 064270 000004 MOV #ERPRT,@ERRVEC ;RESTORE ERROR TRAP VECTOR
6085 032652 013737 177776 000006 MOV @PSW,@ERRVEC+2
6086 032660 112737 000340 000006 MOVB #PR7,@ERRVEC+2
6087 032666 042737 000020 000006 BIC #BIT4,@ERRVEC+2
6088 032674 005037 177766 CLR @CPUERR ;CLEAR ERROR REG
6089 032700
6090 ;*****
6091 ;*TEST 66 MEMORY MANAGEMENT REGISTER TESTS
6092 ;* PDR TEST - THIS TEST WRITES 64. RANDOM #'S INTO EACH PDR REGISTER
6093 ;* NOTE: IF MEM MGMT IS ENABLED ONLY PDR/PAR PAIRS 4-6 ARE TESTED.
6094 ;*****
6095 032700 000004 TST66: SCOPE
```

```

6096 032702 112737 000066 001202      MOV      #66,@$TSTNM      ;LOAD TEST NUMBER
6097 032710 013737 001202 177570      MOV      @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
6098
6099 032716 012702 033150      KTPDR:  MOV      #PDRtbl,R2      ;SET TABLE ADDRESS OF PDR'S
6100 032722 012705 100360      MOV      #100360,R5      ;SET BIT MASK
6101 032726 012200      1$:    MOV      (R2)+,R0      ;GET PDR ADDRESS
6102 032730 001435      BEQ      100$            ;EXIT ON '0' TERMINATOR
6103 032732 012716 000010      2$:    MOV      #8,(SP)      ;SET LOOP COUNT (FOR 8 REGS)
6104 032736 105737 001531      TSTB    @MMON            ;BRANCH IF MEM MGMT DISABLED
6105 032742 0C1404      BEQ      3$              ;
6106 032744 062700 000010      ADD     #10,R0            ;SET R0 TO PDR4
6107 032750 012716 000003      MOV     #3,(SP)          ;AND LIMIT TO TEST 3 PDRS
6108 032754 012703 000040      3$:    MOV     #32,R3         ;SET DATA COUNT
6109 032760 005004      CLR     R4                ;INITIALIZE DATA TO BE WRITTEN
6110 032762 040504      4$:    BIC     R5,R4           ;CLEAR NON-SETTABLE BITS
6111 032764 010410      MOV     R4,(R0)          ;WRITE INTO PDR
6112 032766 021004      CMP     (R0),R4          ;AND CHECK DATA READ BACK
6113 032770 001013      BNE     99$              ;GO TO ERROR CALL
6114 032772 005104      COM     R4                ;COMPLEMENT DATA
6115 032774 040504      BIC     R5,R4           ;CLEAR NON-SETTABLE BITS
6116 032776 010410      MOV     R4,(R0)          ;WRITE COMPLEMENT DATA INTO PDR
6117 033000 021004      CMP     (R0),R4          ;AND CHECK
6118 033002 001006      BNE     99$              ;GO TO ERROR CALL
6119 033004 060104      ADD     R1,R4            ;STEP DATA
6120 033006 077313      SOB     R3,4$           ;
6121 033010 005020      5$:    CLR     (R0)+           ;STEP TO NEXT REGISTER
6122 033012 005316      DEC     (SP)             ;DECREMENT REGISTER COUNT
6123 033014 001357      BNE     3$              ;
6124 033016 000743      BR      1$              ;GET NEXT SET OF 8 REGISTERS
6125
6126 033020 104000      99$:   HLT                ;ERROR! INCORRECT DATA READ
6127
6128
6129 033022 000772      BR      5$              ;BACK FROM PDR. ADDRESS OF
6130 033024
6131
6132
6133
6134
6135 033024 000004      TST67: SCOPE
6136 033026 112737 000067 001202      MOV      #67,@$TSTNM      ;LOAD TEST NUMBER
6137 033034 013737 001202 177570      MOV      @$TSTNM,@$DISPLAY ;:DISPLAY TEST NUMBER
6138 033042 012702 033166      KTPAR:  MOV      #PARTBL,R2      ;GET TABLE ADDRESS OF PAR'S
6139 033046 005005      CLR     R5                ;
6140 033050 012200      1$:    MOV      (R2)+,R0      ;GET PAR ADDRESS
6141 033052 001435      BEQ     100$            ;EXIT ON '0' TERMINATOR
6142 033054 012716 000010      2$:    MOV      #8,(SP)      ;SET LOOP COUNT (FOR 8 REGS.)
6143 033060 105737 001531      TSTB    @MMON            ;BRANCH IF MEM MGMT DISABLED
6144 033064 001404      BEQ     3$              ;
6145 033066 062700 000010      ADD     #10,R0            ;SET R0 TO PAR4
6146 033072 012716 000003      MOV     #3,(SP)          ;AND LIMIT TEST TO 3 PARS
6147 033076 012703 000040      3$:    MOV     #32,R3         ;SET DATA COUNT
6148 033102 005004      CLR     R4                ;INITIALIZE DATA
6149 033104 040504      4$:    BIC     R5,R4           ;CLEAR NON-SETTABLE BITS
6150 033106 010410      MOV     R4,(R0)          ;WRITE INTO PAR
6151 033110 021004      CMP     (R0),R4          ;AND CHECK

```

```

*****
*TEST 67      PAR TEST
*      PAR TEST - THIS TEST WRITES 64. COMPLEMENTING RANDOM #'S INTO EACH PAR.
*****

```

```
6152 033112 001013      BNE 99$      ;TAKE ERROR EXIT
6153 033114 005104      COM R4       ;COMPLEMENT DATA
6154 033116 040504      BIC R5,R4   ;CLEAR NON-SETTABLE BITS
6155 033120 010410      MOV R4,(R0) ;WRITE COMPLEMENT DATA
6156 033122 021004      CMP (R0),R4 ;AND CHECK
6157 033124 001006      BNE 99$      ;TAKE ERROR EXIT
6158 033126 060104      ADD R1,R4   ;STEP DATA
6159 033130 077313      SOB R3,4$   ;LOOP UNTIL FINISHED
6160
6161 033132 005020      5$: CLR (R0)+
6162 033134 005316      DEC (SP)    ;DECREMENT REGISTER COUNT
6163 033136 001357      BNE 3$     ;BRANCH IF 8 REGS NOT DONE
6164 033140 000743      BR 1$
6165
6166 033142 104000      99$: HLT    ;ERROR! INCORRECT DATA READ BACK
6167                                     ;FROM PAR. ADDRESS OF PAR IS IN
6168                                     ;R0, DATA IS IN R4
6169 033144 000772      BR 5$     ;DO NEXT REGISTER
6170 033146
6171 033146 000416      100$: BR TST70 ;;GO TO NEXT TEST
6172 ;TABLES FOR PDR & PAR TESTS ABOVE
6173 033150 172300      PDRTBL: .WORD KIPDR0
6174 033152 177600      .WORD UIPDR0
6175 033154 172200      .WORD SIPDR0 ;CHANGED TO '0' IF 11/40
6176 033156 172320      .WORD KDPDR0
6177 033160 177620      .WORD UDPDR0
6178 033162 172220      .WORD SDPDR0
6179 033164 000000      .WORD 0 ;TERMINATOR
6180
6181 033166 172340      PARTBL: .WORD KIPAR0
6182 033170 177640      .WORD UIPAR0
6183 033172 172240      .WORD SIPAR0 ;CHANGED TO '0' IF 11/40
6184 033174 172360      .WORD KDPAR0
6185 033176 177660      .WORD UDPAR0
6186 033200 172260      .WORD SDPAR0
6187 033202 000000      .WORD 0 ;TERMINATOR
6188
6189 ;*****
6190 ;*TEST 70 CHECK KT ABORT LOGIC
6191 ;* THIS TEST CHECKS KT ABORT LOGIC. TEST CREATES AN ABORT CONDITION
6192 ;* AND INSURES THAT ABORT IS TAKEN PROPERLY. NOTE: TEST IS EXECUTED ONLY
6193 ;* IF TEST IS ENTERED WITH MEM MGMT ENABLED.
6194 ;*****
6195 033204 000004      TST70: SCOPE
6196 033206 112737 000070 001202      MOVB #70,@%STSTNM ;LOAD TEST NUMBER
6197 033214 013737 001202 177570      MOV @%STSTNM,@%DISPLAY ;:DISPLAY TEST NUMBER
6198 033222 105737 001531      KTABT: TSTB @%MON ;BRANCH IF MEM MGMT DISABLED
6199 033226 001515      BEQ KTEX
6200 033230 005037 172350      CLR @%KIPAR4 ;SET UP MEM MGMT REGISTERS
6201 033234 005037 172310      CLR @%KIPDR4 ;TO ABORT IF A MEMORY
6202 033240 005037 177650      CLR @%UIPAR4 ;REFERENCE IS MADE TO
6203 033244 005037 177610      CLR @%UIPDR4 ;ADDRESSES (VIRTUAL) BETWEEN
6204 033250 005037 172250      CLR @%SIPAR4
6205 033254 005037 172210      CLR @%SIPDR4
6206 033260 013746 000250      1$: MOV @%MVEC,-(SP) ;SAVE MEM MGMT VECTOR
6207 033264 013746 000252      MOV @%MVEC+2,-(SP) ;AND PRIORITY
```

```

6208 033270 010746      MOV      PC,-(SP)      ;SET MEM MGMT
6209 033272 062716 000040  ADD      #4$-.,(SP)   ;VECTOR TO 4$ BELOW
6210 033276 012637 000250  MOV      (SP)+,@MMVEC
6211 033302 013737 177776 000252  MOV      @PSW,@MMVEC+2
6212 033310 005000      CLR      R0          ;CLEAR ABORT INDICATOR
6213 033312 010702      MOV      PC,R2       ;SET R2 AND R3 NOTE:
6214 033314 012703 100000  MOV      #100000,R3  ;THE REF VIA R3 CAUSES THE
6215 033320 014223 2$:      MOV      -(R2),(R3)+ ;ABORT
6216 033322 005700 3$:      TST      R0          ;BRANCH IF THE ABORT OCCURRED
6217 033324 001001      BNE     .+4
6218 033326 104000      HLT
6219 033330 000445      BR      100$        ;REPORT ERROR
6220
6221 033332 013700 177776  ;ABORT HERE
6222 033336 000300 4$:      MOV      @PSW,R0     ;SRO SHOULD CONTAIN
6223 033340 006200      SWAB    R0          ;CAUSE FOR ABORT AND
6224 033342 042700 177637  ASR      R0          ;ALSO WHICH SEGMENT
6225 033346 062700 100011  BIC      #177637,R0  ;WAS IN USE WHEN ABORT
6226 033352 020037 177572  ADD      #100011,R0  ;OCCURRED.
6227 033356 001025      CMP     R0,@SR0
6228 033360 012700 033320  BNE     99$
6229 033364 020037 177576  MOV      #2$,R0      ;GET ADDRESS OF INST
6230 033370 001020      CMP     R0,@SR2     ;THAT ABORTED
6231 033372 012700 000362  BNE     99$
6232 033376 120037 177574  MOV      #362,R0
6233 033402 001013      CMPB   R0,@SR1     ;SR1 CONTAINS REGISTER
6234 033404 012700 000023  BNE     99$         ;MODIFICATIONS MADE
6235 033410 120037 177575  MOV      #23,R0
6236 033414 001006      CMPB   R0,@SR1+1
6237 033416 012700 033320  BNE     99$
6238 033422 005720 5$:      MOV      #2$,R0
6239 033424 020016      TST    (R0)+        ;R0=ADDRESS OF INST FOLLOWING ABORT
6240 033426 001001      CMP     R0,(SP)     ;(3$)
6241 033430 000002      BNE     99$
6242
6243 033432 104000      RTI
6244 033434 010716 99$:     HLT
6245 033436 062716 177664  ;ENTER HERE ON ERROR
6246 033442 000002      HLT
6247 033444 012637 000252  MOV      PC,(SP)
6248 033450 012637 000250  ADD      #3$-.,(SP) ;REPORT ERROR
6249 033454 012737 000001 177572  RTI
6250 033462
6251
6252
6253
6254
6255 033462 000004      MOV      #71,@STSTNM ;LOAD TEST NUMBER
6256 033464 112737 000071 001202  MOV      @STSTNM,@DISPLAY ;:DISPLAY TEST NUMBER
6257 033472 013737 001202 177570  BIT      #BITS,@MMR3  ;IS MAP ON?
6258 033500 032737 000040 172516  BNE     MAPTWO      ;BRANCH IF YES
6259 033506 001053      MAPTST: MOV      #MAPLO,R0  ;SET ADRS OF FIRST MAP REGISTER
6260 033510 012700 170200      MOV      #SUPSTK,SP  ;SETUP THE SP
6261 033514 012706 000700      MOV      #1,(SP)    ;SET BIT MASK FOR MAPLO <15-01>
6262 033520 012716 000001      MOV      #177700,R2 ;AND ALSO FOR MAPHO <21-16>
6263 033524 012702 177700

```

```

KTEX:
*****
*TEST 71      MAPPING REGISTER TESTS
*      THIS TEST LOADS RANDOM #'S INTO EACH MAPPING REGISTER
*****
TST71:  SCOPE
        MOVB   #71,@STSTNM      ;LOAD TEST NUMBER
        MOV    @STSTNM,@DISPLAY ;:DISPLAY TEST NUMBER
        BIT    #BITS,@MMR3     ;IS MAP ON?
        BNE   MAPTWO          ;BRANCH IF YES
MAPTST: MOV    #MAPLO,R0       ;SET ADRS OF FIRST MAP REGISTER
        MOV    #SUPSTK,SP      ;SETUP THE SP
        MOV    #1,(SP)        ;SET BIT MASK FOR MAPLO <15-01>
        MOV    #177700,R2     ;AND ALSO FOR MAPHO <21-16>

```

```

6264 033530 012703 000040 1$: MOV #32.,R3 ;SET DATA COUNT
6265 033534 005005 CLR R5 ;SET INITIAL DATA
6266 033536 010504 2$: MOV R5,R4 ;GET DATA
6267 033540 041604 BIC (SP),R4 ;CLEAR UNUSED BITS
6268 033542 010410 MOV R4,(R0) ;LOAD DATA INTO MAPLO <15-01>
6269 033544 021004 CMP (R0),R4 ;CHECK DATA
6270 033546 001032 BNE 99$ ;BRANCH IF INCORRECT
6271 033550 005105 COM R5 ;COMPLEMENT TEST DATA
6272 033552 010504 MOV R5,R4 ;GET TEST DATA
6273 033554 041604 BIC (SP),R4 ;CLEAR UNUSED BITS
6274 033556 010410 MOV R4,(R0) ;LOAD COMPLEMENT DATA
6275 033560 021004 CMP (R0),R4 ;AND CHECK
6276 033562 001024 BNE 99$
6277 033564 005720 TST (R0)+ ;STEP TO NEXT REGISTER
6278 033566 010504 MOV R5,R4 ;GET COMPLEMENT TEST DATA
6279 033570 040204 BIC R2,R4 ;CLEAR UNUSED BITS
6280 033572 010410 MOV R4,(R0) ;LOAD TEST DATA INTO MAPHO <21-16>
6281 033574 021004 CMP (R0),R4 ;AND CHECK
6282 033576 001016 BNE 99$
6283 033600 005105 COM R5 ;COMPLEMENT TEST DATA
6284 033602 010504 MOV R5,R4 ;GET TEST DATA
6285 033604 040204 BIC R2,R4 ;CLEAR UNUSED BITS
6286 033606 010410 MOV R4,(R0) ;LOAD TEST DATA
6287 033610 021004 CMP (R0),R4 ;AND CHECK
6288 033612 001010 BNE 99$
6289 033614 060705 ADD PC,R5 ;FORM NEXT TEST DATA
6290 033616 005740 TST -(R0) ;RESET PTR TO REGISTER <15-01>
6291 033620 077332 SOB R3,2$ ;AND TEST UNTIL ALL #'S USED
6292 033622 022020 CMP (R0)+,(R0)+ ;STEP TO NEXT REGISTER PAIR
6293 033624 022700 170400 CMP #MAPLO+128.,R0 ;BRANCH IF NOT LAST PAIR
6294 033630 001337 BNE 1$
6295 033632 000401 BR MAPTWO
6296
6297 033634 104000 99$: HLT ;ERROR! INCORRECT DATA READ BACK
6298 ;FROM MAP REG. ADRS OF REGISTER 15
6299 ;IN R0, GOOD DATA IS IN R4
6300 033636 005737 001532 MAPTWO: TST @#QV ;QV OR AUTO-ACCEPT?
6301 033642 001411 BEQ RELE7 ;BRANCH IF NO
6302 033644 012737 020000 170204 MOV #20000,@MAPL1 ;SET MAP 1 INCASE ACT11
6303 033652 005037 170206 CLR @MAPH1
6304 033656 005037 170200 CLR @MAPLO
6305 033662 005037 170202 CLR @MAPHO
6306 033666 000004 RELE7: SCOPE
6307 033670 010702 MOV PC,R2
6308 033672 062702 000012 ADD #12,R2
6309 033676 012707 043764 MOV #RELOC,PC ;GO RELOCATE PROGRAM CODE
6310 033702 000000 REL7: WORD 0
6311 ;7777777777777777 LAST ADDRESS OF CODE TO BE RELOCATED 7777777777
6312
6313 ;*****
6314 ;*TEST 72 FLOATING POINT TEST 1
6315 ;*
6316 ;* THIS TEST TAKES TWO RANDOM NUMBERS (A AND B) AND
6317 ;* COMPARES THE RESULTS OF TWO EQUAL CALCULATIONS.
6318 ;* EACH SECTION EVALUATES A DIFFERENT EQUATION AS DESCRIBED BELOW:
6319 ;* SECT1 (A+B)**2 A**2+2*A*B+B**2
  
```

6320
6321
6322
6323 033704 012767 000001 145414
6324 033712 000004
6325 033714 112737 000072 001202
6326 033722 013737 001202 177570
6327 033730 012737 000001 001326
6328
6329
6330
6331 033736 010700
6332 033740 005740
6333 033742 010037 001540
6334 033746 010700
6335 033750 162700 033750
6336 033754 010037 001534
6337 033760 010737 001212
6338 033764 062737 000030 001212
6339 033772 013737 001212 001210
6340 034000 105737 001530
6341 034004 001402
6342 034006 000167 002422
6343 034012 032737 020000 001500
6344 034020 001002
6345 034022 000167 002424
6346 034026 004737 060542
6347 034032 170127 000000
6348 034036 172537 001302
6349 034042 172437 001306
6350 034046 013737 001256 001414
6351 034054 013737 001260 001412
6352 034062 004767 002210
6353 034066 174100
6354 034070 013737 001414 001412
6355 034076 004767 002124
6356 034102 174137 001312
6357 034106 013737 001414 001262
6358
6359
6360
6361 034114 013737 001256 001412
6362 034122 172437 001302
6363 034126 013737 001412 001414
6364 034134 172500
6365 034136 004767 002064
6366 034142 174102
6367 034144 013737 001414 001416
6368
6369
6370 034152 172437 001306
6371 034156 172500
6372 034160 013737 001260 001412
6373 034166 013737 001412 001414
6374 034174 004767 002026
6375 034200 174103

```

**      SECT2  (A+B)*(A-B)=A**2-B**2
**      SECT3  A/B*B=A
:*****
:      MOV     #1,STIMES      ;;DO 1 ITERATION
TST72:  SCOPE
        MOV     #72,@STSTNM   ;LOAD TEST NUMBER
        MOV     @STSTNM,@DISPLAY ;DISPLAY TEST NUMBER
        MOV     #1,@STIMES   ;SET ITERATIONS TO 1

.SBTTL  START OF SECTION 8
:8888888888888888 FIRST ADDRESS TO BE RELOCATED 8888888888
REL8:   MOV     PC,RO          ;GET PC
        TST     -(RO)         ;RO CONTAINS THE ADDRESS OF REL8
        MOV     RO,@FRSTAD    ;SAVE
        MOV     PC,RO        ;GET CURRENT PC
        SUB     #.,RO        ;SUBTRACT RELOCATION FACTOR
        MOV     RO,@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     REL8
        BIT     #FPOPT,@OFT.CP ;FLOATING POINT AVAILABLE?
        BNE     100$        ;BRANCH IF YES
100$:   JSR     PC,@FLTSGI    ;GET RANDOM OPERANDS
        LDFFS   #0          ;INIT FPS
        LDF     @STMP0,AC1   ;LOAD A OPERAND
        LDF     @STMP2,AC0   ;LOAD B OPERAND
        MOV     @SREG0,@SAC1 ;SETUP EXTENDED
        MOV     @SREG1,@SAC0 ;EXPONENTS
        JSR     PC,FLTADD    ;PERFORM THE ADD
        STF     AC1,AC0     ;SETUP AC0 TO
        MOV     @SAC1,@SAC0 ;PERFORM THE SQUARE
        JSR     PC,FLTMPY    ;DO THE MULTIPLY
        STF     AC1,@STMP4   ;SAVE RESULT
        MOV     @SAC1,@SREG2 ;AND SOFTWARE EXP

:      ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
:      ;DO THE A*A FIRST
        MOV     @SREG0,@SAC0 ;GET EXT EXPONENT
        LDF     @STMP0,AC0   ;LOAD OPERAND A
        MOV     @SAC0,@SAC1' ;SET OPERAND B EXT EXPONENT
        LDF     AC0,AC1     ;LOAD B OPERAND
        JSR     PC,FLTMPY    ;EXECUTE THE MULTIPLY
        STF     AC1,AC2     ;SAVE RESULT

:      ;NOW DO THE B*B
        LDF     @STMP2,AC0   ;LOAD B OPERAND
        LDF     AC0,AC1     ;AND EXT EXPONENT
        MOV     @SREG1,@SAC0 ;DO THE MULTIPLY
        MOV     @SAC0,@SAC1' ;SAVE THE RESULT
        JSR     PC,FLTMPY
        STF     AC1,AC3

```

```

6376 034202 013737 001414 001420      MOV      @#SAC1,@#SAC3
6377
6378                                     ;NOW DO THE 2*B*A
6379 034210 012701 001306      MOV      #STMP2,R1
6380 034214 172411      LDF      (R1),AC0      ;LOAD THE B OPERAND
6381 034216 172541      LDF      -(R1),AC1     ;LOAD THE A OPERAND
6382 034220 013737 001260 001412      MOV      @#SREG1,@#SAC0 ;AND THE EXT EXPONENTS
6383 034226 013737 001256 001414      MOV      @#SREG0,@#SAC
6384 034234 004767 001766      JSR      PC,FLTMPY     ;DO THE MULTIPLY
6385 034240 172427 040000      LDF      #*040000,AC0 ;SETUP TO MULTIPLY BY TWO
6386 034244 012737 000002 001412      MOV      #2,@#SAC0
6387 034252 004767 001750      JSR      PC,FLTMPY     ;DO THE MULTIPLY
6388
6389                                     ;NOW SUM THE RESULTS
6390 034256 013737 001420 001412      MOV      @#SAC3,@#SAC0
6391 034264 172403      LDF      AC3,AC0      ;GET RESULT OF B*B
6392 034266 004767 002004      JSR      PC,FLTADD     ;ADD THE RESULT
6393 034272 172402      LDF      AC2,AC0      ;GET RESULT OF A*A
6394 034274 013737 001416 001412      MOV      @#SAC2,@#SAC0
6395 034302 004767 001770      JSR      PC,FLTADD     ;ADD THIS RESULT
6396 034306 174137 001316      STF      AC1,@#STMP6   ;SAVE FINAL RESULT
6397 034312 013737 001414 001264      MOV      @#SAC1,@#SREG3
6398
6399                                     ;NOW CHECK BOTH SIDES OF THE EQUATION
6400                                     ;CALCULATE THE NUMBER OF CORRECT BITS
6401                                     ;PUT LARGST EXPONENT OF A**2 OR B**2 IN SAC2
6402 034320 023737 001416 001420      CMP      @#SAC2,@#SAC3
6403 034326 002003      BGE      1$           ;BRANCH IF SAC2 ALREADY HAS LARGST
6404 034330 013737 001420 001416      MOV      @#SAC3,@#SAC2 ;SAC3 WAS LARGER
6405 034336 163737 001414 001416 1$:  SUB      @#SAC1,@#SAC2 ;NOW CALCULATE NUMBER
6406 034344 152737 000023 001416      SUB      #19,@#SAC2    ;OF CORRECT BITS WITHIN 2
6407 034352 005437 001416      NEG      @#SAC2        ;MAKE RESULT POSITIVE
6408 034356 172437 001312      LDF      @#STMP4,AC0   ;LOAD RESULT OF LEFT HAND SIDE
6409 034362 013737 001262 001412      MOV      @#SREG2,@#SAC0 ;AND EXTENDED EXPONENT
6410 034370 004767 001676      JSR      PC,FLTSUB     ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6411 034374 163737 001264 001414      SUB      @#SREG3,@#SAC1 ;GET DIFFERENCE IN EXT EXPONENTS
6412                                     ;ACTUAL EXP'S ARE EQUAL TO 200
6413 034402 100002      BPL      3$           ;ENSURE RESULT IS POSITIVE
6414 034404 005437 001414      NEG      @#SAC1
6415 034410 023737 001416 001414 3$:  CMP      @#SAC2,@#SAC1 ;ANSWERS WITHIN ALLOWABLE NUMBER?
6416 034416 003401      BLE      SECT2        ;BRANCH IF YES
6417 034420 104014      4$:  ERROR   14         ;RESULTS ARE WRONG
6418                                     ;*****
6419 034422 170127 000000      SECT2: LDFPS   #0
6420                                     ;DO A+B
6421 034426 172537 001302      LDF      @#STMP0,AC1   ;LOAD A OPERAND
6422 034432 172437 001306      LDF      @#STMP2,AC0   ;LOAD B OPERAND
6423 034436 013737 001256 001414      MOV      @#SREG0,@#SAC1
6424 034444 013737 001260 001412      MOV      @#SREG1,@#SAC0
6425 034452 004767 001620      JSR      PC,FLTADD     ;ADD THEM
6426 034456 174102      STF      AC1,AC2      ;SAVE IN AC2
6427 034460 013737 001414 001416      MOV      @#SAC1,@#SAC2 ;AND EXT EXPONENT
6428
6429                                     ;NOW DO THE A-B
6429 034466 172537 001302      LDF      @#STMP0,AC1   ;LOAD OPERAND A
6430 034472 013737 001256 001414      MOV      @#SREG0,@#SAC1 ;AND EXT EXPONENT
6431 034500 172437 001306      LDF      @#STMP2,AC0   ;LOAD OPERAND B
    
```

```

6432 034504 013737 001260 001412      MOV    @#SREG1,@#SAC0
6433 034512 004767 001554              JSR    PC,FLTSUB      ;SUBTRACT THEM
6434                                ;NOW DO (A+B)*(A-B)
6435 034516 172402              LDF    AC2,AC0        ;GET RESULT OF (A+B)
6436 034520 013737 001416 001412      MOV    @#SAC2,@#SAC0
6437 034526 004767 001474              JSR    PC,FLTMPY      ;FORM THE PRODUCT
6438 034532 174137 001312              STF    AC1,@#STMP4    ;SAVE RESULT
6439 034536 013737 001414 001262      MOV    @#SAC1,@#SREG2 ;AND EXT EXPONENT
6440                                ;NOW DO THE B*B
6441 034544 172437 001306              LDF    @#STMP2,AC0    ;LOAD OPERAND B
6442 034550 013737 001260 001412      MOV    @#SREG1,@#SAC0
6443 034556 172500              LDF    AC0,AC1        ;B OPERAND IS IN AC0
6444 034560 013737 001412 001414      MOV    @#SAC0,@#SAC1 ;AND EXT EXPONENT
6445 034566 004767 001434              JSR    PC,FLTMPY      ;
6446 034572 174102              STF    AC1,AC2        ;SAVE RESULT IN AC2
6447 034574 013737 001414 001416      MOV    @#SAC1,@#SAC2
6448                                ;NOW DO THE A*A
6449 034602 172437 001302              LDF    @#STMP0,AC0    ;LOAD OPERAND A
6450 034606 013737 001256 001412      MOV    @#SREG0,@#SAC0
6451 034614 172500              LDF    AC0,AC1
6452 034616 013737 001412 001414      MOV    @#SAC0,@#SAC1
6453 034624 004767 001376              JSR    PC,FLTMPY      ;EXECUTE THE MULTIPLY
6454 034630 013737 001414 001420      MOV    @#SAC1,@#SAC3 ;SAVE EXT EXPO OF A*A
6455                                ;NOW DO A**2-B**2
6456 034636 172402              LDF    AC2,AC0        ;GET B*B
6457 034640 013737 001416 001412      MOV    @#SAC2,@#SAC0 ;A*A IN AC1
6458 034646 004767 001420              JSR    PC,FLTSUB
6459 034652 174137 001316              STF    AC1,@#STMP6    ;SAVE IN MEMORY
6460 034656 013737 001414 001264      MOV    @#SAC1,@#SREG3
6461                                ;NOW COMPUTE THE RESULTS
6462                                ;CALCULATE THE NUMBER OF CORRECT BITS
6463 034664 023737 001416 001420      CMP    @#SAC2,@#SAC3 ;DETERMINE WHICH EXP IS LARGER
6464 034672 002003              BGE    2$             ;BRANCH IF AC2 LARGER
6465 034674 013737 001420 001416      MOV    @#SAC3,@#SAC2 ;PUT LARGEST IN AC2
6466 034702 163737 001414 001416      2$: SUB    @#SAC1,@#SAC2
6467 034710 162737 000025 001416      SUB    #21,@#SAC2
6468 034716 005437 001416              NEG    @#SAC2
6469 034722 172437 001312              LDF    @#STMP4,AC0    ;GET LEFT HAND SIDE
6470 034726 013737 001262 001412      MOV    @#SREG2,@#SAC0
6471 034734 004767 001332              JSR    PC,FLTSUB      ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6472 034740 163737 001264 001414      SUB    @#SREG3,@#SAC1 ;SUB EXT EXPONENTS
6473                                ;ACTUAL EXPONENTS ARE EQUAL
6474 034746 100002              BPL    1$             ;MAKE SURE RESULT IS POSITIVE
6475 034750 005437 001414              NEG    @#SAC1
6476 034754 023737 001416 001414      1$: CMP    @#SAC2,@#SAC1 ;RESULTS WITHIN RANGE ALLOWED?
6477 034762 003401              BLE    SECT3          ;BRANCH IF YES
6478 034764 104014              ERROR  14             ;RESULTS WRONG
6479
6480                                ;*****
6481 034766 172537 001302      SECT3: LDF    @#STMP0,AC1    ;LOAD OPERAND A
6482 034772 172437 001306              LDF    @#STMP2,AC0    ;AND OPERAND B
6483 034776 013737 001256 001414      MOV    @#SREG0,@#SAC1
6484 035004 013737 001260 001412      MOV    @#SREG1,@#SAC0
6485 035012 004767 001232              JSR    PC,FLTDIV      ;GO DIVIDE THEM
6486 035016 004767 001204              JSR    PC,FLTMPY      ;MULTIPLY RESULT BY B
6487 035022 174137 001312              STF    AC1,@#STMP4    ;SAVE RESULT

```



```
6488 035026 013737 001414 001262      MOV      @#SAC1,@#SREG2
6489 035034 172437 001302              LDF      @#STMP0,ACO      ;LOAD OPERAND A
6490 035040 174037 001316              STF      ACO,@#STMP6      ;SAVE INCASE TYPE OUT
6491 035044 013737 001256 001412      MOV      @#SREG0,@#SAC0
6492 035052 013737 001256 001264      MOV      @#SREG0,@#SREG3
6493 035060 004767 001206              JSR      PC,FLTSSUB      ;SUBTRACT RIGHT AND LEFT HAND SIDES
6494 035064 163737 001256 001414      SUB      @#SREG0,@#SAC1  ;SEE IF RESULT OK
6495 035072 100002              BPL      1$              ;ENSURE DIFFERENCE IS POSITIVE
6496 035074 005437 001414              NEG      @#SAC1
6497 035100 022737 000026 001414 1$:    CMP      #22.,@#SAC1      ;RESULTS WITHIN 2 BITS?
6498 035106 003001              BGT      2$              ;BRANCH IF NO
6499 035110 000401              BR       TST73           ;GO TO NEXT TEST
6500 035112 104014              2$:    ERROR  14           ;RESULTS WRONG
```

```
6501
6502
6503      :*****
6504      :*TEST 73      FLOATING POINT TEST 2
6505      :*
6506      :*      THIS TEST TAKES TWO RANDOM NUMBERS (A AND B) AND
6507      :*      COMPARES THE RESULTS OF TWO EQUAL CALCILATIONS.
6508      :*      EACH SECTION EVALUATES A DIFFERENT EQUATION AS DESCRIBED BELOW:
6509      :*      SECT1  (A+B)**2=A**2+2*A*B+B**2
6510      :*      SECT2  (A+B)*(A-B)=A**2-B**2
6511      :*      SECT3  A/B*B=A
6512      :*****
```

```
6512 035114 000004      TST73:  SCOPE
6513 035116 112737 000073 001202      MOV      @#73,@#STSTNM      ;LOAD TEST NUMBER
6514 035124 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
6515 035132 012737 000001 001326      MOV      #1,@#STIMES
6516 035140 004737 060534 100$:  JSR      PC,@#FLTDBL      ;GET RANDOM OPERANDS
6517 035144 170127 000200      LDFPS   #200              ;INIT FPS
6518 035150 172537 001302      LDF      @#STMP0,AC1      ;LOAD A OPERAND
6519 035154 172437 001312      LDF      @#STMP4,ACO      ;LOAD B OPERAND
6520 035160 013737 001256 001414      MOV      @#SREG0,@#SAC1  ;SETUP EXTENDED
6521 035166 013737 001260 001412      MOV      @#SREG1,@#SAC0  ;EXPONENTS
6522 035174 004767 001076      JSR      PC,FLTADD      ;PERFORM THE ADD
6523 035200 174100      STF      AC1,ACO          ;SETUP ACO TO
6524 035202 013737 001414 001412      MOV      @#SAC1,@#SAC0  ;PERFORM THE SQUARE
6525 035210 004767 001012      JSR      PC,FLTMPY      ;DO THE MULTIPLY
6526 035214 174137 001432      STF      AC1,@#FLTMP0    ;SAVE RESULT
6527 035220 013737 001414 001262      MOV      @#SAC1,@#SREG2  ;AND SOFTWARE EXP
```

```
6528
6529      :NOW DO THE RIGHT HAND SIDE OF THE EQUATION
6530      :DO THE A*A FIRST
6531 035226 013737 001256 001412      MOV      @#SREG0,@#SAC0  ;GET EXT EXPONENT
6532 035234 172437 001302      LDF      @#STMP0,ACO      ;LOAD OPERAND A
6533 035240 013737 001412 001414      MOV      @#SAC0,@#SAC1  ;SET OPERAND B EXT EXPONENT
6534 035246 172500      LDF      ACO,AC1          ;LOAD B OPERAND
6535 035250 004767 000752      JSR      PC,FLTMPY      ;EXECUTE THE MULTIPLY
6536 035254 174102      STF      AC1,AC2          ;SAVE RESULT
6537 035256 013737 001414 001416      MOV      @#SAC1,@#SAC2
```

```
6538
6539      :NOW DO THE B*B
6540 035264 172437 001312      LDF      @#STMP4,ACO      ;LOAD B OPERAND
6541 035270 172500      LDF      ACO,AC1          ;LOAD B OPERAND
6542 035272 013737 001260 001412      MOV      @#SREG1,@#SAC0  ;AND EXT EXPONENT
6543 035300 013737 001412 001414      MOV      @#SAC0,@#SAC1
```

```
6544 035306 004767 000714 JSR PC,FLTMPY ;DO THE MULTIPLY
6545 035312 174103 STF AC1,AC3 ;SAVE THE RESULT
6546 035314 013737 001414 001420 MOV @#SAC1,@#SAC3
6547
6548 ;NOW DO THE 2*B*A
6549 035322 012701 001312 MOV #STMP4,R1
6550 035326 172411 LDF (R1),AC0 ;LOAD THE B OPERAND
6551 035330 172541 LDF -(R1),AC1 ;LOAD THE A OPERAND
6552 035332 013737 001260 001412 MOV @#SREG1,@#SAC0 ;AND THE EXT EXPONENTS
6553 035340 013737 001256 001414 MOV @#SREG0,@#SAC1
6554 035346 004767 000654 JSR PC,FLTMPY ;DO THE MULTIPLY
6555 035352 172427 040000 LDF #*040000,AC0 ;SETUP TO MULTIPLY BY TWO
6556 035356 012737 000002 001412 MOV #2,@#SAC0
6557 035364 004767 000636 JSR PC,FLTMPY ;DO THE MULTIPLY
6558
6559 ;NOW SUM THE RESULTS
6560 035370 013737 001420 001412 MOV @#SAC3,@#SAC0
6561 035376 172403 LDF AC3,AC0 ;GET RESULT OF B*B
6562 035400 004767 000672 JSR PC,FLTADD ;ADD THE RESULT
6563 035404 172402 LDF AC2,AC0 ;GET RESULT OF A*A
6564 035406 013737 001416 001412 MOV @#SAC2,@#SAC0
6565 035414 004767 000656 JSR PC,FLTADD ;ADD THIS RESULT
6566 035420 174137 001442 STF AC1,@#FLTMP ;SAVE FINAL RESULT
6567 035424 013737 001414 001264 MOV @#SAC1,@#SREG3
6568
6569 ;NOW CHECK BOTH SIDES OF THE EQUATION
6570 ;CALCULATE THE NUMBER OF CORRECT BITS
6571 ;PUT LARGEST EXPONENT OF A**2 OR B**2 IN SAC2
6572 035432 023737 001416 001420 CMP @#SAC2,@#SAC3
6573 035440 002003 BGE 1$ ;BRANCH IF SAC2 ALREADY HAS LARGEST
6574 035442 013737 001420 001416 MOV @#SAC3,@#SAC2 ;SAC3 WAS LARGER
6575 035450 163737 001414 001416 1$. SUB @#SAC1,@#SAC2 ;NOW CALCULATE NUMBER
6576 035456 162737 000064 001416 SUB #52,@#SAC2 ;OF CORRECT BITS WITHIN 2
6577 035464 005437 001416 NEG @#SAC2 ;MAKE RESULT POSITIVE
6578 035470 172437 001432 LDF @#FLTMP0,AC0 ;LOAD RESULT OF LEFT HAND SIDE
6579 035474 013737 001262 001412 MOV @#SREG2,@#SAC0 ;AND EXTENDED EXPONENT
6580 035502 004767 000564 JSR PC,FLTSUB ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6581 035506 163737 001264 001414 SUB @#SREG3,@#SAC1 ;GET DIFFERENCE IN EXT EXPONENTS
6582 ;ACTUAL EXP'S ARE EQUAL TO 200
6583 035514 100002 BPL 3$ ;ENSURE RESULT IS POSITIVE
6584 035516 005437 001414 NEG @#SAC1
6585 035522 023737 001416 001414 3$: CMP @#SAC2,@#SAC1 ;ANSWERS WITHIN ALLOWABLE NUMBER?
6586 035530 003401 BLE SECT2D ;BRANCH IF YES
6587 035532 104016 4$: ERROR 16 ;RESULTS ARE WRONG
6588 ;*****
6589 035534 170127 000200 SECT2D: LDFPS #200
6590 ;DO A+B
6591 035540 172537 001302 LDF @#STMP0,AC1 ;LOAD A OPERAND
6592 035544 172437 001312 LDF @#STMP4,AC0 ;LOAD B OPERAND
6593 035550 013737 001256 001414 MOV @#SREG0,@#SAC1
6594 035556 013737 001260 001412 MOV @#SREG1,@#SAC0
6595 035564 004767 000506 JSR PC,FLTADD ;ADD THEM
6596 035570 174102 STF AC1,AC2 ;SAVE IN AC2
6597 035572 013737 001414 001416 MOV @#SAC1,@#SAC2 ;AND EXT EXPONENT
6598
6599 ;NOW DO THE A-B
LDF @#STMP0,AC1 ;LOAD OPERAND A
```

```

6600 035604 013737 001256 001414      MOV      @#SREG0,@#SAC1 ;AND EXT EXPONENT
6601 035612 172437 001312              LDF      @#STMP4,ACO   ;LOAD OPERAND B
6602 035616 013737 001260 001412      MOV      @#SREG1,@#SAC0
6603 035624 004767 000442              JSR      PC,FLTSUB    ;SUBTRACT THEM
6604                                     ;NOW DO (A+B)*(A-B)
6605 035630 172402              LDF      AC2,ACO      ;GET RESULT OF (A+B)
6606 035632 013737 001416 001412      MOV      @#SAC2,@#SAC0
6607 035640 004767 000362              JSR      PC,FLTMPY    ;FORM THE PRODUCT
6608 035644 174137 001432              STF      AC1,@#FLTMP0 ;SAVE RESULT
6609 035650 013737 001414 001262      MOV      @#SAC1,@#SREG2 ;AND EXT EXPONENT
6610                                     ;NOW DO THE B*B
6611 035656 172437 001312              LDF      @#STMP4,ACO   ;LOAD OPERAND B
6612 035662 013737 001260 001412      MOV      @#SREG1,@#SAC0
6613 035670 172500              LDF      ACO,AC1      ;B OPERAND IS IN ACO
6614 035672 013737 001412 001414      MOV      @#SAC0,@#SAC1 ;AND EXT EXPONENT
6615 035700 004767 000322              JSR      PC,FLTMPY    ;
6616 035704 174102              STF      AC1,AC2      ;SAVE RESULT IN AC2
6617 035706 013737 001414 001416      MOV      @#SAC1,@#SAC2
6618                                     ;NOW DO THE A*A
6619 035714 172437 001302              LDF      @#STMP0,ACO   ;LOAD OPERAND A
6620 035720 013737 001256 001412      MOV      @#SREG0,@#SAC0
6621 035726 172500              LDF      ACO,AC1
6622 035730 013737 001412 001414      MOV      @#SAC0,@#SAC1
6623 035736 004767 000264              JSR      PC,FLTMPY    ;EXECUTE THE MULTIPLY
6624 035742 013737 001414 001420      MOV      @#SAC1,@#SAC3 ;SAVE EXT EXPO OF A*A
6625                                     ;NOW DO A**2-B**2
6626 035750 172402              LDF      AC2,ACO      ;GET B*B
6627 035752 013737 001416 001412      MOV      @#SAC2,@#SAC0 ;A*A IN AC1
6628 035760 004767 000306              JSR      PC,FLTSUB    ;
6629 035764 174137 001442              STF      AC1,@#FLTMP1 ;SAVE IN MEMORY
6630 035770 013737 001414 001264      MOV      @#SAC1,@#SREG3
6631                                     ;NOW COMPUTE THE RESULTS
6632                                     ;CALCULATE THE NUMBER OF CORRECT BITS
6633 035776 023737 001416 001420      CMP      @#SAC2,@#SAC3 ;DETERMINE WHICH EXP IS LARGER
6634 036004 002003              BGE      2$           ;BRANCH IF AC2 LARGER
6635 036006 013737 001420 001416      MOV      @#SAC3,@#SAC2 ;PUT LARGEST IN AC2
6636 036014 163737 001414 001416      2$: SUB      @#SAC1,@#SAC2
6637 036022 162737 000065 001416      SUB      #53,@#SAC2
6638 036030 005437 001416              NEG      @#SAC2
6639 036034 172437 001432              LDF      @#FLTMP0,ACO ;GET LEFT HAND SIDE
6640 036040 013737 001262 001412      MOV      @#SREG2,@#SAC0
6641 036046 004767 000220              JSR      PC,FLTSUB    ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6642 036052 163737 001264 001414      SUB      @#SREG3,@#SAC1 ;SUB EXT EXPONENTS
6643                                     ;ACTUAL EXPONENTS ARE EQUAL
6644 036060 100002              BPL      1$           ;MAKE SURE RESULT IS POSITIVE
6645 036062 005437 001414              NEG      @#SAC1
6646 036066 023737 001416 001414      1$: CMP      @#SAC2,@#SAC1 ;RESULTS WITHIN RANGE ALLOWED?
6647 036074 003401              BLE      SECT3D      ;BRANCH IF YES
6648 036076 104016              ERROR    16          ;RESULTS WRONG
6649
6650                                     ;*****
6651 036100 172537 001302      SECT3D: LDF      @#STMP0,AC1 ;LOAD OPERAND A
6652 036104 172437 001312              LDF      @#STMP4,ACO   ;AND OPERAND B
6653 036110 013737 001256 001414      MOV      @#SREG0,@#SAC1
6654 036116 013737 001260 001412      MOV      @#SREG1,@#SAC0
6655 036124 004767 000120              JSR      PC,FLTDIV    ;GO DIVIDE THEM

```

```
6656 036130 004767 000072 JSR PC,FLTMPY ;MULTIPLY RESULT BY B
6657 036134 174137 001432 STF AC1,@FLTMP0 ;SAVE RESULT
6658 036140 013737 001414 001262 MOV @SAC1,@SREG2
6659 036146 172437 001302 LDF @STMP0,ACO ;LOAD OPERAND A
6660 036152 174037 001442 STF ACO,@FLTMP1 ;SAVE INCASE TYPE OUT
6661 036156 013737 001256 001412 MOV @SREG0,@SAC0
6662 036164 013737 001256 001264 MOV @SREG0,@SREG3
6663 036172 004767 000074 JSR PC,FLTSUB ;SUBTRACT RIGHT AND LEFT HAND SIDES
6664 036176 163737 001256 001414 SUB @SREG0,@SAC1 ;SEE IF RESULT OK
6665 036204 100002 BPL 1$ ;ENSURE DIFFERENCE IS POSITIVE
6666 036206 005437 001414 NEG @SAC1
6667 036212 022737 000066 001414 1$: CMP #54,@SAC1 ;RESULTS WITHIN 2 BITS?
6668 036220 003505 BLE RELE8 ;BRANCH IF YES
6669 036222 104016 ERROR 16 ;RESULTS WRONG
6670 036224 000503 BR RELE8
6671
6672 ;*****
6673 ;SBTTL FLOATING POINT MULTIPLY ROUTINE
6674 ;* THIS ROUTINE MULTIPLIES THE CONTENTS OF ACO AND AC1
6675 ;* AND LEAVES THE RESULT IN AC1. IT ALSO TAKES CARE OF
6676 ;* THE SOFTWARE EXPONENTS THAT ARE KEPT IN SAC0 AND SAC1.
6677 ;*****
6678 036226 063737 001412 001414 FLTMPY: ADD @SAC0,@SAC1 ;ADD SOFTWARE EXPONENTS
6679 036234 171100 MULF ACO,AC1 ;DO THE MULTIPLY
6680 036236 012746 100400 MOV #100400,-(SP) ;PUT CONTROL WORD ON STACK
6681 036242 004737 060674 JSR PC,@EXPEXT ;CALCULATE EXT EXPONENT
6682 036246 000207 1$: RTS PC ;RETURN
6683
6684 ;*****
6685 ;SBTTL FLOATING POINT DIVIDE ROUTINE
6686 ;* THIS ROUTINE DIVIDES THE CONTENTS OF AC1 BY ACO
6687 ;* AND LEAVES THE RESULT IN AC1.
6688 ;*****
6689 036250 163737 001412 001414 FLTDIV: SUB @SAC0,@SAC1 ;ADJUST SOFTWARE EXPONENTS
6690 036256 174500 DIVF ACO,AC1 ;EXECUTE THE DIVIDE
6691 036260 012746 100400 MOV #100400,-(SP) ;PUT CONTROL WORD ON STACK
6692 036264 004737 060674 JSR PC,@EXPEXT ;CALCULATE EXT EXPONENT
6693 036270 000207 1$: RTS PC ;RETURN
6694
6695 ;*****
6696 ;SBTTL FLOATING POINT ADD ROUTINE
6697 ;* THIS ROUTINE ADDS THE CONTENTS OF ACO TO AC1.
6698 ;* THIS CAN ONLY BE DONE IF THE SOFTWARE EXPONENTS
6699 ;* ARE CLOSE ENOUGH TOGETHER SUCH THAT AN ADJUSTMENT
6700 ;* OF THE REAL EXPONENT LEAVES A NON-ZERO NUMBER.
6701 ;*****
6702 036272 010667 000134 FLTSUB: MOV SP,SUBFLG ;SET SUBTRACT FLAG
6703 036276 023737 001412 001414 FLTADD: CMP @SAC0,@SAC1 ;CHECK SOFTWARE EXPONENTS
6704 036304 003016 BGT 1$
6705 036306 001434 BEQ 2$
6706 ;ACCUMULATOR 1 IS LARGER THAN ACCUMULATOR 0
6707 036310 013702 001414 MOV @SAC1,R2 ;GET OPERAND B SOFTWARE EXP
6708 036314 163702 001412 SUB @SAC0,R2 ;GET DIFFERENCE IN SOFTWARE EXP'S
6709 036320 020227 000071 CMP R2,#57. ;EXP WITHIN DBL PREC RANGE?
6710 036324 002003 BGE 7$ ;BRANCH IF ADD NOT REQUIRED
6711 ;RESULT IS OPERAND B
```

6712 036326 005402
6713 036330 176402
6714 036332 000422
6715 036334 176427 177703
6716 036340 000417
6717
6718
6719 036342 013702 001412
6720 036346 163702 001414
6721 036352 013737 001412 001414
6722 036360 020227 000071
6723 036364 002003
6724 036366 005402
6725 036370 176502
6726 036372 000402
6727
6728
6729 036374 176527 177703
6730
6731 036400 005767 000026
6732 036404 001402
6733 036406 173100
6734 036410 000401
6735 036412 172100
6736 036414 012746 100400
6737 036420 004737 060674
6738 036424 005067 000002
6739 036430 000207
6740 036432 000000
6741 036434 000004
6742 036436 010702
6743 036440 062702 000012
6744 036444 012707 043764
6745 036450 000000
6746
6747
6748
6749
6750 036452 005737 001502
6751 036456 001002
6752 036460 000167 004060
6753 036464
6754
6755
6756
6757
6758
6759
6760
6761
6762
6763
6764 036464 012767 000001 142654
6765 036472 000004
6766 036474 112737 000074 001202
6767 036502 013737 001202 177570

NEG R2
LDEXP R2,AC0 ;RELOAD THE EXPONENT
BR 2\$
7\$: LDEXP #-75,AC0 ;FAKE EXPONENT SO HARDWARE
BR 2\$;WILL DETECT OUT OF RANGE
:
:ACCUMULATOR 0 IS LARGER THAN ACCUMULATOR 1
1\$: MOV @#SAC0,R2 ;GET SOFTWARE EXP OF OPERAND A
SUB @#SAC1,R2 ;GET DIFFERENCE IN EXP'S
MOV @#SAC0,@#SAC1 ;MAKE SOFTWARE EXP'S EQUAL
CMP R2,#57. ;EXP WITHIN DBL PREC RANGE?
BGE 4\$;BRANCH IF NO
NEG R2
LDEXP R2,AC1 ;RELOAD THE EXPONENT
BR 2\$
:
:ACCUMULATOR 0 IS MUCH LARGER THAN ACCUMULATOR 1 SO RESULT IS 0
4\$: LDEXP #-75,AC1 ;FAKE EXPONENT SO HARDWARE
;WILL DETECT OUT OF RANGE
2\$: TST SUBFLG ;ADD OR SUBTRACT?
BEQ 5\$;BRANCH IF ADD
SUBF AC0,AC1
BR 6\$
5\$: ADDF AC0,AC1 ;EXECUTE THE ADD
6\$: MOV #100400,-(SP) ;PUT CONTROL WORD ON STACK
JSR PC,@#EXPEXT ;CALCULATE EXT EXPONENT
3\$: CLR SUBFLG ;INIT SUBTRACT FLAG
RTS PC ;RETURN
SUBFLG: .WORD
RELE8: SLOPE
MOV PC,R2
ADD #12,R2
MOV #RELOC,PC ;GO RELOCATE PROGRAM CODE
REL88: .WORD 0
;8888888888888888 LAST ADDRESS OF CODE TO BE RELOCATED 8888888888
:
TST @#KB11E ;IS THIS A KB11-E OR KB11-EM?
BNE DOMFPT ;BR IF IT IS
JMP ENDCIS ;SKIP THIS TEST IF NOT. MFPT AND CIS NOT THERE
DOMFPT:
:*****
: *TEST 74 CHECK MFPT INSTRUCTION (KB11-E/EM ONLY)
: * THE MFPT INSTRUCTION IS NOT AVAILABLE ON THE KB11-B/C/CM BUT
: * IF THIS IS A KB11-E/EM THIS TEST IS RUN. MFPT RETURNS
: * DATA TO R0 IN THE FOLLOWING FORMAT:
: * BIT 0 - 1 INDICATES 11/44 CPU
: * BIT 1 - 1 INDICATES KB11-E/EM CPU (SHOULD ALWAYS COME UP IN THIS TEST)
: * BIT 8 - 1 INDICATES CISP PRESENT
: * BIT 9 - 1 INDICATES FP PRESENT
:*****
MOV #1,\$TIMES ;:DO 1 ITERATION
TST74: SCOPE
MOVB #74,@#STSTNM ;LOAD TEST NUMBER
MOV @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER

6768
6769
6770
6771 036510 010700
6772 036512 005740
6773 036514 010037 001540
6774 036520 010700
6775 036522 162700 036522
6776 036526 010037 001534
6777 036532 010737 001212
6778 036536 062737 000030 001212
6779 036544 013737 001212 001210
6780 036552 105737 001530
6781 036556 001402
6782 036560 000167 003742
6783 036564 012703 000002
6784 036570 105737 001505
6785 036574 001402
6786 036576 052703 000400
6787 036602 032737 020000 001500 1\$:
6788 036610 001402
6789 036612 052703 001000
6790 036616 000007 2\$:
6791 036620 020003
6792 036622 001405
6793 036624 010337 001302
6794 036630 010037 001304
6795 036634 104017
6796 036636 105737 001505
6797 036642 0C1002
6798 036644 000167 003656
6799 036650
6800
6801
6802
6803 036650 000004
6804 036652 112737 000075 001202
6805 036660 013737 001202 177570
6806 036666 013767 001550 002272
6807 036674 013767 001552 002266
6808 036702 062737 000036 001212
6809
6810 036710 016737 002252 001550
6811 036716 016737 002246 001552
6812 036724 012700 041266
6813 036730 063700 001534
6814 036734 012001 1\$:
6815 036736 001407
6816 036740 010102
6817 036742 063702 001534
6818 036746 011112
6819 036750 063712 001534
6820 036754 000767
6821
6822 036756 016700 002302 2\$:
6823 036762 016701 002276

SBTTL START OF SECTION 9
:9999999999999999 FIRST ADDRESS TO BE RELOCATED 9999999999
REL9: MOV PC,R0 ;GET PC
TST -(R0) ;R0 CONTAINS THE ADDRESS OF REL9
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 RELE9
MOV #2,R3 ;R3 IS DATA PATTERN. BIT 1 WILL ALWAYS BE SET
TSTB @#CISP ;CISP FOUND?
BEQ 1\$;BR IF NOT
BIS #400,R3 ;BIT 8 SHOULD BE SET FOR CISP
BIT #20000,@#OPT.CP ;FP FOUND?
BEQ 2\$;BR IF NOT
BIS #1000,R3 ;BIT 9 WILL BE SET FOR FP
MFPT ;EXECUTE INSTRUCTION
CMP R0,R3 ;MATCH?
BEQ DONE7 ;DONE IF SO
MOV R3,@#STMPC ;SET UP EXPECTED (GOOD) DATA
MOV R0,@#STMP1 ;SET UP RECEIVED (BAD) DATA
ERROR 17 ;ERROR PRINTOUT
DONE7: TSTB @#CISP ;IS CISP PRESENT?
BNE DOCIS ;BR IF IT IS
JMP RELE9 ;SKIP CIS TEST BUT RELOCATE SECTION FOR MFPT
DOCIS:
:*****
:*TEST 75 COMMERCIAL INSTRUCTION SET TEST
:*****
TST75: SCOPE
MOVB #75,@#STSTNM ;LOAD TEST NUMBER
MOV @#STSTNM,@#DISPLAY ;DISPLAY TEST NUMBER
MOV @#SLONUM,SAVRNL ;SAVE SEED LOW
MOV @#SHINUM,SAVRNH ;SAVE SEED HIGH
ADD #30,@#SLPERR ;ADJUST LOOP ON ERROR POINTER
;START OF TEST TO MODIFY ABSOLUTE ADDRESSES WHEN RELOCATING
MOV SAVRNL,@#SLONUM ;RESTORE SEED LO
MOV SAVRNL,@#SHINUM ;RESTORE SEED HI
MOV #OFFTAB,R0 ;GET OFFSET TO OFFSET TABLE
ADD @#FACTOR,R0 ;ADD FACTOR TO GET ADDRESS OF TABLE
1\$: MOV (R0)+,R1 ;GET OFFSET OF STRING
BEQ 3\$;DONE IF OFFSET = 0
MOV R1,R2 ;SAVE R1
ADD @#FACTOR,R2 ;ADD FACTOR TO OFFSET
MOV (R1),(R2) ;MOVE ORIGINAL DATA TO ADDRESS
ADD @#FACTOR,(R2) ;ADD FACTOR FOR ADDRESS
BR 1\$;GO MODIFY NEXT ADDRESS
;SET PASS COUNT AND GET RANDOM DATA
2\$: MOV BUFFAD,R0 ;SET SRC ADDRESS FOR DATA
MOV BUFFAD,R1 ;SET DST ADDRESS FOR DATA

Address	OpCode	Source	Destination	Label	Instruction	Comment
6824	036766	062701	000310		ADD #200,R1	:ADJUST
6825	036772	004737	060436	4S:	JSR PC,2*SRAND	:GET RANDOM NUMBER
6826	036776	013710	001550		MOV 2*SLONUM,(R0)	:STORE NUMBER IN SOURCE ONE
6827	037002	042710	100200		BIC #100200,(R0)	:MAKE NUMBER BETWEEN 0 AND 177
6828	037006	012021			MOV (R0)+,(R1)+	:STORE NUMBER IN DST FOR TEST (MPC)
6829	037010	020067	002164		CMF R0,DST.1A	:DONE FILLING SOURCE ONE YET
6830	037014	002766			BLT 4S	:NO GET NEXT RANDOM NUMBER
6831						:YES GO TO FIRST TEST
6832						:TEST MPC INSTRUCTION COMPARE SRC 1 TO SRC 1
6833	037016	004767	001560	COMP:	JSR PC,SETUP	:SET UP DESCRIPTORS
6834	037022	076144			CMPCI	:COMPARE STRINGS
6835	037024	041172		SRC1:	.WORD SRC.1D	:SOURCE ONE DESCRIPTOR
6836	037026	041176		DST1:	.WORD DST.1D	:DST DESCRIPTOR
6837	037030	000040			.WORD	:FILL WITH SPACES
6838	037032	001403			BEQ MOVE	:NO ERROR GO TO NEXT TEST
6839	037034	004767	001602		JSR PC,CISER	:GET ERROR DATA
6840	037040	104020			ERROR 20	:REPORT ERROR
6841					:MOVE CHAR STRING	
6842	037042	004767	001534	MOVE:	JSR PC,SETUP	:SET STRING DESCRIPTORS
6843	037046	076067			L3D7	:LOAD DESC INTO REG
6844	037050	041172		SRC2:	.WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6845	037052	041176		DST2:	.WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6846	037054	041254		CHAR1:	.WORD CHAR	
6847	037056	076030			MOVC	:MOVE STRING
6848	037060	076144			CMPCI	:COMPARE SRC AND DST
6849	037062	041172		SRC3:	.WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6850	037064	041176		DST3:	.WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6851	037066	000040			.WORD	:FILL WITH SPACES
6852	037070	001403			BEQ SCAN	:IF EQUAL NEXT TEST
6853	037072	004767	001544		JSR PC,CISER	:GET ERROR DATA
6854	037076	104020			ERROR 20	:REPORT ERROR
6855					:SCAN,MOVC	
6856	037100	004767	001476	SCAN:	JSR PC,SETUP	:SET UP DESCRIPTORS
6857	037104	112767	000001 002124		MOVB #1,SET.1D	:SET CHAR MASK FOR SPAN AND SCAN
6858	037112	076142		NXSCAN:	SCANCI	:SCAN
6859	037114	041172		SRC4:	.WORD SRC.1D	:SOURCE DESC
6860	037116	041236		SET1:	.WORD SET.1D	:PTR TO CHAR SET DESC
6861	037120	001003			BNE FNDSN	:CHAR FOUND MOVE STRING
6862	037122	106367	002110		ASLB SET.1D	:NOT FOUND SHIFT MASK
6863	037126	000771			BR NXSCAN	:LOOK AGAIN
6864	037130	010067	002036	FNDSN:	MOV R0,SRC.1D	:MOV NEW ADDRESS TO DESC
6865	037134	010167	002034		MOV R1,SRC.1A	:MOV NEW LENGTH TO DESC
6866	037140	076130			MOVCI	:MOV TEXT STARTING WITH CHAR FOUND
6867	037142	041172		SRC5:	.WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6868	037144	041176		DST4:	.WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6869	037146	000040			.WORD	:FILL WITH SPACES
6870	037150	076144			CMPCI	:COMPARE SRC AND DST
6871	037152	041172		SRC6:	.WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6872	037154	041176		DST5:	.WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6873	037156	000040			.WORD	:FILL WITH SPACES
6874	037160	001403			BEQ SPAN	:STRINGS EQUAL NEXT TEST
6875	037162	004767	001454		JSR PC,CISER	:NOT EQUAL GET ERROR DATA
6876	037166	104020			ERROR 20	:REPORT ERROR
6877					:SPAN AND MOVC	
6878	037170	004767	001406	SPAN:	JSR PC,SETUP	:SETUP DESC
6879	037174	012767	000001 002034		MOV #1,SET.1D	:SET MASK

6880	037202	076067				NXSPAN:	L3D7			:LOAD DESC
6881	037204	041172				SRC7:	.WORD	SRC.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6882	037206	041176				DST6:	.WORD	DST.1D		:DUMMY DESC TO GET SET.1D TO R4
6883	037210	041236				SET2:	.WORD	SET.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6884	037212	076043					SPANC			:FIND CHAR
6885	037214	001003					BNE	FNDSPN		:FOUND MOVE STRING
6886	037216	106367	002014				ASLB	SET.1D		:NOT SHIFT MASK
6887	037222	000767					BR	NXSPAN		:AND LOOK AGAIN
6888	037224	010067	001742			FNDSPN:	MOV	R0,SRC.1D		:GET ADDRESS
6889	037230	010167	001740				MOV	R1,SRC.1A		:SET LENGTH
6890	037234	076067					L3D7			:LOAD DESC
6891	037236	041172				SRC8:	.WORD	SRC.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6892	037240	041176				DST7:	.WORD	DST.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6893	037242	041254				CHAR2:	.WORD	CHAR		
6894	037244	076030					MOVC			:MOVE STING BEGINNING WITH CHAR
6895	037246	076067					L3D7			:LOAD DESC
6896	037250	041172				SRC9:	.WORD	SRC.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6897	037252	041176				DST8:	.WORD	DST.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6898	037254	041254				CHAR3:	.WORD	CHAR		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6899	037256	076044					CMPC			:COMPARE SRC AND DST
6900	037260	001403					BEQ	MATCH		:STRINGS EQUAL NEXT TEST
6901	037262	004767	001354				JSR	PC,CISER		:NOT EQUAL GET ERROR DATA
6902	037266	104020					ERROR	20		:REPORT ERROR
6903										:MATCH AND MOVE REVERSE
6904	037270	004767	001306			MATCH:	JSR	PC,SETUP		:SET UP DESC
6905	037274	076027					L2D7			:LOAD DESC INTO REGISTERS
6906	037276	041172				SRC10:	.WORD	SRC.1D		:SOURCE POINTER
6907	037300	041202				OBJ1:	.WORD	OBJ.1D		:OBJECT POINTER
6908	037302	076045					MATC			:MATCH STRINGS
6909	037304	010167	001664				MOV	R1,SRC.1A		:GET NEW SRC ADDRESS
6910	037310	012767	000031 001654				MOV	#*D25,SRC.1D		:GET NEW SRC LENGTH
6911	037316	076067					L3D7			:LOAD DESCRIPTORS
6912	037320	041172				SRC11:	.WORD	SRC.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6913	037322	041202				DST9:	.WORD	OBJ.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6914	037324	041254				CHAR4:	.WORD	CHAR		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6915	037326	076044					CMPC			:COMPARE RESULTS
6916	037330	001403					BEQ	MOVER		:FOUND NEXT TEST
6917	037332	004767	001304				JSR	PC,CISER		:ERROR DATA
6918	037336	104020					ERROR	20		:REPORT ERROR
6919										:MOVE REVERSE
6920	037340	004767	001236			MOVER:	JSR	PC,SETUP		:SET UP DESC
6921	037344	076131					MOVRCI			:MOVE REVERSE
6922	037346	041172				SRC12:	.WORD	SRC.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6923	037350	041176				DST10:	.WORD	DST.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6924	037352	000040					.WORD	'		:FILL WITH SPACES
6925	037354	076067					L3D7			:LOAD DESCRIPTORS
6926	037356	041176				DST11:	.WORD	DST.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6927	037360	041176				DST12:	.WORD	DST.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6928	037362	000040					.WORD	'		
6929	037364	076031					MOVRC			:MOVE REVERSE AGAIN
6930	037366	076067					L3D7			:LOAD DESC FORCOMPARE
6931	037370	041172				SRC13:	.WORD	SRC.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6932	037372	041176				DST13:	.WORD	DST.1D		:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6933	037374	000040					.WORD	'		:FILL WITH SPACES
6934	037376	076044					CMPC			:COMPARE STRINGS
6935	037400	001403					BEQ	MOVT		:EQUAL NEXT TEST

Address	OpCode	OpCode	OpCode	Instruction	Comment
6936	037402	004767	001234	JSR PC,CISER	:GET ERROR DATA
6937	037406	104020		ERROR 20	:REPORT ERROR
6938				:MOVE TRANSLATE	
6939	037410	004767	001166	MOVT: JSR PC,SETUP	:SET UP DESC
6940	037414	076132		MOVT:CI	:MOVE TRANSLATE
6941	037416	041172		SRC14: .WORD SRC.1D	:SRC DESC PTR
6942	037420	041176		DST14: .WORD DST.1D	:DEST DESC PTR
6943	037422	000040		.WORD	:FILL WITH SPACES
6944	037424	040732		TRANS1: .WORD TRANS	:TRANSLATE TABLE ADDRESS
6945	037426	076132		MOVT:CI	:MOVE TRANS AGAIN
6946	037430	041176		DST15: .WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6947	037432	041176		DST16: .WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6948	037434	000040		.WORD	:FILL WITH SPACES
6949	037436	040732		TRANS2: .WORD TRANS	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6950	037440	076144		CMPCI	:COMPARE SRC AND DST
6951	037442	041172		SRC15: .WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6952	037444	041176		DST17: .WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6953	037446	000040		.WORD	:FILL WITH SPACES
6954	037450	001403		BEQ LOCATE	:STRINGS EQUAL NEXT TST
6955	037452	004767	001164	ERR4: JSR PC,CISER	:GET ERROR DATA
6956	037456	104020		ERROR 20	:REPORT ERROR
6957				:LOCATE AND MOVE CHARACTER	
6958	037460	004767	001116	LOCATE: JSR PC,SETUP	:SETUP DESCRIPTORS
6959	037464	076140		NXLOC: LOCCI	:LOCATE CHARACTER
6960	037466	041172		SRC16: .WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6961	037470	000040		LOCCHR: .WORD	
6962	037472	001003		BNE FNDLOC	:FOUND MOVE STRING
6963	037474	105267	177770	INCB LOCCHR	:NOT FOUND INC CHAR FOR SEARCH
6964	037500	000771		BR NXLOC	:LOOK AGAIN FOR NEW CHAR
6965	037502	010067	001464	FNDLOC: MOV R0,SRC.1D	:ADDRESS OF CHAR FOUND TO SCR.1
6966	037506	010167	001462	MOV R1,SRC.1A	:LENGTH OF STRING
6967	037512	016703	001462	MOV DST.1A,R3	:MOVE DST ADDRESS TO R2
6968	037516	016702	001454	MOV DST.1D,R2	:MOVE STRING LENGTH TO R3
6969	037522	012704	000040	MOV # ,R4	:FILL CHAR
6970	037526	076030		MOV:CI	:MOVE STRING BEGINING WITH CHAR FOUND
6971	037530	076144		CMPCI	:COMPARE SOURCE AND DEST
6972	037532	041172		SRC17: .WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6973	037534	041176		DST18: .WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6974	037536	000040		.WORD	:FILL WITH SPACES
6975	037540	001403		BEQ SKIP	:STRINGS EQUAL NEXT TEST
6976	037542	004767	001074	JSR PC,CISER	:NOT EQUAL ERROR
6977	037546	104020		ERROR 20	:REPORT ERROR
6978				:SKIP AND MOVE CHAR STRING	
6979	037550	004767	001026	SKIP: JSR PC,SETUP	:SETUP DESCRIPTORS
6980	037554	076141		NXSKIP: SKPCI	:SKIP CHAR
6981	037556	041172		SRC18: .WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6982	037560	000040		SKPCHR: .WORD	
6983	037562	001003		BNE FNDSKP	:CHAR FOUND GO MOVE STRING
6984	037564	005267	177770	INC SKPCHR	:NOT FOUND INC CHAR
6985	037570	000771		BR NXSKIP	:LOOK AGAIN
6986	037572	010067	001374	FNDSKP: MOV R0,SRC.1D	:GET NEW SRC ADDRESS
6987	037576	010167	001372	MOV R1,SRC.1A	:NEW SOURCE LENGTH
6988	037602	076130		MOV:CI	:MOVE STRING
6989	037604	041172		SRC19: .WORD SRC.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6990	037606	041176		DST19: .WORD DST.1D	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
6991	037610	000040		.WORD	:FILL WITH SPACES

6992 037612 076067
6993 037614 041172
6994 037616 041176
6995 037620 041254
6996 037622 076044
6997 037624 001403
6998 037626 004767 001010
6999 037632 104020
7000
7001
7002 037634 016700 001424
7003 037640 016705 001420
7004 037644 062705 000070
7005 037650 005002
7006 037652 012703 041132
7007 037656 063703 001534
7008 037662 011001
7009 037664 005004
7010 037666 042301
7011 037670 022301
7012 037672 002001
7013 037674 161301
7014 037676 005723
7015 037700 050102
7016 037702 005204
7017 037704 022704 000004
7018 037710 002366
7019 037712 010220
7020 037714 020500
7021 037716 103354
7022 037720 012701 041206
7023 037724 063701 001534
7024 037730 012702 041232
7025 037734 063702 001534
7026 037740 042711 070000
7027 037744 052711 010000
7028 037750 062701 000004
7029 037754 020102
7030 037756 103770
7031
7032 037760 076152
7033 037762 041206
7034 037764 041206
7035 037766 001403
7036 037770 004767 000720
7037 037774 104020
7038
7039 037776 076156
7040 040000 041206
7041 040002 041232
7042 040004 000001
7043 040006 076156
7044 040010 041212
7045 040012 041222
7046 040014 000001
7047 040016 076156

L3D7
SRC20: .WORD SRC.1D
DST20: .WORD DST.1D
CHAR5: .WORD CHAR
CMPC
BEQ DECDAT
JSR PC,CISER
ERROR 20
;DECIMAL ARITHMETIC TESTS
;SETUP DECIMAL DATA
DECDAT: MOV BUFFAD,R0
MOV BUFFAD,R5
ADD #56.,R5
1\$: CLR R2
MOV #MSKTAB,R3
ADD @#FACTOR,R3
MOV (R0),R1
CLR R4
2\$: BIC (R3)+,R1
CMP (R3)+,R1
BGE 3\$
SUB (R3),R1
3\$: TST (R3)+
BIS R1,R2
INC R4
CMP #4,R4
BGE 2\$
MOV R2,(R0)+
CMP R5,R0
BHS 1\$
MOV #A.DSC,R1
ADD @#FACTOR,R1
MOV #D.DSC,R2
ADD @#FACTOR,R2
4\$: BIC #070000,(R1)
BIS #10000,(R1)
ADD #4,R1
CMP R1,R2
BLO 4\$
;TEST COMPARE NUMERIC
CMPNI
A1: .WORD A.DSC
A2: .WORD A.DSC
BEQ NUMRIC
JSR PC,CISER3
ERROR 20
;CALCULATE [(10A+10B)-10C]
NUMRIC: ASHNI
A3: .WORD A.DSC
D1: .WORD D.DSC
.WORD 1
ASHNI
B1: .WORD B.DSC
E1: .WORD E.DSC
.WORD 1
ADDNI

;LOAD DESCRIPTORS FOR COMPARE
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;COMPARE STRINGS
;EQUAL NEXT TEST
;NOT EQUAL GET ERROR DATA
;REPORT ERROR
;GET BUFFAD TO R0 FOR INDEX
;CLR R2 TO ACCUMULATE NIBBLES
;GET OFFSET OF MSKTAB
;ADJUST ADDRESS
;LOAD R1
;CLR FOR COUNTER
;CLEAR OFF UNDESIED NIBBLES
;IS NIBBLE LESS THAN 9
;YES DONT SUBTRACT
;GREATER THAN 9 SUB 6
;INC R3 TWICE IF NO SUB
;STORE NIBBLE IN R2
;INC NIBBLE COUNT
;4 NIBBLES DONE YET
;NO DO AGAIN
;STORE VALID DATA IN SOURCE
;NO DO AGAIN
;SET DATA TYPE
;ADD FACTOR FOR ADDRESS
;GET OFFSET OF D.DSC
;ADJUST TO GET ADDRESS
;CLEAR TYPE BITS
;MAKE UNSIGNED ZONED DATA
;GET NEXT DATA TYPE SPECIFIER
;TEST FOR DONE
;NOT DONE DO AGAIN
;COMPARE EQUAL STRINGS
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;EQUAL NEXT TEST
;GET ERROR DATA
;REPORT ERROR
;SHIFT A
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;SHIFT COUNT
;SHIFT B
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;THIS LOCATION MODIFIED WHEN TEST RELOCATES
;SHIFT COUNT
;10A+10B

Address	Offset	Value	Label	Instruction	Comment
7048	040020	041232	D2:	.WORD D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7049	040022	041222	E2:	.WORD E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7050	040024	041226	F1:	.WORD F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7051	040026	076156		ASHNI	:SHIFT C
7052	040030	041216	C1:	.WORD C.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7053	040032	041232	D3:	.WORD D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7054	040034	000001		.WORD 1	:SHIFT COUNT
7055	040036	076151		SUBNI	:10A+10B-10C
7056	040040	041232	F2:	.WORD D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7057	040042	041226	D4:	.WORD F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7058	040044	041222	E3:	.WORD E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7059				:CALCULATE 10*[(A-C)+B]	
7060	040046	076067		L3D7	:LOAD DESCRIPTORS
7061	040050	041206	A4:	.WORD A.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7062	040052	041216	C2:	.WORD C.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7063	040054	041226	F3:	.WORD F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7064	040056	076051		SUBN	:SUB A-C
7065	040060	076067		L3D7	:LOAD DESC
7066	040062	041212	B2:	.WORD B.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7067	040064	041226	F4:	.WORD F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7068	040066	041232	D5:	.WORD D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7069	040070	076050		ADDN	:ADD A-C+B
7070	040072	076067		L3D7	:LOAD DESC
7071	040074	041232	D6:	.WORD D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7072	040076	041226	F5:	.WORD F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7073	040100	041252	ONE1:	.WORD ONE	
7074	040102	076056		ASHN	:MULT BY 10
7075				:COMPARE RESULTS	
7076	040104	076077		L2D7	:LOAD DESC
7077	040106	041222	E4:	.WORD E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7078	040110	041226	F6:	.WORD F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7079	040112	076052		CMPI	:COMPARE STRINGS
7080	040114	001403		BEQ CONNUM	:NEXT TEST IF EQUAL
7081	040116	004767	000554	JSR PC,CISER2	:GET ERROR DATA
7082	040122	104020		ERROR 20	:REPORT ERROR
7083				:CONVERT DATA TYPES	
7084				:LONG -> NUMERIC -> LONG	
7085				:NUMERIC -> PACKED -> NUMERIC	
7086	040124	076157	CONNUM:	(VTLNI	:CONVERT LONG TO NUMERIC
7087	040126	041222	E5:	.WORD E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7088	040130	041242	LONG1:	.WORD LONG.1	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7089	040132	076153		(VTNLI	:CONVERT NUMERIC TO LONG
7090	040134	041222	E18:	.WORD E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7091	040136	041246	LONG4:	.WORD LONG.2	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7092	040140	026767	001076 001100	CMPI LONG.1, LONG.2	:CHECK FIRST HALF #LONG WORD
7093	040146	001004		BNE 1\$:NOT EQUAL ERROR
7094	040150	026767	001070 001072	CMPI LONG.1+2, LONG.2+2	:EQUAL CHECK SECOND HALF
7095	040156	001403		BEQ NUMPAC	:EQUAL NEXT TEST
7096	040160	004767	000474	JSR PC,CISER1	:GET ERROR DATA
7097	040164	104020		ERROR 20	:REPORT ERROR
7098				:CONVERT NUM TO PACK TO NUM	
7099	040166	076155	NUMPAC:	(VTNPI	:CONVERT NUM TO PACKED
7100	040170	041206	A5:	.WORD A.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7101	040172	041222	E6:	.WORD E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7102	040174	076154		(VTPNI	:CONVERT BACK TO NUM
7103	040176	041222	E7:	.WORD E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES

7104 040200 041226
7105 040202 076156
7106 040204 041206
7107 040206 041222
7108 040210 000000
7109 040212 076152
7110 040214 041222
7111 040216 041226
7112 040220 001403
7113 040222 004767 000450
7114 040226 104020
7115
7116 040230 012701 041206
7117 040234 063701 001534
7118 040240 012702 041232
7119 040244 063702 001534

F7: .WORD F.DSC
ASHNI
A6: .WORD A.DSC
E8: .WORD E.DSC
.WORD 0
CMPNI
E9: .WORD E.DSC
F8: .WORD F.DSC
BEQ PACDAT
JSR PC,CISER2
ERROR 20
:PACKED DECIMAL ARITHMETIC
PACDAT: MOV #A.DSC,R1
ADD @WFACTOR,R1
MOV #D.DSC,R2
ADD @WFACTOR,R2

:THIS LOCATION MODIFIED WHEN TEST RELOCATES
:TRANSFER A RO E 32 BYTES
:THIS LOCATION MODIFIED WHEN TEST RELOCATES
:THIS LOCATION MODIFIED WHEN TEST RELOCATES
:SHIFT COUNT
:COMPARE RESULTS
:THIS LOCATION MODIFIED WHEN TEST RELOCATES
:THIS LOCATION MODIFIED WHEN TEST RELOCATES
:EQUAL NEXT TEST
:GO GET ERROR DATA
:REPORT ERROR
:SET DATA TYPE
:ADJUST FOR ADDRESS
:GET OFFSET TO D.DSC
:ADJUST FOR ADDRESS

7120	040250	042711	070000	18:	BIC	#070000,(R1)	:MAKE UNSIGNED PACKED DATA
7121	040254	052711	060000		BIS	#060000,(R1)	:SET TYPE BITS
7122	040260	062701	000004		ADD	#4,R1	:NEXT DATA TYPE SPEC
7123	040264	020102			CMP	R1,R2	:DONE YET
7124	040266	101770			BLOS	18	:NO DO AGAIN
7125	040270	146777	000666 000712		BICB	HIMASK,2A	:CLR HI NIBBLE TO MAKE VALID PACKED STRING
7126	040276	146777	000660 000710		BICB	HIMASK,2B	:CLR HI NIB OF B
7127	040304	146777	000652 000706		BICB	HIMASK,2C	:CLR HI NIB OF C
7128	040312	016700	000746		MOV	BUFFAD,R0	:GET ADDRESS OF BUFF
7129	040316	146760	000642 000016		BICB	LOMASK,14,(R0)	:CLEAR SIGN NIBBLE
7130	040324	156760	000633 000016		BISB	SIGN,14,(R0)	:SET SIGN NIBBLE OF A
7131	040332	146760	000626 000043		BICB	LOMASK,35,(R0)	:CLEAR SIGN NIBBLE
7132	040340	156760	000617 000043		BISB	SIGN,35,(R0)	:SET SIGN
7133	040346	146760	000612 000057		BICB	LOMASK,47,(R0)	:CLEAR SIGN NIBBLE
7134	040354	156760	000603 000057		BISB	SIGN,47,(R0)	:SET SIGN
7135							:TEST COMPARE PACKED
7136	040362	076172		CMPPAK:	CMPPI		:COMPARE EQUAL STRINGS
7137	040364	041206		A7:	.WORD	A.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7138	040366	041206		A8:	.WORD	A.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7139	040370	001401			BEQ	PACKED	:EQUAL GO TEST
7140	040372	104020			ERROR	20	:REPORT ERROR
7141							:CALCULATE 10*[(B+C)]-10*[(B**2)-(C**2)/(B-C)]
7142	040374	076174		PACKED:	MULPI		:MULT A*A
7143	040376	041212		B3:	.WORD	B.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7144	040400	041212		B4:	.WORD	B.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7145	040402	041232		E10:	.WORD	D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7146	040404	076174			MULPI		:MULT B*B
7147	040406	041216		C3:	.WORD	C.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7148	040410	041216		C4:	.WORD	C.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7149	040412	041222		F9:	.WORD	E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7150	040414	076171			SUBPI		:SUB E-F
7151	040416	041216		F10:	.WORD	C.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7152	040420	041212		E11:	.WORD	B.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7153	040422	041226		E12:	.WORD	F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7154	040424	001005			BNE	NOZERO	:BRANCH IF RESULT NOT ZERO
7155	040426	076176			ASHPI		
7156	040430	041216		C7:	.WORD	C.DSC	
7157	040432	041212		B7:	.WORD	B.DSC	
7158	040434	000001			.WORD	1	
7159	040436	000756			BR	PACKED	:DO ANOTHER CALCULATION TO GET RID OF ZERO
7160	040440	076171		NOZERO:	SUBPI		:SUB A-B
7161	040442	041222		C5:	.WORD	E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7162	040444	041232		B5:	.WORD	D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7163	040446	041222		F11:	.WORD	E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7164	040450	076067			L3D7		
7165	040452	041232		D8:	.WORD	D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7166	040454	041222		E14:	.WORD	E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7167	040456	041252		ONE3:	.WORD	ONE	:SHIFT COUNT
7168	040460	076175			DIVPI		:DIVIDE E/F
7169	040462	041226		F12:	.WORD	F.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7170	040464	041222		E13:	.WORD	E.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7171	040466	041232		D7:	.WORD	D.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES
7172	040470	076076			ASHP		:SHIFT 10*SCRCH
7173							:CALCULATE 10*(A+B) REGISTER MODE
7174	040472	076067			L3D7		:LOAD DESCRIPTORS
7175	040474	041212		B6:	.WORD	B.DSC	:THIS LOCATION MODIFIED WHEN TEST RELOCATES

```

7176 040476 041216 C6: .WORD C.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7177 040500 041232 D9: .WORD D.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7178 040502 076070 ADDP ;ADD A+B
7179 040504 076067 L3D7 ;LOAD DESCRIPTORS
7180 040506 041232 D10: .WORD D.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7181 040510 041226 F13: .WORD F.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7182 040512 041252 ONE2: .WORD ONE
7183 040514 076076 ASHP ;SHIFT 10*SCRATCH
7184 ;COMPARE RESULTS
7185 040516 076172 CMPP1 ;E=F ???
7186 040520 041222 E15: .WORD E.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7187 040522 041226 F14: .WORD F.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7188 040524 001403 BEQ CONPAK
7189 040526 004767 000110 JSR PC,CISER ;GET ERROR DATA
7190 040532 104020 ERROR 20 ;REPORT ERROR
7191 ;CONVERT DATA TYPES
7192 ;LONG -> PACKED -> LONG
7193 ;LONG -> NUMERIC -> NUMERIC
7194 040534 076177 CONPAK: CVTLPI ;CONVERT LONG TO PACKED
7195 040536 041222 E16: .WORD E.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7196 040540 041242 LONG2: .WORD LONG.1 ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7197 040542 076173 CVTLPI ;CONVERT PACKED TO LONG
7198 040544 041222 E17: .WORD E.DSC ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7199 040546 041246 LONG3: .WORD LONG.2 ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7200 040550 026767 000466 000470 CMP LONG.1, LONG.2 ;COMPARE RESULTS
7201 040556 001004 BNE 1$
7202 040560 026767 000460 000462 CMP LONG.1+2, LONG.2+2 ;COMPARE RESULTS OF SECOND WORD
7203 040566 001403 BEQ DONE
7204 040570 004767 000064 1$: JSR PC,CISER1 ;GET PC, ERROR DATA
7205 040574 104020 ERROR 20 ;REPORT ERROR
7206 040576 000167 001724 DONE: JMP RELE9 ;GO ON TO NEXT TEST
7207
7208 ;SUBROUTINE-SETUP
7209 ;SETS UP CHAR STRING DESCRIPTORS
7210 ;USAGE: JSR PC, SETUP
7211 ;NO ARGUMENTS
7212 040602 012767 000310 000362 SETUP: MOV #200., SRC.1D ;SET SOURCE LENGTH
7213 040610 016767 000450 000356 MOV BUFFAD, SRC.1A ;SET SOURCE ADDRESS
7214 040616 012767 000310 000352 MOV #200., DST.1D ;DEST LENGTH
7215 040624 016767 000434 000346 MOV BUFFAD, DST.1A ;DEST ADDRESS
7216 040632 062767 000310 000340 ADD #200., DST.1A ;ADJUST FOR DST
7217 040640 000207 RTS PC ;RETURN
7218 ;SUBROUTINES-CISER, CISER1, CISER2 AND CISER3
7219 ;GETS SHOULD AND WAS DATA AND ADDRESSES
7220 ;USAGE: JSR PC, CISER(X)
7221 ;NO ARGUMENTS
7222 040642 016737 000326 001302 CISER: MOV SRC.1A, @STMP0 ;SHOULD BE ADDRESS
7223 040650 016737 000324 001304 MOV DST.1A, @STMP1 ;WAS ADDRESS
7224 040656 000207 RTS PC ;RETURN
7225 040660 016737 000356 001302 CISER1: MOV LONG.1P, @STMP0 ;SHOULD BE ADDRESS
7226 040666 016737 000354 001304 MOV LONG.2P, @STMP1 ;WAS ADDRESS
7227 040674 000207 RTS PC ;RETURN
7228 040676 016737 000322 001302 CISER2: MOV E, @STMP0 ;SHOULD BE ADDRESS
7229 040704 016737 000320 001304 MOV F, @STMP1 ;WAS ADDRESS
7230 040712 000207 RTS PC ;RETURN
7231 040714 016737 000270 001302 CISER3: MOV A, @STMP0 ;SHOULD BE ADDRESS
    
```

7232	040722	016737	000262	001304	MOV	A,@STMP1	:WAS ADDRESS
7233	040730	000207			RTS	PC	:RETURN
7234							
7235						:TRANSLATE TABLE	
7236						:USED BY MOVE TRANSLATE	
7237						:USED BY SPAN AND SCAN INSTRUCTIONS	
7238						:128 CHAR ASCII	
7239							
7240	040732	177	176	175	TRANS:	.BYTE	177,176,175,174,173,172,171,170
7241	040735	174	173	172			
7242	040740	171	170				
7243	040742	167	166	165		.BYTE	167,166,165,164,163,162,161,160
7244	040745	164	163	162			
7245	040750	161	160				
7246	040752	157	156	155		.BYTE	157,156,155,154,153,152,151,150
7247	040755	154	153	152			
7248	040760	151	150				
7249	040762	147	146	145		.BYTE	147,146,145,144,143,142,141,140
7250	040765	144	143	142			
7251	040770	141	140				
7252	040772	137	136	135		.BYTE	137,136,135,134,133,132,131,130
7253	040775	134	133	132			
7254	041000	131	130				
7255	041002	127	126	125		.BYTE	127,126,125,124,123,122,121,120
7256	041005	124	123	122			
7257	041010	121	120				
7258	041012	117	116	115		.BYTE	117,116,115,114,113,112,111,110
7259	041015	114	113	112			
7260	041020	111	110				
7261	041022	107	106	105		.BYTE	107,106,105,104,103,102,101,100
7262	041025	104	103	102			
7263	041030	101	100				
7264	041032	077	076	075		.BYTE	077,076,075,074,073,072,071,070
7265	041035	074	073	072			
7266	041040	071	070				
7267	041042	067	066	065		.BYTE	067,066,065,064,063,062,061,060
7268	041045	064	063	062			
7269	041050	061	060				
7270	041052	057	056	055		.BYTE	057,056,055,054,053,052,051,050
7271	041055	054	053	052			
7272	041060	051	050				
7273	041062	047	046	045		.BYTE	047,046,045,044,043,042,041,040
7274	041065	044	043	042			
7275	041070	041	040				
7276	041072	037	036	035		.BYTE	037,036,035,034,033,032,031,030
7277	041075	034	033	032			
7278	041100	031	030				
7279	041102	027	026	025		.BYTE	027,026,025,024,023,022,021,020
7280	041105	024	023	022			
7281	041110	021	020				
7282	041112	017	016	015		.BYTE	017,016,015,014,013,012,011,010
7283	041115	014	013	012			
7284	041120	011	010				
7285	041122	007	006	005		.BYTE	007,006,005,004,003,002,001,000
7286	041125	004	003	002			
7287	041130	001	000				

7288
7289
7290
7291
7292 041132 177760
7293 041134 000011
7294 041136 000006
7295 041140 177417
7296 041142 000220
7297 041144 000140
7298 041146 170377
7299 041150 004400
7300 041152 003000
7301 041154 007777
7302 041156 070000
7303 041160 100000
7304 041162 360
7305 041163 014
7306 041164 017
7307 041166
7308 041166 000000
7309 041170 000000
7310
7311
7312
7313 041172 000310
7314 041174 041706
7315 041176 000310
7316 041200 042216
7317 041202 000031
7318 041204 041770
7319 041206 000034
7320 041210 041706
7321 041212 000016
7322 041214 041742
7323 041216 000012
7324 041220 041760
7325 041222 000037
7326 041224 041776
7327 041226 000037
7328 041230 042036
7329 041232 000037
7330 041234 042076
7331 041236 000001
7332 041240 040732
7333 041242 001020 000000
7334 041246 000002
7335 041252 000001
7336 041254 000040
7337 041256 040732
7338 041260 041242
7339 041262 041246
7340 041264 041706
7341
7342
7343

;MASK TABLES
;FOR MAKING VALID DECIMAL DATA

MSKTAB: .WORD 177760
 .WORD 11
 .WORD 6
 .WORD 177417
NINTAB: .WORD 220
 .WORD 140
 .WORD 170377
 .WORD 4400
SIXTAB: .WORD 3000
 .WORD 7777
 .WORD 70000
 .WORD 100000
HIMASK: .BYTE 360
SIGN: .BYTE 014
LOMASK: .BYTE 017
 .EVEN
SAVRNL: .WORD 0
SAVRNH: .WORD 0

;CHARACTER STRING DESCRIPTOR TABLE

SRC.1D: .WORD 200.
SRC.1A: .WORD BUFF
DST.1D: .WORD 200.
DST.1A: .WORD BUFF+200.
OBJ.1D: .WORD 25.
OBJ.1A: .WORD BUFF+50.
A.DSC: .WORD 28.
A: .WORD BUFF
B.DSC: .WORD 14.
B: .WORD BUFF+28.
C.DSC: .WORD 10.
C: .WORD BUFF+42.
E.DSC: .WORD 31.
E: .WORD BUFF+56.
F.DSC: .WORD 31.
F: .WORD BUFF+88.
D.DSC: .WORD 31.
D: .WORD BUFF+120.
SET.1D: .WORD 1
TRANS4: .WORD TRANS
LONG.1: .WORD 528.,0
LONG.2: .BLKW 2
ONE: .WORD 1
CHAR: .WORD '
TRANS3: .WORD TRANS
LONG1P: .WORD LONG.1
LONG2P: .WORD LONG.2
BUFFAD: .WORD BUFF

;OFFTAB CONTAINS ALL ABSOLUTE ADDRESSES TO BE MODIFIED WHEN RELOCATING

7344	041266	037024	037050	037062	OFFTAB: .WORD	SRC1,SRC2,SRC3,SRC4,SRC5,SRC6,SRC7,SRC8,SRC9,SRC10
7345	041274	037114	037142	037152		
7346	041302	037204	037236	037250		
7347	041310	037276				
7348	041312	037320	037346	037370	.WORD	SRC11,SRC12,SRC13,SRC14,SRC15,SRC16,SRC17,SRC18,SRC19,SRC20
7349	041320	037416	037442	037466		
7350	041326	037532	037556	037604		
7351	041334	037614				
7352	041336	037026	037052	037064	.WORD	DST1,DST2,DST3,DST4,DST5,DST6,DST7,DST8,DST9,DST10
7353	041344	037144	037154	037206		
7354	041352	037240	037252	037322		
7355	041360	037350				
7356	041362	037356	037360	037372	.WORD	DST11,DST12,DST13,DST14,DST15,DST16,DST17,DST18,DST19,DST20
7357	041370	037420	037430	037432		
7358	041376	037444	037534	037606		
7359	041404	037616				
7360	041406	037424	037436	041256	.WORD	TRANS1,TRANS2,TRANS3,OBJ1,SET1,SET2
7361	041414	037300	037116	037210		
7362	041422	037054	037242	037254	.WORD	CHAR1,CHAR2,CHAR3,CHAR4,CHAR5
7363	041430	037324	037620			
7364	041434	037762	037764	040000	.WORD	A1,A2,A3,A4,A5,A6,A7,A8
7365	041442	040050	040170	040204		
7366	041450	040364	040366			
7367	041454	040010	040062	040376	.WORD	B1,B2,B3,B4,B5,B6,B7
7368	041462	040400	040444	040474		
7369	041470	040432				
7370	041472	040030	040052	040406	.WORD	C1,C2,C3,C4,C5,C6,C7
7371	041500	040410	040442	040476		
7372	041506	040430				
7373	041510	040002	040020	040032	.WORD	D1,D2,D3,D4,D5,D6,D7,D8,D9,D10
7374	041516	040042	040066	040074		
7375	041524	040466	040452	040500		
7376	041532	040506				
7377	041534	040012	040022	040044	.WORD	E1,E2,E3,E4,E5,E6,E7,E8,E9,E10
7378	041542	040106	040126	040172		
7379	041550	040176	040206	040214		
7380	041556	040402				
7381	041560	040420	040422	040464	.WORD	E11,E12,E13,E14,E15,E16,E17,E18
7382	041566	040454	040520	040536		
7383	041574	040544	040134			
7384	041600	040024	040040	040054	.WORD	F1,F2,F3,F4,F5,F6,F7,F8,F9,F10
7385	041606	040064	040076	040110		
7386	041614	040200	040216	040412		
7387	041622	040416				
7388	041624	040446	040462	040510	.WORD	F11,F12,F13,F14
7389	041632	040522				
7390	041634	041174	041200	041204	.WORD	SRC.1A,DST.1A,OBJ.1A
7391	041642	041210	041214	041220	.WORD	A,B,C,D,E,F,BUFFAD,TRANS4
7392	041650	041234	041224	041230		
7393	041656	041264	041240			
7394	041662	041242	041246	040130	.WORD	LONG.1P, LONG.2P, LONG1, LONG2, LONG3, LONG4
7395	041670	040540	040546	040136		
7396	041676	040100	040512	040456	.WORD	ONE1, ONE2, ONE3
7397	041704	000000			.WORD	Q ;TABLE TERMINATER
7398						
7399						

7400
7401
7402
7403
7404 041706 000310
7405
7406 042526 000004
7407 042530 010702
7408 042532 062702 000012
7409 042536 012707 043764
7410 042542 000000
7411
7412
7413 042544
7414
7415
7416
7417 042544 000240
7418 042546 112737 000076 001202
7419 042554 013737 001202 177570
7420 042562 113737 001202 177570
7421 042570 005037 001534
7422 042574 012704 000100
7423 042600 032737 000400 001500
7424 042606 001002
7425 042610 000167 000204
7426 042614 132777 000200 136422
7427 042622 001774
7428 042624 012737 001553 001270
7429 042632 106277 136406
7430 042636 000001
7431
7432
7433 042640
7434
7435
7436
7437
7438
7439 042640 132737 000020 177776
7440 042646 001071
7441 042650 030477 136370
7442 042654 001375
7443 042656 112737 000300 177776
7444 042664 150477 136354
7445 042670 100375
7446 042672 032737 001000 001500
7447 042700 001447
7448 042702 012737 042772 000064
7449 042710 012737 043004 000100
7450 042716 012737 000340 000102
7451 042724 005027
7452 042726 000000
7453 042730 000240
7454 042732 000240
7455 042734 000240

:BUFFER SPACE
:200 WORDS LONG
:USED FOR SOURCE AND DESTINATIONS
BUFF: .BLKW 200.
RELE9: SCOPE
MOV PC,R2
ADD #15,PC
MOV #R1,...,PC ;GO RELOCATE PROGRAM CODE
FEL99: .WORD 0
999999999999 LAST ADDRESS OF CODE TO BE RELOCATED 9999999999
ENDCIS:
:*****
:*TEST 76 TELETYPE AND CLOCK TESTS
:*****
TST76: NOP
MOVB #76,@STSTNM ;LOAD TEST NUMBER
MOV @STSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
MOVB @STSTNM,@SWR
TTYCHK: CLR @FACTOR
MOV #100,R4 ;SET R4 = CONSTANT 100
BIT #TTOPT,@OPT.CP ;BRANCH IF TTY
BNE 1\$;ON SYSTEM
JMP ARBFIN ;JUMP IF NOT
1\$: BITB #200,@STPS ;CHECK IF TTY IS READY
BEQ 1\$
MOV #NULLS-1,@SREG5;SET ADDRESS OF ASCII STRING TO TYPE
ASRB @STPS ;SET IE BIT. SEE TPISR FOR INT SERVICE.
WAIT ;WAIT FOR INTERRUPT
DUMMY:
:ROUTINE TO CHECK PRIORITY ARBITRATION LOGIC
:THE BELOW TEST WILL INHIBIT INTERRUPTS ON LEVEL 6 AND ABOVE (LOCKING
:OUT THE LINE CLOCK) AND THEN SET UP THE TTY TO INTERRUPT. NEXT THE
:PRIORITY LEVEL WILL BE SET TO 0 ALLOWING INTERRUPTS IN WHICH CASE
:THE LINE CLOCK (AT LEVEL 6) SHOULD INTERRUPT BEFORE THE TTY (AT LEVEL 4).
1\$: BITB #20,@PSW
BNE ARBEX ;EXIT TEST IF 'T' BIT SET
2\$: BIT R4,@STPS ;WAIT FOR TTY TO BE NOT
BNE 2\$;BUSY
MOVB #300,@PSW ;SET PRIORITY LEVEL 6
3\$: BISB R4,@STPS ;SET IE BIT
BPL 3\$;AND WAIT FOR READY
BIT #LKOPT,@OPT.CP ;LINE CLOCK AVAILABLE?
BEQ ARBFIN ;BRANCH IF NO
MOV #7\$,@TPVEC ;SET TTY VECTOR
MOV #8\$,@LKVEC ;SET CLOCK VECTORS
MOV #PR7,@LKVEC+2
CLR (PC)+ ;CLEAR CHECK WORD
4\$: .WORD 0
NOP
NOP
NOP

```
7456 042736 010437 177546      MOV      R4,@#LKS
7457 042742 113700      5$:      MOVB   @#(PC)+,R0      ;GET CLOCK STATUS & BRANCH IF READY
7458 042744 177546      6$:      .WORD  LKS                  ;CONTAINS ADDRESS OF L CLOCK STAT
7459 042746 100375      BPL      5$
7460 042750 000240      NOP
7461                                ;AT THIS TIME BOTH THE CLOCK
7462 042752 105037 177776      CLR      @#PSW              ;ARE READY TO INTERRUPT
7463                                ;SET PRIORITY LEVEL 0
7464                                ;A CLOCK INTERRUPT WILL OCCUR (3$) AND LOC 4$ WILL BE INCREMENTED
7465                                ;AFTER THE CLOCK SERVICE A TTY INTERRUPT WILL OCCUR. THE TTY INT SERV
7466                                ;ICE WILL SHIFT LEFT 4$.
7467 042756 022767 000002 177742      CMP      #2,4$              ;CHECK THAT THE CLOCK
7468 042764 001415      BEQ      ARBF IN            ;& TTY INTERRUPTED IN
7469 042766 104000      HLT
7470 042770 000413      BR       ARBF IN            ;THE PROPER SEQUENCE
7471
7472 042772 005077 136246      7$:      CLR      @#STPS            ;CLEAR IE BIT
7473 042776 006367 177724      ASL      4$                 ;SHIFT INDICATOR
7474 043002 000002      RTI
7475                                ;RETURN
7476 043004 005267 177716      8$:      INC      4$
7477 043010 012737 054300 000100      MOV      #LKS RV,@#LKVEC    ;SET CLOCK VECTORS
7478 043016 000002      RTI
7479
7480
7481 043020 012737 063434 000064      ARBF IN: MOV   #TPISR,@#TPVEC ;RESTORE TTY VECTOR
7482 043026 005077 136212      CLR      @#STPS            ;CLEAR IE BIT
7483 043032
7484                                ARBF EX:
7485                                ;*****
7486                                ;*TEST 77      TURN ON UBE AND MBT
7487                                ;*      TURN ON THE MASS BUS TESTER AND UNIBUS EXERCISER IF PRESENT
7488                                ;*****
7488 043032 000240      TST77:  NOP
7489 043034 112737 000077 001202      MOV      #77,@#STSTNM      ;LOAD TEST NUMBER
7490 043042 013737 001202 177570      MOV      @#STSTNM,@#DISPLAY ;:DISPLAY TEST NUMBER
7491 043050 113737 001202 177570      MOV      @#STSTNM,@#SWR
7492 043056 032737 001000 001500      BIT      #LKOPT,@#OPT.CP    ;BRANCH IF NOT AVAIL
7493 043064 001411      BEQ      UBESET
7494 043066 012737 054300 000100      MOV      #LKS RV,@#LKVEC
7495 043074 012737 000340 000102      MOV      #PR7,@#LKVEC+2
7496 043102 052737 000100 177546      BIS      #100,@#LKS         ;SET IE BIT
7497
7498                                ;*****
7499                                ;TURN ON THE UNIBUS EXERCISER IF PRESENT
7500 043110 105737 001500      UBESET: TSTB @#OPT.CP      ;IS UBE OPTION AVAILABLE?
7501 043114 100015      BPL      MBTSET            ;BRANCH IF NO
7502 043116 032737 010000 177570      BIT      #SW12,@#SWR        ;INHIBIT UBF?
7503 043124 001011      BNE      MBTSET            ;BRANCH IF YES
7504 043126 032737 000040 172516      BIT      #BIT5,@#MMP3       ;IS MAP ON?
7505 043134 001045      BNE      STMM              ;BRANCH IF YES
7506 043136 004737 062214      JSR      PC,@#UBEINIT       ;INITIALIZE UBE
7507 043142 012772 064545 000000      MOV      #64545,@(R2)      ;START UBE
7508
7509                                ;*****
7510                                ;TURN ON THE MASS BUS TESTER IF PRESENT
7511 043150 032737 002000 001500      MBTSET: BIT @#MBTOPT,@#OPT.CP ;IS MBT AVAIL. ABLE?
```

7512 043156 001434
7513 043160 032737 000010 177570
7514 043166 001030
7515 043170 122737 000060 001560
7516 043176 001024
7517 043200 105737 001531
7518 043204 001021
7519 043206 052777 000047 137104
7520 043214 012777 000007 137076
7521 043222 005077 137062
7522 043226 012777 053766 137074
7523 043234 012777 000240 137070
7524 043242 112777 000161 137036
7525
7526
7527
7528
7529
7530
7531 043250 112737 000100 001202
7532 043256 112737 000100 177570
7533
7534
7535 043264 032737 000100 177570
7536 043272 001402
7537 043274 000167 002622
7538
7539
7540
7541
7542
7543
7544 043300 013727 177776
7545 043304 000000
7546 043306 012737 000200 177776
7547 043314 004767 017642
7548
7549
7550 043320 012700 077406
7551 043324 010037 172300
7552 043330 010037 172302
7553 043334 010037 172304
7554 043340 010037 172306
7555 043344 010037 172310
7556 043350 010037 172312
7557 043354 010037 172314
7558 043360 010037 172316
7559
7560 043364 005037 172340
7561 043370 012737 000200 172342
7562 043376 012737 000400 172344
7563 043404 012737 000600 172346
7564 04342 013737 001546 172350
7565 043420 013737 172350 172352
7566 043426 062737 000600 172352
7567 043434 012737 177600 172356

BEQ STMM ;BRANCH IF NO
BIT #SW3,@#SWR ;INHIBIT MBT?
BNE STMM ;BRANCH IF YES
CMPB #60,@#SUBPASS ;FIRST SUB-PASS?
BNE STMM ;BRANCH IF NO
TSTB @#MMON ;MEM MGMT ON?
BNE STMM ;BRANCH IF YES
MBT1: BIS #47,@#BTBL+12 ;CLEAR THE MBT
MOV #7,@#BTBL+12 ;SELECT UNIT 7
CLR @#BTBL+2 ;CLEAR THE WORD COUNT
MOV @#BTSRV,@#BTBL+22 ;SETUP INTERRUPT VECTOR
MOV #PR5,@#BTBL+24 ;SET VECTOR PSW
MOVB #161,@#BTBL ;START MBT
:*****
: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
:*****
STMM: MOVB #100,@#STSTM ;LOAD TEST NUMBER
MOVB #100,@#SWR ;IN SWICH REG TOO
;ANY TIME TEST 100 IS IN ERROR REPORT
;THEN ERROR OCCURRED DURING RELOCATION
BIT #SW6,@#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 @#PSW,(PC)+ ;SAVE CURRENT PSW
OLDPSW: .WORD 0
MOV #PR4,@#PSW
JSR PC,CLRTBIT ;CO CLEAR 'T' BIT IF SET
:NOW SETUP MEMORY MANAGEMENT REGISTERS.
MOV #77406,R0 ;SET CONSTANT=R/W UP 4K WORDS
MOV R0,@#KIPDR0 ;SET KIPDR0 THROUGH 7 R/W UP 4K WORDS
MOV R0,@#KIPDR1
MOV R0,@#KIPDR2
MOV R0,@#KIPDR3
MOV R0,@#KIPDR4
MOV R0,@#KIPDR5
MOV R0,@#KIPDR6
MOV R0,@#KIPDR7
CLR @#KIPAR0 ;SET UP KIPDR0 THROUGH 3 FOR NO RELOCATION
MOV #200,@#KIPAR1
MOV #400,@#KIPAR2
MOV #600,@#KIPAR3
MOV @#NEXPAR,@#KIPAR4 ;PAR4 MAPS TO BEGINNING OF RELOCATION SPOT
MOV @#KIPAR4,@#KIPAR5
ADD #600,@#KIPAR5 ;PAR5 MAPS TO TOP 4K PAGE (NEXPAR START ADDRESS + 12K)
MOV #177600,@#KIPAR7 ;AND OF COUSE THE I/O PAGE

```
7568 ;NOW SETUP USER MEM MGMT REGISTERS
7569 043442 010037 177600 1S: MOV R0,@UIPDR0 ;SET UP USER MEM MGMT REGS
7570 043446 010037 177602 MOV R0,@UIPDR1
7571 043452 010037 177604 MOV R0,@UIPDR2
7572 043456 010037 177606 MOV R0,@UIPDR3
7573 043462 010037 177616 MOV R0,@UIPDR7
7574 043466 016737 136054 177640 MOV NEXPAR,@UIPAR0
7575 043474 013737 177640 177642 MOV @UIPAR0,@UIPAR1
7576 043502 062737 000200 177642 ADD #200,@UIPAR1
7577 043510 013737 177640 177644 MOV @UIPAR0,@UIPAR2
7578 043516 062737 000400 177644 ADD #400,@UIPAR2
7579 043524 013737 177640 177646 MOV @UIPAR0,@UIPAR3
7580 043532 062737 000600 177646 ADD #600,@UIPAR3
7581 043540 013737 172356 177656 MOV @KIPAR7,@UIPAR7
7582
7583 043546 010037 172200 MOV R0,@SIPDR0 ;SET UP SUPERVISOR MEM MGMT REGS
7584 043552 010037 172202 MOV R0,@SIPDR1
7585 043556 010037 172204 MOV R0,@SIPDR2
7586 043562 010037 172206 MOV R0,@SIPDR3
7587 043566 010037 172216 MOV R0,@SIPDR7
7588 043572 016737 135750 172240 MOV NEXPAR,@SIPAR0
7589 043600 013737 172240 172242 MOV @SIPAR0,@SIPAR1
7590 043606 062737 000200 172242 ADD #200,@SIPAR1
7591 043614 013737 172240 172244 MOV @SIPAR0,@SIPAR2
7592 043622 062737 000400 172244 ADD #400,@SIPAR2
7593 043630 013737 172240 172246 MOV @SIPAR0,@SIPAR3
7594 043636 062737 000600 172246 ADD #600,@SIPAR3
7595 043644 013737 172356 172256 MOV @KIPAR7,@SIPAR7
7596 043652 012737 000001 177572 MOV #1,@SR0 ;ENABLE MEM MGMT
7597 043660 012737 000060 172516 MOV #60,@SR3 ;SETUP SR3
7598 043666 110637 001531 MOVB SP,@MON ;SET MEM MGMT ON IND ON
7599 043672 005037 000006 RETRY: CLR @ERRVEC+2
7600 043676 012737 046120 000004 MOV #ENDMEM,@ERRVEC;SET TIME OUT TRAP VECTOR
7601 043704 013701 000116 MOV @CACHVEC+2,R1 ;SAVE CACHVEC PSW
7602 043710 005037 000116 CLR @CACHVEC+2
7603 043714 012737 046120 000114 MOV #ENDMEM,@CACHVEC ;SET UP CACHE VECTOR FOR HOLE
7604 043722 012702 100000 MOV #100000,R2 ;SETUP GENERAL REGISTERS
7605 043726 012700 000000 MOV #0,R0 ;DATA WILL BE RELOCATED FROM
7606 ;ADDRESS IN R0 TO ADDRESS IN R2 AFTER MAPPING
7607 043732 012703 127776 MOV #127776,R3 ;SELECT PAR5 + 2K (14K TOTAL REQUIRED)
7608 043736 010013 MOV R0,(R3) ;TRAP TO ENDMEM IF INSUFFICIENT MEMORY
7609 043740 012737 064270 000004 MOV #ERPRT,@ERRVEC ;RESTORE ERROR TRAP VECTOR
7610 043746 010137 000116 MOV R1,@CACHVEC+2
7611 043752 012737 063464 000114 MOV #.PARSRV,@CACHVEC ;RESTORE CACHE VECTOR
7612 043760 000137 044160 JMP @MON
7613
7614 ;*****
7615 .SBTTL RELOCATION ROUTINE
7616 ;* THIS ROUTINE IS USED TO RELOCATE THE 9 SUBTESTS UP TO 28K.
7617 ;* IF RELOCATION BY AN I/O DEVICE IS SELECTED, CONTROL IS PASSED
7618 ;* TO THE I/O MONITOR.
7619 ;* ENTER WITH:
7620 ;* FRSTAD=PHYSICAL ADDRESS OF FIRST CODE
7621 ;* FACTOR=NUMBER OF BYTES ABOVE BASE CODE
7622 ;* R2 =LAST PHYSICAL ADDRESS OF THE SECTION
7623 ;* EXIT TO I/O MONITOR WITH:
7624 ;* OLDBASE FIRST PHYSICAL ADDRESS TO BE RELOCATED
```

```

7624      :*          NUBASL =FIRST PHYSICAL ADDRESS TO RELOCATE TO
7625      :*          IOWC  =TWO'S COMPLIMENT WORD COUNT
7626      :*****
7627 043764 032737 000100 177570 RELOC: BIT    #SW6,@SWR      ;IS RELOCATION DISABLED?
7628 043772 001067          BNE    EXITRE      ;BRANCH IF YES
7629 043774 105737 001531      TSIB    @MON      ;IS MEMORY MGMT ON?
7630 044000 001064          BNE    EXITRE      ;BRANCH IF YES
7631 044002 013700 001540      MOV    @FRSTAD,R0 ;GET FIRST ADDRESS TO BE RELOCATED
7632 044006 010005          MOV    R0,R5
7633      ;LAST ADDRESS IS IN R2
7634 044010 010203          MOV    R2,R3      ;SAVE LAST ADDRESS
7635 044012 010204          MOV    R2,R4
7636 044014 160004          SUB    R0,R4      ;R4 NOW HAS BYTE COUNT
7637 044016 010437 001536      MOV    R4,@SFACOR ;SAVE BYTE COUNT
7638 044022 005737 001534      TST    @FACTOR   ;FIRST RELOC IS TO END*AG+2
7639 044026 001004          BNE    1$        ;BRANCH IF NOT EXECUTING BASE CODE
7640 044030 010237 044156      MOV    R2,@RETPC  ;SAVE RETURN PC TO NEXT SECTION
7641 044034 013702 001542      MOV    @FRSTMEM,R2 ;GET FIRST ADDRESS TO RELOCATE TO
7642 044040 060204          1$: ADD    R2,R4      ;R4 NOW CONTAINS LAST MEM ADDRESS
7643 044042 020437 001544      CMP    R4,@LSTMEM ;ENOUGH MEMORY?
7644 044046 101042          BHI    NOMEM     ;BRANCH IF NO
7645 044050 160204          SUB    R2,R4      ;R4 NOW HAS BYTE COUNT
7646 044052 005037 001534      CLR    @FACTOR
7647 044056 032737 000400 177570 BIT    #SW8,@SWR      ;INHIB RELOC BY I/O DEVICE?
7648 044064 001414          BEQ    RELNIO    ;BRANCH IF YES
7649 044066 010037 001566      MOV    R0,@OLDBASE ;SAVE START ADDRESS
7650 044072 010237 001570      MOV    R2,@NWBASL ;SAVE NEW BASE ADDRESS
7651 044076 005037 001572      CLR    @NWBASH
7652 044102 006204          ASR    R4        ;MAKE IT A WORD COUNT
7653 044104 005404          NEG    R4        ;GET TWO'S COMPLIMENT
7654 044106 010437 001574      MOV    R4,@IOWC  ;SAVE R4 AS WORDCOUNT
7655 044112 000167 000122      JMP    ENTER2    ;GO TO I/O MONITOR
7656      ;RELOCATE BY CPU-MEMORY MANAGEMENT OFF
7657 044116 012022      RELNIO: MOV    (R0)+,(R2)+ ;RELOCATE CODE
7658 044120 020003          CMP    R0,R3     ;DONE YET?
7659 044122 001375          BNE    RELNIO    ;BRANCH IF NO
7660 044124 004737 062534      JSR    PC,@CHKDAT ;GO CHECK DATA
7661 044130 102010          BVC    EXITRE
7662 044132 010037 001302      MOV    R0,@STMP0 ;SAVE R0 FOR TYPEOUT
7663 044136 010237 001522      MOV    R2,@VADR  ;SAVE R2
7664 044142 004737 062432      JSR    PC,@CNVADR ;CONVERT R2 TO A PHYSICAL ADR
7665 044146 104006          ERROR 6
7666 044150 000401          BR    NOMEM
7667 044152 010207      EXITRE: MOV    R2,PC ;GO EXECUTE RELOCATED CODE
7668 044154 011707      NOMEM: MOV    (PC),PC ;GO TO NEXT SECTION
7669 044156 000000      RETPC: .WORD 0 ;CONTAINS PC OF NEXT SECTION
7670      ;*****
7671      .SBTTL I/O RELOCATION MONITOR
7672      ;* THIS ROUTINE IS USED TO SCHEDULE I/O DEVICES FOR SUBTEST
7673      ;* RELOCATION AND PROGRAM RELOCATION. THE I/O DEVICE UNIT
7674      ;* NUMBER IS DETERMINED, THE BUS ADDRESS CALCULATED, THE WORD
7675      ;* COUNT CALCULATED AND PASSED TO THE DEVICE HANDLER.
7676      ;*****
7677 044160 012737 044166 001212 IOMON: MOV    #1$,@SLPERR ;SETUP ERROR LOOP
7678 044166 012737 000000 001566 1$: MOV    #0,@OLDBASE
7679 044174 013705 001546      MOV    @NEXPAR,R5 ;SETUP R4 AND R5
    
```

7680	044200	005004				CLR	R4		:TO FORM 22 BIT ADDRESS
7681	044202	073427	000006			ASHC	#6,R4		:FORM 22 BIT ADDRESS
7682	044206	010537	001570			MOV	R5,@#NWBASL		:SAVE LOWER 16 BITS
7683	044212	010437	001572			MOV	R4,@#NWBASH		:SAVE UPPER 6 BITS
7684	044216	032737	000400	177570		BIT	#SW8,@#SWR		:RELOCATE VIA I/O?
7685	044224	001002				BNE	2\$:BRANCH UF YES
7686	044226	000167	001440			JMP	RELOCP		:GO RELOCATE VIA CP
7687	044232	012737	174000	001574	2\$:	MOV	#174000,@#IOWC		:SET WORD COUNT TO 2K
7688	044240	005037	001306			ENTER2: CLR	@#STMP2		
7689	044244	012737	177776	001604		MOV	#-2,@#RNTBINX		:SETUP RUN TABLE INDEX
7690	044252	005037	001302			CLR	@#STMP0		
7691	044256	005002				CLR	R2		:CLEAR LEGAL DEV FLAG
7692	044260	032737	000040	177570	41\$:	BIT	#SW5,@#SWR		:INHIBIT ROUND ROBIN?
7693	044266	001416				BEQ	50\$:BRANCH IF NO
7694	044270	005737	001302			TST	@#STMP0		:FLAG SET?
7695	044274	001027				BNE	43\$:BRANCH IF YES
7696	044276	113737	177570	001600		MOVB	@#SWR,@#DEVINDX		:GET DEVICE FROM SWITCHES
7697	044304	042737	177770	001600		BIC	#177770,@#DEVINDX		:MASK LOWER 3 BITS
7698	044312	006337	001600			ASL	@#DEVINDX		:ADJUST FOR WORD INDEX
7699	044316	005237	001302			INC	@#STMP0		:SET FLAG
7700	044322	000414				BR	43\$:CONTINUE
7701	044324	012705	000010		50\$:	MOV	#10,R5		:SET SOB COUNT
7702	044330	022737	000016	001600	40\$:	CMP	#16,@#DEVINDX		:LAST DEVICE YET?
7703	044336	001003				BNE	42\$:BRANCH IF NO
7704	044340	012737	177776	001600	48\$:	MOV	#-2,@#DEVINDX		:INIT DEVICE INDEX
7705	044346	062737	000002	001600	42\$:	ADD	#2,@#DEVINDX		:INCREMENT INDEX
7706	044354	013703	001600		43\$:	MOV	@#DEVINDX,R3		:GET INDEX
7707	044360	012737	000401	001304		MOV	#401,@#STMP1		:INIT UNIT MASK
7708	044366	012704	000010			MOV	#10,R4	:SET SOB COUNT	
7709	044372	133763	001304	001642	44\$:	BITB	@#STMP1,SYSSIZE(R3)		:IS THIS UNIT EXISTENT?
7710	044400	001405				BEQ	52\$:BRANCH IF NO
7711	044402	005202				INC	R2		:SET LEGAL DEVICE FLAG
7712	044404	133763	001305	001643		BITB	@#STMP1+1,SYSSIZE+1(R3)		:HAS IT BEEN USED?
7713	044412	001516				BEQ	11\$:BRANCH IF NO
7714	044414	006337	001304		52\$:	ASL	@#STMP1		:SELECT NEXT UNIT
7715	044420	077414				SOB	R4,44\$:CONTINUE
7716	044422	005737	001302			TST	@#STMP0		:INHIBIT ROUND ROBIN?
7717	044426	001013				BNE	45\$:BRANCH IF YES
7718	044430	077541				SOB	R5,40\$:CONTINUE
7719	044432	005702				TST	R2		:ANY DEVICES AT ALL?
7720	044434	001442				BEQ	46\$:BRANCH IF NO
7721	044436	012704	000010			MOV	#10,R4		:SET SOB COUNT
7722	044442	012701	001643			MLV	#SYSSIZE+1,R1		:GET ADR OF SIZE TABLE
7723	044446	105021			47\$:	CLRB	(R1)+		:CLEAR ALL USED BITS
7724	044450	005201				INC	R1		:IN ALL DEVICES
7725	044452	077403				SOB	R4,47\$:CONTINUE
7726	044454	000701				BR	41\$		
7727	044456	005702			45\$:	TST	R2		:WAS IT A LEGAL DEVICE?
7728	044460	001403				BEQ	49\$:BRANCH IF NO
7729	044462	105063	001643			CLRB	SYSSIZE+1(R3)		:CLEAR ALL USED BITS THIS DEV
7730	044466	000732				BR	43\$		
7731	044470	010367	000016		49\$:	MOV	R3,60\$		
7732	044474	062767	064440	000010		ADD	#MSGINX,60\$:GEN MESSAGE ADR
7733	044502	017767	000004	000002		MOV	@60\$,60\$		
7734	044510	104400				TYPE			
7735	044512	000000			60\$:	.WORD			

Address	OpCode	Operand 1	Operand 2	Operand 3	Operand 4	Comment
7736	044514	104400	044522			TYPE .65\$::TYPE ASCIZ STRING
7737	044520	000407				BR 64\$::GET OVER THE ASCIZ
7738						::65\$: .ASCIZ /UNAVAILABLE/<CRLF>
7739	044540					64\$: BR ENTER2
7740	044540	000637				46\$: TSTB @PQV :ACT1?
7741	044542	105737	001532			BNE 51\$:BRANCH IF YES
7742	044546	001016				INC #-1
7743	044550	005227	177777			BNE 51\$
7744	044554	001013				TYPE .67\$::TYPE ASCIZ STRING
7745	044556	104400	044564			BR 66\$::GET OVER THE ASCIZ
7746	044562	000410				::67\$: .ASCIZ ?NO I/O DEVICES?<CRLF>
7747						66\$:
7748	044604					51\$: TSTB @MNON :MGMT ON?
7749	044604	105737	001531			BNE 61\$:BRANCH IF YES
7750	044610	001012				MOV @OLDBASE,R0 :RESTORE R0
7751	044612	013700	001566			MOV @NBASL,R2 :RESTORE R2
7752	044616	013702	001570			MOV @SFACOR,R3 :GET RELOCATION FACTOR
7753	044622	013703	001536			ADD R0,R3 :FORM LAST ADDRESS
7754	044626	060003				MOV R0,R5 :SETUP R5
7755	044630	010005				JMP RELNIO :GO RELOCATE WITH CP
7756	044632	000167	177260			61\$: MOV #100000,R2 :SETUP REGISTERS
7757	044636	012702	100000			CLR R0 :WITH FROM AND TO ADDRESS
7758	044642	005000				JMP @RELOCP :RELOCATE VIA CP
7759	044644	000137	045672			11\$: TSTB RP3HSTAT(R3) :IS HANDLER BUSY?
7760	044650	105763	001736			BMI 8\$:BRANCH IF NO
7761	044654	100405				TST @STMP0 :ROUND ROBIN?
7762	044656	005737	001302			BNE 11\$:BRANCH IF NO
7763	044662	001372				JMP 41\$
7764	044664	000167	177370			8\$: TST RP3HSTAT(R3) :DID HANDLER FAIL?
7765	044670	005763	001736			BPL 62\$:BRANCH IF NO
7766	044674	100005				TST @STMP0 :ROUND ROBIN
7767	044676	005737	001302			BEQ 62\$:BRANCH IF YES
7768	044702	001402				JMP @15\$
7769	044704	000137	045652			62\$: BISB @STMP1+1,SYSSIZE+1(R3) :SET UNIT USED BIT
7770	044710	153763	001305	001643		CLR R2
7771	044716	005002				30\$: ROR @STMP1 :ENCODE THE BIT POSITION
7772	044720	006037	001304			INC R2 :INTO A UNIT NUMBER
7773	044724	005202				BCC 30\$
7774	044726	103374				DEC R2
7775	044730	005302				MOV R2,@UNITNO :SAVE UNIT NUMBER
7776	044732	010237	001602			10\$: MOV @IOWC,RP3HWC(R3) :GIVE WORD COUNT TO HANDLER
7777	044736	013763	001574	001756		MOV R3,R4
7778	044744	010304				MOV #3,R4 :ENCODE DEVICE FOR RUNTABLE
7779	044746	072427	000003			ASH @UNITNO,R4 :ENCODE UNIT NUMBER
7780	044752	053704	001602			ASL R4
7781	044756	006304				ADD #2,@RNTBINX :INCREMENT RUN TABLE INDEX
7782	044760	062737	000002	001604		MOV @RNTBINX,R2 :GET RUN TABLE INDEX
7783	044766	013702	001604			MOVB R4,RUNTABL+1(R2) :ENTER DEV & UNIT IN TABLE
7784	044772	110462	001663			MOV @UNITNO,RP3UNIT(R3) :GIVE HANDLER UNIT NUMBER
7785	044776	013763	001602	002052		MOV #PR5,@RESVEC+2 :SETUP RESERVED VECTOR PSW
7786	045004	012737	000240	000012		MOV RP3HANA(R3),@RESVEC :SETUP RESERVED VECTOR
7787	045012	016337	002066	000010		ASL R3 :ADJUST INDEX
7788	045020	006303				MOV @OLDBASE,RP3OLD(R3) :GIVE HANDLER OLD BASE ADDRESS
7789	045022	013763	001566	001772		MOV @NBASL,RP3NWL(R3) :GIVE HANDLER
7790	045030	013763	001570	002022		MOV @NBASH,RP3NWH(R3) :NEW BASE ADDRESS
7791	045036	013763	001572	002024		

7792	045044	005063	001774		CLR	RP3OLD+2(R3)	:ENSURE OLD BASE HIGH IS CLR
7793	045050	000010			CALL	HANDLER	
7794	045052	105737	001531		TSTB	@MMON	:IS MEMORY MANAGEMENT ON?
7795	045056	001416			BEQ	13\$:BRANCH IF NO
7796	045060	022737	000014	001604	CMP	#14,@RNTBINX	:TRANSFERED 14K YET?
7797	045066	001412			BEQ	13\$:BRANCH IF YES
7798	045070	062737	010000	001566	ADD	#10000,@OLDBASE	:ADD 2K
7799	045076	062737	010000	001570	ADD	#10000,@NWBASL	:TO BASE
7800	045104	005537	001572		ADC	@NWBASH	:ADDRESSES
7801	045110	000137	044260		JMP	@41\$	
7802	045114	113705	001631		MOV	@LTICKS+1,R5	:GET SECOND COUNT
7803	045120	062705	000002		ADD	#2,R5	:INCREMENT BY TWO
7804	045124	162705	000074		SUB	#60.,R5	:ENSURE RESULT IS 59 OR LESS
7805	045130	100002			BPL	31\$	
7806	045132	062705	000074		ADD	#60.,R5	:COUNT WAS LESS THAN 58-RESTORE
7807	045136	012700	000010		MOV	#10,R0	:SET SOB COUNT
7808	045142	005002			CLR	R2	
7809	045144	005003			CLR	R3	
7810	045146	005004			CLR	R4	
7811	045150	066203	001736		ADD	RP3HSTAT(R2),R3	:ADD ALL THE HANDLER
7812	045154	005504			ADC	R4	:STATUS WORDS. WHEN ALL
7813	045156	062702	000002		ADD	#2,R2	:TRANSFERS ARE FINISHED
7814	045162	077006			SOB	R0,14\$:RESULT WILL BE 2000
7815	045164	006103			ROL	R3	: (WITHOUT ROTATE)
7816	045166	005504			ADC	R4	
7817	045170	022703	004000		CMP	#4000,R3	:ALL DONE?
7818	045174	001406			BEQ	32\$:BRANCH IF YES
7819	045176	123705	001631		CMPB	@LTICKS+1,R5	:TWO SECONDS ELAPSED YET?
7820	045202	001355			BNE	31\$:BRANCH IF NO
7821	045204	104015			ERROR	15	:DEVICE HUNG
7822	045206	000177	134000		JMP	@SLPERR	:RESTART RELOCATION
7823	045212	005704			TST	R4	:ANY DEVICE ERRORS?
7824	045214	001402			BEQ	82\$:BRANCH IF NO
7825	045216	000167	000430		JMP	15\$:ERROR
7826	045222	105737	001531		TSTB	@MMON	:MEM MGMT ON?
7827	045226	001012			BNE	25\$:BRANCH IF YES
7828	045230	013705	001566		MOV	@OLDBASE,R5	:SETUP R5 FOR DATA CHECK
7829	045234	010500			MOV	R5,R0	
7830	045236	063700	001536		ADD	@\$FACTOR,R0	:GET LAST ADDRESS
7831							:OF GOOD DATA
7832	045242	013702	001570		MOV	@NWBASL,R2	
7833	045246	063702	001536		ADD	@\$FACTOR,R2	:GET LAST ADDRESS
7834							:OF DATA TO BE CHECKED
7835	045252	000411			BR	22\$:CONTINUE
7836	045254	012700	070000		MOV	#70000,R0	:GET LAST ADR + 2 OF GOOD DATA
7837	045260	012702	110000		MOV	#110000,R2	:GET LAST ADR + 2 OF DATA TO BE CHECKED
7838	045264	013737	172352	172350	MOV	@KIPAR5,@KIPAR4	:SET UP PAR4 FOR TOP 4K BANK
7839	045272	012705	002344		MOV	#\$ERRTB,R5	:DON'T CHECK BELOW \$ERRTB
7840	045276	004737	062534		JSR	PC,@CHKDAT	:GO CHECK DATA
7841	045302	102413			BVS	81\$:BRANCH IF ERROR
7842	045304	105737	001204		TSTB	@\$ERFLG	:ANY ERRORS?
7843	045310	001002			BNE	83\$:BRANCH IF YES
7844	045312	000167	000462		JMP	EXIT	:RETURN
7845	045316	032737	001000	177570	BIT	#SW9,@\$WR	:LOOP ON ERROR?
7846	045324	001473			BEQ	100\$+2	:BRANCH IF NO
7847	045326	000167	000244		JMP	20\$:GO DO FUNCTION AGAIN

7848	045332	005001		81\$:	CLR	R1		
7849	045334	010037	001302		MOV	R0,@\$STMP0		
7850	045340	010237	001522		MOV	R2,@\$VADR		
7851	045344	010003			MOV	R0,R3	;SAVE ERROR ADDRESS	
7852	045346	005004			CLR	R4		
7853	045350	105737	001531		TSTB	@\$MMON	;IS MEM MGMT ON?	
7854	045354	001406			BEQ	16\$;BRANCH IF NO	
7855	045356	162703	010000	17\$:	SUB	#10000,R3	;SUBTRACT 2K FROM ERROR ADDRESS	
7856	045362	100403			BMI	16\$;BRANCH IF BLOCK IS FOUND	
7857	045364	062704	000002		ADD	#2,R4	;COUNT ONE MORE BLOCK	
7858	045370	000772			BR	17\$;CONTINUE	
7859							;R4 NOW CONTAINS INDEX OF ERROR FOR RUN TIME TABLE	
7860	045372	116404	001663	16\$:	MOVB	RUNLBL+1(R4),R4	;GET DEVICE THAT FAILED	
7861	045376	042704	177400		BIC	#177400,R4	;ENSURE HIGH BYTE CLEAR	
7862	045402	006204			ASR	R4	;THROW AWAY LSB	
7863	045404	005005			CLR	R5	;ENSURE R5 CLEAR	
7864	045406	073427	177775		ASHC	#-3,R4	;GET UNIT NUMBER IN R5	
7865	045412	010500			MOV	R5,R0		
7866	045414	072027	177763		ASH	#-15,R0		
7867	045420	042700	177770		BIC	#177770,R0		
7868	045424	010037	001310		MOV	R0,@\$STMP3		
7869	045430	010403			MOV	R4,R3	;AND DEVICE INDEX IN R4 & R5	
7870	045432	010337	001306		MOV	R3,@\$STMP2		
7871	045436	012737	000001	001304	MOV	#1,@\$STMP1	;ENCODE 3 BIT UNIT NO INTO	
7872	045444	162705	020000	19\$:	SUB	#20000,R5	;ONE BIT IN THE LOW BYTE OF \$TMP1	
7873	045450	103403			BCS	18\$;BRANCH IF DONE	
7874	045452	006137	001304		ROL	@\$STMP1	;SELECT NEXT UNIT	
7875	045456	000772			BR	19\$;CONTINUE	
7876	045460	012737	045576	001212	18\$:	MOV	#20\$,@\$SLPERR	;SETUP LOOP RETURN
7877	045466	005701			TST	R1	;DEVICE ERROR?	
7878	045470	001010			BNE	100\$;BRANCH IF YES	
7879	045472	104010			ERROR	10	;DATA CHECK ERROR	
7880	045474	105737	001531		TSTB	@\$MMON	;MGMT ON?	
7881	045500	001002			BNE	70\$;BRANCH IF YES	
7882	045502	000137	044240	71\$:	JMP	@\$ENTER2		
7883	045506	000137	044160	70\$:	JMP	@\$IOMON		
7884	045512	104007		100\$:	ERROR	7		
7885	045514	042763	100000	001736	BIC	#BIT15,RP3HSTAT(R3)	;CLEAR THE ERROR	
7886	045522	022703	000002		CMP	#2,R3	;RK05 ERROR?	
7887	045526	002405			BLT	90\$;BRANCH IF RH70	
7888	045530	003016			BGT	92\$;BRANCH IF RPO3	
7889	045532	112777	000001	134426	MOVB	#1,@\$RKCS	;RK CONTROLLER CLEAR	
7890	045540	000412			BR	92\$		
7891	045542	022703	000012	90\$:	CMP	#12,R3	;RS04?	
7892	045546	001004			BNE	91\$;BRANCH IF NO	
7893	045550	052777	000040	134476	BIS	#BIT5,@\$RSCS2	;CLEAR RS CONTROLLER	
7894	045556	000403			BR	92\$		
7895	045560	052777	000040	134426	91\$:	BIS	#BIT5,@\$RP4CS2	;CLEAR RP04 CONTROLLER
7896	045566	105737	001531	92\$:	TSTB	@\$MMON	;MGMT ON?	
7897	045572	001345			BNE	70\$;BRANCH IF YES	
7898	045574	000742			BR	71\$		
7899	045576	052763	000400	001736	20\$:	BIS	#BIT8,RP3HSTAT(R3)	;SET REPEAT FLAG IN HANDLER
7900	045604	016337	002066	000010	MOV	RP3HANA(R3),@\$RESVEC	;SETUP RESERVED INSTRUCTION VECTOR	
7901	045612	000010			CALLHANDLER			
7902	045614	105763	001736	21\$:	TSTB	RP3HSTAT(R3)	;HANDLER FINISHED?	
7903	045620	100375			BPL	21\$;BRANCH IF NO	

```

7904 045622 005763 001736      TST      RP3HSTAT(R3)      ;ANY ERROR?
7905 045626 100714              BMI      18$              ;BRANCH IF YES
7906 045630 005701              TST      R1               ;DEVICE ERROR?
7907 045632 001002              BNE      80$              ;BRANCH IF YES
7908 045634 000167 177356      JMP      32$+4            ;GO CHECK DATA
7909 045640 032737 001000 177570 80$: BIT      @#BIT9,@#SWR    ;STILL LOOPING?
7910 045646 001353              BNE      20$              ;BRANCH IF YES
7911 045650 000721              BR       100$+2           ;CONTINUE TEST
7912                                ;THE FOLLOWING CODE HANDLES DEVICE ERROR ON RELOCATION
7913 045652 005004      15$: CLR      R4              ;SET INDEX
7914 045654 010601              MOV      SP,R1
7915 045656 005764 001700      24$: TST      RUNTRAK(R4)   ;SEARCH FOR DEVICE ERROR
7916 045662 100643              BMI      16$              ;BRANCH IF ERROR
7917 045664 062704 000002      ADD      #2,R4            ;INCREMENT INDEX
7918 045670 000772              BR       24$              ;CONTINUE SEARCH
7919                                ;RELOCATE BY CPU-MEMORY MANAGEMENT ON
7920 045672 012703 010000      RELOCP: MOV     #4096.,R3   ;4K COUNTER
7921 045676 012022      1$: MOV      (R0)+,(R2)+   ;RELOCATE CODE
7922 045700 077302              SOB      R3,1$           ;BR IF NOT DONE 4K WORDS
7923 045702 023737 172350 172352      CMP      @#KIPAR4,@#KIPAR5 ;DONE 16K YET?
7924 045710 001414              BEQ      2$              ;BR IF DONE
7925 045712 062737 000200 172350      ADD      #200,@#KIPAR4    ;MAP TO NEXT 4K SPACE
7926 045720 012702 100000      MOV      #100000,R2       ;MAP WITH R2 (PAR4)
7927 045724 023737 172350 172352      CMP      @#KIPAR4,@#KIPAR5 ;DOING LAST 4K BANK?
7928 045732 001357              BNE      RELOCP          ;BR IF NOT
7929 045734 012703 004000      MOV      #2048.,R3       ;2K COUNTER
7930 045740 000756              BR       1$              ;RELOCATE LAST 2K ONLY (14K TOTAL)
7931 045742 012705 001736      2$: MOV      #RP3HSTAT,R5  ;DON'T CHECK BELOW RP3HSTAT
7932 045746 004737 062534      JSR      PC,@#CHKDAT     ;CHECK DATA
7933 045752 102012              BVC      EXIT
7934 045754 010037 001302      MOV      R0,@#STMP0
7935 045760 010237 001522      MOV      R2,@#VADR
7936 045764 104006              ERROR    6
7937 045766 013737 001546 172350      MOV      @#NEXPAR,@#KIPAR4 ;RESTORE PAR4
7938 045774 000167 175672      JMP      RETRY
7939 046000 105737 001531      EXIT:  TSTB     @#MMON    ;MEM MGMT ON?
7940 046004 001002              BNE      .+6             ;BRANCH IF YES
7941 046006 000137 044152      JMP      @#EXITRE
7942 046012 062737 000077 001546      ADD      #77,@#NEXPAR    ;SET VALUE FOR NEXT RELOCATION
7943 046020 013737 172350 172340      MOV      @#KIPAR4,@#KIPAR0
7944 046026 063737 172350 172342      ADD      @#KIPAR4,@#KIPAR1
7945 046034 063737 172350 172344      ADD      @#KIPAR4,@#KIPAR2
7946 046042 063737 172350 172346      ADD      @#KIPAR4,@#KIPAR3
7947                                ;*****
7948                                ;PROGRAM IS NOW EXECUTING IN KERNEL MODE RELOCATED TO ADDRESS AS SPEC-
7949                                ;IFIED IN KIPAR0. FOR EX. IF KIPAR0=1600 THEN PROGRAM EXECUTING AT
7950                                ;ADDRESS 160000+(PC)
7951 046050 013700 172340      MOV      @#KIPAR0,R0     ;GET PAR0
7952 046054 072027 177771      ASH      #-7,R0         ;GET BITS <14:7> IN LOW BYTE
7953 046060 110037 001203      MOV      R0,@#STSTNM+1  ;PUT IN DSPLAY REG HIGH BYTE
7954 046064 012706 001200      MOV      #KERSTK,SP     ;SET KERNEL STACK PTR
7955 046070 005037 177776      CLR      @#PSW
7956 046074 016746 175204      MOV      OLDPSW,-(SP)   ;RESTORE OLD PSW
7957 046100 012746 010000      MOV      #LOOP,-(SP)
7958 046104 105737 001530      TSTB     @#NEXEC       ;BRANCH IF TEST CODE TO
7959 046110 001402              BEQ      1$              ;BE EXECUTED

```

```

7960 046112 012716 043250          MOV    #STMM,(SP)
7961 046116 000002          1$:   RTI                      ;RESTART PROGRAM AT LOOP
7962
7963          ;WHEN RELOCATION ABOVE 28K IS COMPLETE PROGRAM TRAPS TO ENDMEM.
7964 046120 022626          ENDMEM: CMP    (SP)+,(SP)+      ;POP STACK TWICE
7965 046122 005037 177572          ENDM:  CLR    @#SRO             ;DISABLE MEM MGMT
7966 046126 042737 000020 172516          BIC    #BIT4,@#MMR3          ;CLEAR 22 BIT MODE
7967
7968          ;*****
7969          ;AT THIS TIME A 'SUB-PASS' HAS BEEN COMPLETED.
7970          ;PROGRAM NOW EXECUTING IN KERNEL MODE AT PC AS SHOWN (NO RELOCATION)
7971 046134 012737 000700 001546          MOV    #700,@#NEXPAR        ;RESET NEXT VALUE FOR PAR REGISTERS
7972 046142 005737 003574          TST    @#PROT
7973 046146 001403          BEQ    2$
7974 046150 012737 001600 001546          MOV    #1600,@#NEXPAR
7975 046156 105037 001531          2$:   CLR    @#MMON             ;SET MEM MGMT ON IND - OFF
7976          ;*****
7977          ;SBTTL END OF SUB-PASS ROUTINE
7978          ;* THIS ROUTINE SETS UP THE PSW AND MAINTENANCE REGISTERS
7979          ;* FOR THE NEXT SUB-PASS. IT THEN STARTS THE PRINTER
7980          ;* (IF NOT ON ACT11) FOR TYPING THE END OF SUB-PASS MESSAGE.
7981          ;*****
7982          END:
7983 046162 012737 064270 000004          END1:  MOV    #ERPRT,@#ERRVEC
7984 046170 012737 063464 000114          MOV    #.PARSRV,@#CACHVEC    ;RESTORE CACHE PARITY VECTOR
7985 046176 010137 000116          MOV    R1,@#CACHVEC+2        ;AND PSW
7986 046202 005037 177776          CLR    @#PSW                 ;CLEAR MODE BITS IN PSW
7987 046206 004767 014750          JSR    PC,CLRTBIT            ;GO CLEAR 'T' BIT IF SET
7988 046212 012706 001200          MOV    #KERSTK,SP           ;SET KERNEL STACK PTR
7989 046216 032777 000100 133020          BIT    #100,@#STPS          ;CHECK IF OUTPUT DEVICE IS BUSY
7990 046224 001374          BNE    -6                   ;IS AVAILABLE
7991 046226 105237 001560          1$:   INCB   @#SUBPASS
7992 046232 113702 001560          MOVB   @#SUBPASS,R2
7993 046236 162702 000060          SUB    #6,R2
7994 046242 022702 000006          CMP    #6,R2                ;END OF TEST?
7995 046246 001013          BNE    2$                   ;BRANCH IF NOT AT END
7996 046250 012737 000060 001560          MOV    #60,@#SUBPASS        ;INIT SUBPASS COUNT TO ASCII 0
7997 046256 005037 177750          CLR    @#MAINT              ;CLEAR MAINTENANCE REG
7998 046262 005037 001640          CLR    @#SMAINT            ;CLEAR SOFTWARE VALUE
7999 046266 005046          CLR    -(SP)
8000 046270 012746 046376          MOV    #SEOP,-(SP)
8001 046274 000002          RTI
8002 046276 006302          2$:   ASL    R2
8003 046300 012737 001520 000014          MOV    #SRTRN,@#BITVEC      ;SET 'T' TRAP VECTOR
8004 046306 012737 001557 001270          MOV    #SUBPASS-,@#SREG5
8005 046314 106277 132724          ASRB   @#STPS
8006 046320 016246 064364          MOV    PSWTAB(2),-(SP)      ;PUSH NEXT PASS PSW ON STACK
8007 046324 012746 010000          MOV    #LOOP,-(SP)         ;RESART PROGRAM AT LOOP
8008 046330 016237 064400 001640          MOV    MRGTAB(R2),@#SMAINT
8009 046336 016237 064400 177750          MOV    MRGTAB(R2),@#MAINT
8010 046344 105737 001532          3$:   TST    @#OV                ;OV PASS?
8011 046350 001011          BNE    RTI1                 ;BRANCH IF YES
8012 046352 122777 000200 132664          CMPB   #200,@#STPS          ;IS PRINTER READY?
8013 046360 001371          BNE    3$                   ;BRANCH IF NO
8014 046362 012737 065313 001270          MOV    #MSG20-1,@#SREG5
8015 046370 106277 132650          ASRB   @#STPS                ;TYPE END SUBPASS MESSAGE
    
```

```
8016 046374 000002 RTI1: RTI ;RESTART PROGRAM AT LOOP WITH NEW PSW
8017 ;(FROM TABLE BELOW)
8018
8019 ;:*****
8020
8021 .SBTTL END OF PASS ROUTINE
8022
8023 ;*INCREMENT THE PASS NUMBER ($PASS)
8024 ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
8025 ;*WHERE XXXXX AND YYYYYY ARE DECIMAL NUMBERS
8026 ;*IF THERES A MONITOR GO TO IT
8027 ;*IF THERE ISN'T JUMP TO LOOP
8028
8029 $EOP:
8030 046376 004737 056506 JSR PC,@TYPTIME
8031 046402 005067 132574 CLR $STNM ;:ZERO THE TEST NUMBER
8032 046406 005067 132714 CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
8033 046412 005267 132562 INC $PASS ;:INCREMENT THE PASS NUMBER
8034 046416 042767 100000 132554 BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
8035 046424 005327 DEC (PC)+ ;:LOOP?
8036 046426 000001 $EOPCT: .WORD 1
8037 046430 003063 BGT $DOAGN ;:YES
8038 046432 012737 MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
8039 046434 000001 $ENDCT: .WORD
8040 046436 046426 $EOPCT
8041 046440 104400 046446 TYPE ,65$ ;:TYPE ASCIZ STRING
8042 046444 000407 BR 64$ ;:GET OVER THE ASCIZ
8043 ;:65$: .ASCIZ <12><15>/END PASS #/
8044 64$:
8045 046464 016746 132510 MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
8046 ;:TYPE PASS NUMBER
8047 046470 104410 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
8048 046472 104400 046500 TYPE ,67$ ;:TYPE ASCII STRING
8049 046476 000421 BR 66$ ;:GET OVER ASCII
8050 ;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
8051 66$:
8052 046542 016746 132446 MOV $ERTTL,-(SP) ;:SAVE $ERTTL FOR TYPEOUT
8053 ;:TOTAL NUMBER OF ERRORS
8054 046546 104410 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
8055 046550 104400 001337 TYPE ,SRLF ;:TYPE CARRIAGE RETURN, LINE FEED
8056 046554 005067 132434 CLR $ERTTL ;:CLEAR ERROR TOTAL
8057 046560 013700 000042 $GET42: MOV @42,R0 ;:GET MONITOR ADDRESS
8058 046564 001405 BEQ $DOAGN ;:BRANCH IF NO MONITOR
8059 046566 000005 RESET ;:CLEAR THE WORLD
8060 046570 004710 $ENDAD: JSR PC,(R0) ;:GO TO MONITOR
8061 046572 000240 NOP ;:SAVE ROOM
8062 046574 000240 NOP ;:FOR
8063 046576 000240 NOP ;:ACT11
8064 046600 $DOAGN:
8065 046600 000137 010000 JMP @LOOP ;:RETURN
8066 046604 377 377 000 $ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
8067 046610 .EVEN
8068 ;:*****
8069 .SBTTL RP11/RP03 HANDLER
8070 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
8071 ;:*****
```

8072	046610	104420			RP3DRV: SAVREG		
8073	046612	105037	001736		CLRB	@#RP3HSTA	:CLEAR DONE FLAG
8074	046616	032737	000400	001736	BIT	#BIT8,@#RP3HSTA	:REPEAT FLAG SET?
8075	046624	001403			BEQ	8\$:BRANCH IF NO
8076	046626	104422			RESREG		
8077	046630	000137	050462		JMP	@#RP3RPT	
8078	046634	013737	001604	001614	8\$: MOV	@#RNTBINX,@#RP311	:SAVE RUN TABLE INDEX
8079	046642	032737	000020	177570	BIT	#SW4,@#SWR	:INHIBIT RND DSK ADR?
8080	046650	001403			BEQ	1\$:BRANCH IF NO
8081	046652	005000			CLR	R0	
8082	046654	005001			CLR	R1	
8083	046656	000410			BR	4\$	
8084	046660	004737	060436		1\$: JSR	PC,@#SRAND	:GO GET RANDOM NUMBER
8085	046664	013700	001552		MOV	@#SHINUM,R0	:GET HI NUMBER
8086	046670	013701	001550		MOV	@#SLONUM,R1	:GET LO NUMBER
8087	046674	073027	177771		ASHC	#-7,R0	:ADJUST TO FORM CYL ADR
8088	046700	042700	177000		4\$: BIC	#177000,R0	:GET RID OF UNUSED BITS
8089	046704	022700	000624		CMP	#624,R0	:LEGAL CYL?
8090	046710	100003			BPL	5\$:BRANCH IF YES
8091	046712	062700	000624		ADD	#624,R0	:MAKE IT LEGAL
8092	046716	000770			BR	4\$	
8093	046720	013702	001614		5\$: MOV	@#RP311,R2	:GET RUN TABLE INDEX
8094	046724	016203	001662		MOV	RUNTABL(R2),R3	:GET DEVICE ID
8095	046730	042703	000777		BIC	#777,R3	:ID ONLY
8096	046734	050300			BIS	R3,R0	:COMBINE WITH CYL ADR
8097	046736	010062	001662		MOV	R0,RUNTABL(R2)	:PUT BACK IN TABLE
8098	046742	072127	177775		ASH	#-3,R1	:GEN TRK-SECT ADR
8099	046746	010103			MOV	R1,R3	:SAVE
8100	046750	042701	160377		6\$: BIC	#160377,R1	:GET RID OF ALL BUT TRK
8101	046754	022701	011400		CMP	#11400,R1	:LEGAL TRAK?
8102	046760	100003			BPL	2\$:BRANCH IF YES
8103	046762	062701	011400		ADD	#11400,R1	:MAKE IT LEGAL
8104	046766	000770			BR	6\$	
8105	046770	042703	177760		2\$: BIC	#177760,R3	:GET SECTOR ADR
8106	046774	022703	000011		CMP	#11,R3	:IS IT LEGAL?
8107	047000	100003			BPL	3\$:BRANCH IF YES
8108	047002	062703	000011		ADD	#11,R3	:MAKE IT LEGAL
8109	047006	000770			BR	2\$	
8110	047010	050301			3\$: BIS	R3,R1	:COMBINE TRK-SECT
8111	047012	010162	001700		MOV	R1,RUNTRAK(R2)	:PUT IN TABLE
8112	047016	010037	002104		MOV	R0,@#RP3HDC	:SAVE DESIRED CYL
8113	047022	010137	002152		MOV	R1,@#RP3DA	:SAVE DSK ADR
8114	047026	112737	177775	002132	MOV8	#-3,@#RP3TRY	:INIT TRY COUNT
8115	047034	032737	000040	172516	BIT	#BIT5,@#MMR3	:MAP ON?
8116	047042	001405			BEQ	7\$:BRANCH IF NO
8117	047044	005046			CLR	-(SP)	:PUT DEVICE ID ON STACK
8118	047046	013746	001772		MOV	@#RP3OLD,-(SP)	:PUT ADR OF BUS ADR ON STK
8119	047052	004737	062676		JSR	PC,@#GETMAP	:GET MAP REGISTER
8120	047056	012737	000103	001612	7\$: MOV	#103,@#RP310	:GET FUNCTION
8121	047064	013700	001774		MOV	@#RP3OLD+2,R0	:GET BAE BITS
8122	047070	072027	000004		ASH	#4,R0	:SHIFT TO BITS 4 & 5
8123	047074	050037	001612		BIS	R0,@#RP310	:COMBINE WITH FUNCTION
8124	047100	010037	001774		MOV	R0,@#RP3OLD+2	
8125	047104	013700	002052		MOV	@#RP3UNIT,R0	
8126	047110	072027	000010		ASH	#10,R0	:SHIFT UNIT NO TO RIGHT BITS
8127	047114	050037	001612		BIS	R0,@#RP310	:COMBINE WITH FUNC & BAE


```
8184 047412 062700 000012          ADD    #12,R0          ;MAKE IT LEGAL
8185 047416 042700 000020          BIC    #BIT4,R0       ;GET RID OF CARRY FROM ADD
8186 047422 042703 000017 1$:  BIC    #17,R3        ;GET SURFACE ADDRESS
8187 047426 050300                BIS    R3,R0         ;GENER COMP SECT-SURF ADDRESS
8188 047430 010062 001700          MOV    R0,RUNTRAK(R2) ;SAVE IN RUN TRAK TABLE
8189 047434 072127 000005          ASH   #5,R1          ;ADJUST CYLINDER ADDRESS
8190 047440 050100                BIS    R1,R0         ;CONCATINATE TRK & SECT ADDR
8191 047442 013701 002054          MOV    @RKUNIT,R1    ;GET UNIT NUMBER
8192 047446 072127 000015          ASH   #15,R1         ;ADJUST
8193 047452 050100                BIS    R1,R0         ;CONCATINATE UNIT,TRK,SURF,SECT
8194 047454 010037 002106          MOV    R0,@RKHDA     ;SAVE
8195 047460 112737 177775 002133  MOVB   #-3,@RKTRY    ;SET RETRY COUNT
8196 047466 032737 000040 172516  BIT    #BIT5,@MMR3   ;MAP ON?
8197 047474 001406                BEQ   2$            ;BRANCH IF NO
8198 047476 012746 000001          MOV    #1,-(SP)     ;PUT DEVICE ID ON STACK
8199 047502 012746 001776          MOV    #RKOLD,-(SP) ;PUT ADDRESS OF ADR ON STACK
8200 047506 004737 062676          JSR   PC,@GETMAP    ;GET MAP REG
8201 047512 012767 000103 132076 2$:  MOV    #103,RK10    ;SET FUNCTION
8202 047520 013700 002000          MOV    @RKOLD+2,R0  ;GET BA EXTENDED
8203 047524 072027 000004          ASH   #4,R0         ;ADJUST
8204 047530 050037 001616          BIS    R0,@RK10     ;PUT IN WITH FUNCTION
8205 047534 010037 002000          MOV    R0,@RKOLD+2  ;SAVE IN MEMORY
8206 047540 104422                RESREG
8207 047542 013777 002106 132424  RKWTRY: MOV   @RKHDA,@RKDA ;LOAD DISK ADDRESS
8208 047550 032777 000100 132404  BIT    #BIT6,@RKDS  ;UNIT READY?
8209 047556 001774                BEQ   .-6           ;BRANCH IF NO
8210 047560 013777 001760 132402  MOV    @RKHWC,@RKWC ;LOAD WORD COUNT
8211 047566 013777 001776 132376  MOV    @RKOLD,@RKBA ;LOAD BUS ADDRESS
8212 047574 012777 051330 132374  MOV    #RKSrv,@RKVEC ;LOAD INTERRUPT VECTOR
8213 047602 005077 132372          CLR   @RKPSW        ;
8214 047606 005037 002122          CLR   @RKFUN        ;SET FUNCTION TO WRITE
8215 047612 013777 001616 132346  MOV    @RK10,@RKCS  ;LOAD FUNCTION AND GO
8216 047620 000006                RTT                ;RETURN
```

```
8217
8218
8219
8220
8221
8222 047622 104420
8223 047624 105037 001746
8224 047630 032737 000400 001746
8225 047636 001403
8226 047640 104422
8227 047642 000137 052156
8228 047646 013737 001604 001622 6$:  JMP    @RP4RPT
8229 047654 105037 001746          CLR   @RP4HSTA     ;CLEAR DONE FLAG
8230 047660 032737 000020 177570  BIT    #SW4,@SWR    ;REPEAT FLAG SET?
8231 047666 001403                BEQ   1$            ;BRANCH IF NO
8232 047670 005000                CLR   R0
8233 047672 005001                CLR   R1
8234 047674 000410                BR    4$
8235 047676 004737 060436 1$:  JSR   PC,@$SRAND    ;GET RANDOM NUMBER
8236 047702 013700 001552          MOV    @$HINUM,R0   ;GET HI NUMBER
8237 047706 013701 001550          MOV    @$LONUM,R1   ;GET LO NUMBER
8238 047712 073027 177771          ASHC  #-7,R0        ;ADJUST TO FORM CYL. ADH.
8239 047716 042700 177000 4$:  BIC    #177000,R0   ;GET RID OF UNUSED BITS
```


8240	047722	022700	000631				
8241	047726	100003		OMP	#631,R0		:LEGAL CYLINDER
8242	047730	062700	000631	BPL	SS		:BRANCH IF YES
8243	047734	000770		ADD	#631,R0		:MAKE IT LEGAL
8244				BR	4S		
8245	047736	013702	001622	SS	MOV	@RP411,R2	:GET RUN TABLE INDEX

```
8246 047742 016203 001662      MOV      RUNTBL(R2),R3      ;GET DEVICE ID
8247 047746 042703 000777      BIC      #777,R3          ;SAVE ID ONLY
8248 047752 050003              BIS      R0,R3            ;COMBINE WITH CYL ADR
8249 047754 010362 001662      MOV      R3,RUNTBL(R2)    ;PUT IN RUN TABLE
8250 047760 072127 177775      ASH      #-3,R1           ;GEN TRAK-SECT ADR
8251 047764 042701 160340      BIC      #160340,R1       ;GET RID OF UNUSED BITS
8252 047770 010103              MOV      R1,R3            ;SAVE
8253 047772 042701 000037      BIC      #37,R1           ;GET RID OF SECT BITS
8254 047776 022701 011000      CMP      #11000,R1        ;LEGAL TRAK?
8255 050002 100004              BPL      2$                ;BRANCH IF YES
8256 050004 062701 011000      ADD      #11000,R1        ;MAKE IT LEGAL
8257 050010 042701 020000      BIC      #BIT13,R1        ;GET RID OF ADD CARRY
8258 050014 042703 177740      2$:     BIC      #177740,R3    ;GET SECTOR ADR
8259 050020 022703 000025      CMP      #25,R3          ;LEGAL SECTOR
8260 050024 100004              BPL      3$                ;BRANCH IF YES
8261 050026 062703 000025      ADD      #25,R3          ;MAKE IT LEGAL
8262 050032 042703 000040      BIC      #BIT5,R3         ;GET RID OF ADD CARRY
8263 050036 050301              3$:     BIS      R3,R1         ;COMBINE TRAK-SECTOR
8264 050040 010162 001700      MOV      R1,RUNTRAK(R2)  ;PUT TRAK-SECT IN TABLE
8265 050044 010037 002114      MOV      R0,@RP4HDC      ;SAVE CYLINDER ADR
8266 050050 010137 002112      MOV      R1,@RP4HDA      ;SAVE TRAK-SECTOR ADR
8267 050054 112737 177775 002135      MOV      #-3,@RP4TRY     ;SET TRY COUNT
8268 050062 104422              RESREG
8269 050064 004767 000026      RP4WTRY: JSR      PC,LDRP4 ;LOAD RP4 REGISTERS
8270 050070 012777 052202 132140      MOV      #RP4SRV,@RP4VEC ;LOAD INTERRUPT VECTOR
8271 050076 005077 132136      CLR      @RP4PSW
8272 050102 005037 002126      CLR      @RP4FUN
8273 050106 112777 000161 132066      MOV      #161,@RP4CS1   ;SET FUNCTION TO WRITE
8274 050114 000002              RTI                       ;LOAD FUNCTION AND GO
8275                                     ;RETURN
8276 050116 013777 002062 132070      LDRP4:  MOV      @RP4UNIT,@RP4CS2 ;LOAD UNIT NUMBER
8277 050124 012777 010000 132102      MOV      #BIT12,@RP4OF   ;SET FORMAT TO 16 BIT
8278 050132 013777 002114 132064      MOV      @RP4HDC,@RP4DC  ;LOAD CYLINDER ADR
8279 050140 013777 002112 132044      MOV      @RP4HDA,@RP4DA  ;LOAD TRAK-SECTOR
8280 050146 013777 001766 132030      MOV      @RP4HWC,@RP4WC  ;LOAD WORD COUNT
8281 050154 013777 002014 132026      MOV      @RP4OLD+2,@RP4BAE ;LOAD EXTENDED ADR BITS
8282 050162 013777 002012 132016      MOV      @RP4OLD,@RP4BA  ;LOAD BUS ADR
8283 050170 000207              RTS      PC                ;RETURN
8284
8285 ;:*****
8286 ;.SBTTL RH70/RS04 HANDLER
8287 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
8288 ;:*****
8289 050172 104420              RSDRV:  SAVREG
8290 050174 105037 001750      CLR      @RSHSTAT        ;CLEAR DONE FLAG
8291 050200 032737 000400 001750      BIT      #BIT8,@RSHSTAT  ;REPEAT FLAG SET?
8292 050206 001403              BEQ      3$                ;BRANCH IF NO
8293 050210 104422              RESREG
8294 050212 000137 052666      JMP      @RSRPT
8295 050216 013737 001604 001624 3$:     MOV      @RNTBINX,@RS11  ;SAVE RUN TABLE INDEX
8296 050224 032737 000020 177570      BIT      #SW4,@SWR
8297 050232 001403              BEQ      1$                ;RANDOM DSK ADR?
8298 050234 005000              CLR      R0                ;BRANCH IF YES
8299 050236 005001              CLR      R1
8300 050240 000407              BR      4$
8301 050242 004737 060436      1$:     JSR      PC,@SRAND        ;GET RANDOM NUMBER
```

```
8302 050246 013700 001552      MOV      @#SHINUM,R0
8303 050252 072027 177774      ASH      #-4,R0
8304 050256 010001                MOV      R0,R1          ;SAVE RANDOM NUMBER
8305 050260 042700 170077      4$: BIC      #170077,R0    ;GET TRACK ADR
8306 050264 022700 007600      CMP      #7600,R0      ;IS IT LEGAL?
8307 050270 100003                BPL      5$            ;BRANCH IF YES
8308 050272 062700 007600      ADD      #7600,R0      ;MAKE IT LEGAL
8309 050276 000770                BR       4$
8310 050300 013702 001624      5$: MOV      @#RS11,R2    ;GET RUN TABLE INDEX
8311 050304 072027 177772      ASH      #-6,R0        ;ADJUST TRACK ADR
8312 050310 110062 001662      MOVB     R0,RUNTABL(R2) ;SAVE TRAK ADR IN RUN TBL
8313 050314 042701 177700      6$: BIC      #177700,R1    ;GET SECTOR ADR
8314 050320 022701 000077      CMP      #77,R1        ;IS IT LEGAL?
8315 050324 100003                BPL      2$            ;BRANCH IF YES
8316 050326 062701 000077      ADD      #77,R1        ;MAKE IT LEGAL
8317 050332 000770                BR       6$
8318 050334 010162 001700      7$: MOV      R1,RUNTRAK(R2) ;SAVE IN RUN TRAK TABLE
8319 050340 072027 000006      ASH      #6,R0          ;ADJUST TRACK ADDR
8320 050344 050100                BIS      R1,R0          ;COMBINE SECTOR TRAK
8321 050346 010037 002116      MOV      R0,@#RSHDA    ;SAVE AS DSK ADR
8322 050352 112737 177775 002136  MOVB     #-3,@#RSTRY    ;SET TRY COUNT
8323 050360 104422                RESREG
8324 050362 004737 050422      RSWTRY: JSR      PC,@#LDRS ;GO LOAD REGISTERS
8325 050366 012777 052712 131670  MOV      #RSSRV,@#RSVEC ;SET INTERRUPT VECTOR
8326 050374 005077 131666      CLR      @#RSPSW
8327 050400 005037 002130      CLR      @#RSFUN
8328 050404 105777 131650      1$: TSTB     @#RSDS      ;SET FUNCTION TO WRITE
8329 050410 001775 131650                BEQ      1$            ;IS DRIVE READY?
8330 050412 112777 000161 131622  MOVB     #161,@#RSCS1  ;BRANCH IF NO
8331 050420 000002                RTI                    ;LOAD FUNCTION AND GO
8332
8333 050422 013777 002064 131624  LDRS:  MOV      @#RSUNIT,@#RSCS2 ;LOAD UNIT NUMBER
8334 050430 013777 002116 131614  MOV      @#RSHDA,@#RSDA ;LOAD DSK ADR
8335 050436 013777 001770 131600  MOV      @#RSHWC,@#RSWC ;LOAD WORD COUNT
8336 050444 013777 002020 131576  MOV      @#RSOLD+2,@#RSBAE ;LOAD EXTENDED ADDRESS
8337 050452 013777 002016 131566  MOV      @#RSOLD,@#RSBA ;LOAD BUS ADDRESS
8338 050460 000207                RTS                     ;RETURN
8339
8340
8341
8342
8343
8344 050462 000005                RP3RPT: RESET
8345 050464 005337 002120                DEC      @#RP3FUN      ;RESTORE FUNCTION
8346 050470 022737 000001 002120  CMP      #1,@#RP3FUN    ;WHAT IS IT?
8347 050476 001472                BEQ      RP31          ;BRANCH IF WC
8348 050500 002402                BLT      1$            ;BRANCH IF WRITE
8349 050502 000137 047126                JMP      @#RP3WTRY     ;BRANCH TO READ
8350 050506 000167 000414      1$: JMP      RP33
8351 050512 005237 002120      RP3SRV: INC      @#RP3FUN ;INCREMENT FUNCTION
8352 050516 022737 000002 002120  CMP      #2,@#RP3FUN    ;WHAT IS IT?
8353 050524 001501                BEQ      RP3WCK        ;BRANCH TO WRITE CHECK
8354 050526 100002                BPL      +6
8355 050530 000137 051166                JMP      @#RP3READ
8356
8357
```

;FUNCTION JUST EXECUTED WAS A WRITE

8358	050534	032737	000400	001736		BIT	#BIT8,@RP3HSTAT	:REPEAT FLAG SET?
8359	050542	001036				BNE	RP3LOOP	:BRANCH IF YES
8360	050544	005777	131374			TST	@RP3CS	:ANY ERRORS?
8361	050550	100045				BPL	RP31	:BRANCH IF NO
8362	050552	105737	002132			TSTB	@RP3TRY	:TRIED 3 TIMES?
8363	050556	001415				BEQ	RP3ERR	:BRANCH IF YES
8364	050560	112777	000001	131356		MOVB	#BIT0,@RP3CS	:CLEAR THE DRIVE
8365	050566	105777	131352			TSTB	@RP3CS	:CONTROLLER READY?
8366	050572	100375				BPL	.-4	:BRANCH IF NO
8367	050574	105237	002132			INCB	@RP3TRY	:INCREMENT TRY COUNT
8368	050600	013746	177776			MOV	@PSW,-(SP)	:MAINTAIN SAME PSW
8369	050604	012746	047126			MOV	@RP3WTRY,-(SP)	:SET RETRY ADDRESS
8370	050610	000002				RTI		:RETURN
8371	050612	012737	100200	001736	RP3ERR:	MOV	#100200,@RP3HSTA	:SET ERROR BIT IN HAND. STA
8372	050620	010046				MOV	RO,-(SP)	:SAVE RO
8373	050622	013790	001614			MOV	@RP311,RO	:GET RUNTABLE INDEX
8374	050626	052760	100000	001700		BIS	#BIT15,RUNTRAK(RO)	:SET ERROR BIT
8375	050634	012600				MOV	(SP)+,RO	:RESTORE RO
8376	050636	000002				R.I		:RETURN
8377								
8378	050640	012737	100200	001736	RP3LOOP:	MOV	#100200,@RP3HSTAT	:SET DONE AND ERROR
8379	050646	005777	131272			TST	@RP3CS	:ANY ERRORS?
8380	050652	100403				BMI	1\$:BRANCH IF YES
8381	050654	042737	100000	001736		BIC	#BIT15,@RP3HSTAT	:CLEAR ERROR BIT
8382	050662	000002			1\$:	RTI		:RETURN
8383								
8384	050664	112737	177775	002132				
8385	050672	012737	007107	001612	RP31:	MOVB	#-3,@RP3TRY	:INIT TRY COUNT
8386	050700	053737	001774	001612		MOV	#107,@RP310	:SET FUNCTION
8387	050706	053737	002052	001612		BIS	@RP3OLD+2,@RP310	:SET BAE BITS
8388	050714	004737	047174			BIS	@RP3UNIT,@RP310	:SET UNIT BITS
8389	050720	013777	001612	131216	RP32:	JCR	PC,@LDRP3	:LOAD RP3 REGISTERS
8390	050726	000002				MOV	@RP310,@RP3CS	:LOAD FUNCTION AND GO
8391						RTI		:RETURN
8392								
8393	050730	032737	000400	001736				
8394	050736	001340			RP3WCK:	BIT	#BIT8,@RP3HSTAT	:REPEAT FLAG SET?
8395	050740	005777	131200			BNE	RP3LOOP	:BRANCH IF YES
8396	050744	100031				TST	@RP3CS	:ANY ERRORS?
8397	050746	005737	001614			BPL	1\$:BRANCH IF NO
8398	050752	001422				TST	@RP311	:FIRST 2K?
8399	050754	105737	002132		5\$:	BEQ	4\$:BRANCH IF YES
8400	050760	001714				TSTB	@RP3TRY	:TRIED 3 TIMES?
8401	050762	005337	002120			BEQ	RP3ERR	:BRANCH IF YES
8402	050766	112777	000001	131150		DEC	@RP3FUN	:RESTORE FUNCTION
8403	050774	105777	131144			MOVB	#BIT0,@RP3CS	:CLEAR THE DRIVE
8404	051000	100375				TSTB	@RP3CS	:CONTROLLER READY?
8405	051002	105237	002132			BPL	.-4	:BRANCH IF NO
8406	051006	013746	177776			INCB	@RP3TRY	:INCREMENT TRY COUNT
8407	051012	012746	050714			MOV	@PSW,-(SP)	
8408	051016	000002				MOV	@RP32,-(SP)	
8409	051020	032777	000010	131114	4\$:	RTI		:GO TRY AGAIN
8410	051026	001752				BIT	#BIT3,@RP3ERR	:WRITE CHECK ERROR?
8411						BEQ	5\$:BRANCH IF NO
8412								
8413	051030	112737	177775	002132	1\$:	MOVB	#-3,@RP3TRY	:RESTORE TRY COUNT

```
8414 051036 032737 000040 172516 BIT #BITS,@MMR3 ;MAP ON?
8415 051044 001407 BEQ 2$ ;BRANCH IF NO
8416 051046 005046 CLR -(SP) ;PUT DEVICE ID ON STACK
8417 051050 004737 063142 JSR PC,@GIVEMAP ;RETURN MAP REGISTER
8418 051054 012746 002022 MOV @RP3NWL,-(SP) ;PUT ADR OF BUS ADR ON STK
8419 051060 004737 062676 JSR PC,@GETMAP ;GET MAP REGISTERS
8420 051064 010046 2$: MOV R0,-(SP) ;SAVE R0
8421 051066 013700 002024 MOV @RP3NWH,R0 ;GET BAE BITS
8422 051072 072027 000004 ASH #4,R0 ;ADJUST
8423 051076 010037 002024 MOV R0,@RP3NWH ;SAVE
8424 051102 012600 MOV (SP)+,R0 ;RESTORE R0
8425 051104 012737 000105 001612 MOV #105,@RP310 ;SET FUNCTION
8426 051112 053737 002024 001612 BIS @RP3NWH,@RP310 ;SET BAE BITS
8427 051120 053737 002052 001612 BIS @RP3UNIT,@RP310 ;SET UNIT NUMBER
8428 051126 013777 002102 131016 RP33: MOV @RP3HDA,@RP3DA ;LOAD DSK ADR
8429 051134 013777 002104 131012 MOV @RP3HDC,@RP3DC ;LOAD CYL
8430 051142 013777 001756 130776 MOV @RP3HWC,@RP3WC ;LOAD WORD COUNT
8431 051150 013777 002022 130772 MOV @RP3NWL,@RP3BA ;LOAD BUS ADR
8432 051156 013777 001612 130760 MOV @RP310,@RP3CS ;LOAD FUNCTION AND GO
8433 051164 000002 RTI ;RETURN
8434
8435 ;FUNCTION JUST EXECUTED WAS A READ
8436 051166 032737 000400 001736 RP3READ:BIT #BIT8,@RP3HSTA ;REPEAT FLAG SET?
8437 051174 001221 BNE RP3LOOP ;BRANCH IF YES
8438 051176 005777 130742 TST @RP3CS ;ANY ERRORS?
8439 051202 100022 BPL 1$ ;BRANCH IF NO
8440 051204 105737 002132 TSTB @RP3TRY ;TRIED 3 TIMES?
8441 051210 001600 BEQ RP3ERR ;BRANCH IF YES
8442 051212 005337 002120 DEC @RP3FUN ;RESTORE FUNCTION
8443 051216 112777 000001 130720 MOVB #BIT0,@RP3CS ;CLEAR THE DRIVE
8444 051224 105777 130714 TSTB @RP3CS ;CONTROLLER READY?
8445 051230 100375 BPL -4 ;BRANCH IF NO
8446 051232 105237 002132 INCB @RP3TRY ;INCREMENT TRY COUNT
8447 051236 013746 177776 MOV @RP3SW,-(SP)
8448 051242 012746 051126 MOV @RP33,-(SP)
8449 051246 000002 RTI ;GO TRY AGAIN
8450 051250 032737 000040 172516 1$: BIT #BITS,@MMR3 ;MAP ON?
8451 051256 001404 BFC 2$ ;BRANCH IF NO
8452 051260 005046 CLR -(SP) ;PUT DEVICE ID IN STK
8453 051262 004737 063142 JSR PC,@GIVEMAP ;RETURN MAP REGISTERS
8454 051266 005726 TST (SP)+ ;RESTORE STACK
8455 051270 112737 000200 001736 2$: MOVB #200,@RP3HSTA ;SET DONE FLAG
8456 051276 000002 RTI ;RETURN
8457
8458 ;*****
8459 ;SBTTL RK11/RK05 SERVICE ROUTINE
8460 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8461 ;*****
8461 051300 000005 RKRPT: RESET
8462 051302 005337 002122 DEC @RKFUN ;RESTORE FUNCTION
8463 051306 022737 000001 002122 CMP #1,@RKFUN ;WHAT IS IT?
8464 051314 001475 BEQ RK1 ;BRANCH IF WC
8465 051316 002402 BLT 1$ ;BRANCH IF WRITE
8466 051320 000137 047542 JMP @RKWTRY ;IT WAS A WRITE
8467 051324 000137 051762 1$: JMP @RK3
8468 051330 062737 000001 002122 RKSrv: ADD #1,@RKFUN ;FIND OUT WHAT FUNCTION
8469 ; WAS EXECUTED
```

```
8470 051336 022737 000002 002122      CMP      #2,@RKFUN      ;WAS IT A WRITE CHECK?
8471 051344 001507                      BEQ      RKWRCK      ;BRANCH IF YES
8472 051346 100002                      BPL      .+6         ;BRANCH IF IT WAS A WRITE
8473 051350 000137 052014      JMP      @RKREAD
8474
3475 ;FUNCTION JUST EXECUTED WAS A WRITE. ANY ERRORS?
8476 051354 032737 000400 001740      BIT      #BIT8,@RKHSTAT ;REPEAT FLAG SET?
8477 051362 001040                      BNF      RKLOOP      ;BRANCH IF YES
8478 051364 005777 130576      TST      @RKCS      ;ANY ERRORS?
8479 051370 100047                      BPL      RK1         ;BRANCH IF NO
8480 051372 105737 002133      TSTB    @RKTRY      ;TRYED 3 TIMES?
8481 051376 001417                      BEQ      RKERR      ;BRANCH IF YES
8482 051400 012777 000001 130560      MOV      #1,@RKCS    ;CLEAR THE ERPOP
8483 051406 004737 052140      JSR      PC,@TIMER   ;WAIT A LITTLE
8484 051412 105777 130550      TSTB    @RKCS      ;WAIT FOR CONT CLR TO FINISH
8485 051416 100375                      BPL      .-4
8486 051420 105237 002133      INCB    @RKTRY      ;INCREMENT TRY COUNT
8487 051424 013746 177776      MOV      @PSW,-(SP)
8488 051430 012746 047542      MOV      #RKWTRY,-(SP)
8489 051434 000002                      RTI
8490 051436 012737 100200 001740      RKERR:  MOV      #100200,@RKHSTAT ;SET ERROR & DONE FLAG
8491 051444 010046                      MOV      R0,-(SP)    ;SAVE R0
8492 051446 013700 001620      MOV      @RK11,R0    ;GET SAVED RUN TABLE INDEX
8493 051452 052760 100000 001700      BIS      #BIT15,RUNTRAK(R0) ;SET ERROR BIT IN RUN TABLE
8494 051460 012600                      MOV      (SP)+,R0    ;RESTORE R0
8495 051462 000002                      RTI                  ;RETURN
8496
8497 051464 012737 100200 001740      RKLOOP:MOV      #100200,@RKHSTAT ;SET DONE AND ERROR BITS
8498 051472 005777 130470      TST      @RKCS      ;ANY ERRORS?
8499 051476 100403                      BMI      1$         ;BRANCH IF YES
8500 051500 042737 100000 001740      BIC      #BIT15,@RKHSTAT ;CLEAR ERROR BIT
8501 051506 000000                      RTI                  ;RETURN
8502 ;WRITE WAS OK, NOW DO A WRITE CHECK
8503 051510 112737 177775 002133      RK1:    MOV      #-3,@RKTRY    ;RESTORE TRY COUNT
8504 051516 012767 000507 130072      MOV      #507,RK10    ;SET FUNCTION TO WRITE
8505 051524 053767 002000 130064      BIS      @RKOLD+2,RK10 ;SET BA EXT BITS
8506 051532 013777 002106 130434      RK2:    MOV      @RKHDA,@RKDA    ;LOAD DISK ADDRESS
8507 051540 013777 001760 130422      MOV      @RKHWC,@RKWC    ;LOAD WORD COUNT
8508 051546 013777 001776 130416      MOV      @RKOLD,@RKBA    ;LOAD BUS ADDRESS
8509 051554 016777 130036 130404      MOV      RK10,@RKCS    ;START FUNCTION
8510 051562 000002                      RTI                  ;RETURN
8511
8512 ;FUNCTION JUST EXECUTED WAS A WRITE CHECK. ANY ERRORS?
8513 051564 032737 000400 001740      RKWRCK:BIT      #BIT8,@RKHSTAT ;REPEAT FLAG SET?
8514 051572 001334                      BNE      RKLOOP      ;BRANCH IF YES
8515 051574 005777 130366      TST      @RKCS      ;ANY ERRORS?
8516 051600 100033                      BPL      1$         ;BRANCH IF NO
8517 051602 005737 001620      TST      @RK11      ;FIRST 2K?
8518 051606 001424                      BEQ      4$         ;BRANCH IF YES
8519 051610 105737 002133      5$:    TSTB    @RKTRY      ;TRYED 3 TIMES?
8520 051614 001710                      BEQ      RKERR      ;BRANCH IF YES
8521 051616 005337 002126 130336      DEC      @RP4FUN      ;SET FUNCTION BACK TO WC
8522 051622 012777 000001                      MOV      #1,@RKCS    ;CLEAR THE ERROR
8523 051630 004737 052140      JSR      PC,@TIMER   ;WAIT A LITTLE
8524 051634 105777 130326      TSTB    @RKCS      ;WAIT FOR CLR TO FINISH
8525 051640 100375                      BPL      .-4
```

```

8526 051642 105237 002133          INCB  @RKTRY          :INCREMENT TRY COUNT
8527 051646 013746 177776          MOV   @PSW,-(SP)
8528 051652 012746 051532          MOV   @RK2,-(SP)
8529 051656 000002                    RTI
8530 051660 032777 040000 130300 4$: BIT   @BIT14,@RKCS    :HARD ERROR?
8531 051666 001350                    BNE   5$             :BRANCH IF YES
8532
8533          :WRITE CHECK WAS OK, NOW DO A READ.
8534 051670 112737 177775 002133 1$: MOVB  #-3,@RKTRY    :RESTORE TRY COUNT
8535 051676 032737 000040 172516    BIT   @BIT5,@MMR3    :MAP ON?
8536 051704 001410                    BEQ   2$             :BRANCH IF NO
8537 051706 012746 000001          MOV   #1,-(SP)      :PUT DEVICE ID ON STACK
8538 051712 004767 011224          JSR   PC,GIVEMAP    :RELINQUISH MAP REG
8539 051716 012746 002026          MOV   @RKNEWL,-(SP) :PUT ADR OF BADR ON STACK
8540 051722 004737 062676          JSR   PC,@GETMAP    :GET MAPREGISTER
8541 051726 010046                    2$: MOV   R0,-(SP)    :SAVE R0
8542 051730 013700 002030          MOV   @RKNEWH,R0    :GET BA EXT
8543 051734 072027 000004          ASH   #4,R0         :ADJUST
8544 051740 010037 002030          MOV   R0,@RKNEWH    :SAVE
8545 051744 012600                    MOV   (SP)+,R0      :RESTORE R0
8546 051746 012767 000105 127642    MOV   #105,RK10     :SET FUNCTION
8547 051754 053767 002030 127634    BIS   @RKNEWH,RK10  :SET BA EXT BITS IN FUNCTION
8548 051762 013777 002106 130204    RK3: MOV   @RKHDA,@RKDA :LOAD DISK ADDRESS
8549 051770 013777 001760 130172    MOV   @RKHWC,@RKWC  :LOAD WORD COUNT
8550 051776 013777 002026 130166    MOV   @RKNEWL,@RKBA :LOAD BUS ADDRESS
8551 052004 016777 127606 130154    MOV   RK10,@RKCS    :LOAD FUNCTION AND GO
8552 052012 000002                    RTI                :RETURN
8553
8554          :FUNCTION JUST EXECUTED WAS A READ. ANY ERRORS?
8555 052014 032737 000400 001740 RKREAD: BIT   @BIT8,@RKHSTAT :REPEAT FLAG SET?
8556 052022 001220                    BNE   RKLOOP        :BRANCH IF YES
8557 052024 005777 130136          TST   @RKCS         :ANY ERRORS?
8558 052030 100026                    BPL   1$             :BRANCH IF NO
8559 052032 105737 002133          TSTB  @RKTRY        :TRIED 3 *TIMES?
8560 052036 001002                    BNE   3$             :BRANCH IF NO
8561 052040 000167 177372          JMP   RKERR
8562 052044 005337 002122          3$: DEC   @RKFUN      :SET FUNCTION BACK TO READ
8563 052050 012777 000001 130110    MOV   #1,@RKCS     :CLEAR THE ERROR
8564 052056 004737 052140          JSR   PC,@TIMER    :WAIT A LITTLE
8565 052062 105777 130100          TSTB  @RKCS         :WAIT FOR CLR TO FINISH
8566 052066 100375                    BPL   -4
8567 052070 105237 002133          INCB  @RKTRY        :INCREMENT TRY COUNT
8568 052074 013746 177776          MOV   @PSW,-(SP)
8569 052100 012746 051762          MOV   @RK3,-(SP)
8570 052104 000002                    RTI
8571 052106 032737 000040 172516 1$: BIT   @BIT5,@MMR3    :MAP ON?
8572 052114 001405                    BEQ   2$             :BRANCH IF NO
8573 052116 012746 000001          MOV   #1,-(SP)      :PUT RK ID ON STACK
8574 052122 004737 063142          JSR   PC,@GIVEMAP   :RELINQUISH MAP REGISTER
8575 052126 005726                    TST   (SP)+         :POP THE STACK
8576 052130 112737 000200 001740 1$: MOVB  #200,@RKHSTAT :SET DON E FLAG
8577 052136 000002                    RTI                :RETURN
8578 052140 005067 000010          TIMER: CLR   1$
8579 052144 105267 000004          2$: INCB  1$
8580 052150 001375                    BNE   2$
8581 052152 000207                    RTS   PC

```

```
8582 052154 000000 1$: .WORD
8583
8584
8585
8586
8587
8588 052156 000005
8589 052160 005337 002126
8590 052164 022737 000001 002126
8591 052172 001501
8592 052174 002560
8593 052176 000137 050064
8594 052202 005237 002126
8595 052206 022737 000002 002126
8596 052214 001504
8597 052216 100566
8598
8599
8600 052220 032737 000400 001746
8601 052226 001050
8602 052230 032777 040000 127762
8603 052236 001457
8604 052240 105737 002135
8605 052244 001426
8606 052246 052777 000040 127740
8607 052254 004737 050116
8608 052260 105237 002135
8609 052264 013746 177776
8610 052270 012746 050064
8611 052274 032737 000400 001746
8612 052302 001006
8613 052304 012777 000007 127670
8614 052312 105777 127702
8615 052316 100375
8616 052320 000002
8617 052322 012737 100200 001746
8618 052330 010046
8619 052332 013700 001622
8620 052336 052760 100000 001700
8621 052344 012600
8622 052346 000002
8623
8624 052350 012737 100200 001746
8625 052356 032777 040000 127634
8626 052364 001003
8627 052366 042737 100000 001746
8628 052374 000002
8629
8630 052376 112737 177775 002135
8631 052404 105777 127610
8632 052410 001775
8633 052412 004737 050116
8634 052416 112777 000151 127556
8635 052424 000002
8636
8637
```

```
*****
:SBTTL RH70/RP04 SERVICE ROUTINE
:* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
*****
RP4RPT: RESET
DEC @RP4FUN ;RESTORE FUNCTION
CMP #1,@RP4FUN ;WHAT IS IT?
BEQ RP41 ;BRANCH IF WC
BLT RP43 ;BRANCH IF READ
JMP @RP4WTRY ;GO TO WRITE
RP4SRV: INC @RP4FUN ;FIND OUT WHAT FUNCTION
CMP #2,@RP4FUN ;WAS JUST EXECUTED
BEQ RP4WCK
BML RP4READ

;WRITE FUNCTION WAS JUST EXECUTED.
BIT #BIT8,@RP4HSTAT ;REPEAT FLAG SET?
BNE RP4LOOP ;BRANCH IF YES
BIT #BIT14,@RP4DS ;ANY ERRORS
BEQ RP41 ;BRANCH IF NO
TSTB @RP4TRY ;TRIED 3 TIMES?
BEQ RP4ERR ;BRANCH IF YES
BIS #BIT5,@RP4CS2 ;CLEAR ALL ERRORS
JSR PC,@LDRP4 ;RELOAD THE UNIT NO
INCB @RP4TRY ;INCREMENT TRY COUNT
MOV @PSW,-(SP) ;SETUP THE STACK TO
MOV @RP4WTRY,-(SP) ;TRY WRITE AGAIN
BIT #BIT8,@RP4HSTAT ;REPEAT FLAG SET?
BNE 2$ ;BRANCH IF YES
MOV #7,@RP4CS1 ;RECALIBRATE
1$: TSTB @RP4DS ;DRIVE READY?
BPL 1$ ;BRANCH IF NO
2$: RTI
RP4ERR: MOV #100200,@RP4HSTA ;SET ERROR & DONE BIT
MOV R0,-(SP) ;SAVE R0
MOV @RP4I1,R0 ;GET RUN TABLE INDEX
BIS #BIT15,RUNTRAK(R0) ;SET ERROR BIT
MOV (SP)+,R0 ;RESTORE R0
RTI ;RETURN

RP4LOOP:MOV #100200,@RP4HSTAT ;SET DONE AND ERROR BITS
BIT #BIT14,@RP4DS ;ANY ERRORS?
BNE 1$ ;BRANCH IF YES
BIC #BIT15,@RP4HSTAT ;CLEAR ERROR BIT
1$: RTI ;RETURN

;WRITE OK...NOW DO A WRITE CHECK.
RP41: MOVB #-3,@RP4TRY ;INITIALIZE TRY COUNT
RP42: TSTB @RP4DS ;IS DRIVE READY?
BEQ RP42 ;BRANCH IF NO
JSR PC,@LDRP4
MOVB #151,@RP4CS1 ;LOAD FUNCTION AND GO
RTI

;FUNCTION JUST EXECUTED WAS A WRITE CHECK
```



```

8638 052426 032737 000400 001746 RP4WCK: BIT #BIT8,@RP4HSTAT ;REPEAT FLAG SET?
8639 052434 001345 BNE RP4LOOP ;BRANCH IF YES
8640 052436 032777 040000 127554 BIT #BIT14,@RP4DS ;ANY ERRORS?
8641 052444 001421 BEQ 1$ ;BRANCH IF NO
8642 052446 105737 002135 3$: TSTB @RP4TRY ;TRIED 3 TIMES?
8643 052452 001723 BEQ RP4ERR ;BRANCH IF YES
8644 052454 005337 002126 DEC @RP4FUN ;SET TO WRITE CHECK
8645 052460 052777 000040 127526 BIS #BIT5,@RP4CS2 ;CLEAR ALL ERRORS
8646 052466 004737 050116 JSR PC,@#LDRP4 ;RELOAD THE UNIT NO
8647 052472 105237 002135 INCB @RP4TRY ;INCREMENT TRY COUNT
8648 052476 013746 177776 MOV @PSW,-(SP)
8649 052502 012746 052404 MOV #RP42,-(SP)
8650 052506 000002 RTI ;TRY AGAIN
8651 052510 032777 040000 127476 1$: BIT #BIT14,@RP4CS2 ;WRITE CHECK ERROR?
8652 052516 001404 BEQ 2$ ;BRANCH IF NO
8653 052520 005737 001622 TST @RP411 ;FIRST 2K?
8654 052524 001401 BEQ 2$
8655 052526 000747 BR 3$
8656
8657 ;WRITE CHECK WAS OK...NOW DO A READ.
8658 052530 112737 177775 002135 2$: MOVB #-3,@RP4TRY ;INITIALIZE TRY COUNT
8659 052536 105777 127456 RP43: TSTB @RP4DS ;IS DRIVE READY?
8660 052542 001775 BEQ RP43 ;BRANCH IF NO
8661 052544 004737 050116 JSR PC,@#LDRP4 ;LOAD REGISTERS
8662 052550 013777 002044 127432 MOV @RP4NWL,@RP4BAE ;LOAD EXTENDED ADR BITS
8663 052556 013777 002042 127422 MOV @RP4NWL,@RP4BA ;LOAD BUS ADR
8664 052564 112777 000171 127410 MOVB #171,@RP4CS1 ;LOAD FUNCTION AND GC
8665 052572 000002 RTI ;RETURN
8666
8667 ;FUNCTION JUST EXECUTED WAS A READ.
8668 052574 032737 000400 001746 RP4READ:BIT #BIT8,@RP4HSTAT ;REPEAT FLAG SET?
8669 052602 001262 BNE RP4LOOP ;BRANCH IF YES
8670 052604 032777 040000 127406 BIT #BIT14,@RP4DS ;ANY ERRORS?
8671 052612 001421 BEQ 1$ ;BRANCH IF NO
8672 052614 105737 002135 TSTB @RP4TRY ;TRIED 3 TIMES?
8673 052620 001640 BEQ RP4ERR ;BRANCH IF YES
8674 052622 005337 002126 DEC @RP4FUN ;SET FUNCTION TO A READ
8675 052626 052777 000040 127360 BIS #BIT5,@RP4CS2 ;CLEAR ALL ERRORS
8676 052634 004737 050116 JSR PC,@#LDRP4 ;RELOAD THE UNIT NO
8677 052640 105237 002135 INCB @RP4TRY ;INCREMENT TRY COUNT
8678 052644 013746 177776 MOV @PSW,-(SP)
8679 052650 012746 052536 MOV #RP43,-(SP)
8680 052654 000002 RTI ;TRY AGAIN
8681 052656 112737 000200 001746 1$: MOVB #200,@RP4HSTA ;SET DONE FLAG
8682 052664 000002 RTI ;RETURN
8683
8684 ;*****
8685 ;SBITL RH70/RS04 SERVICE ROUTINE
8686 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8687 ;*****
8687 052666 000005 RSRPT: RESET
8688 052670 005337 002130 DEC @RSFUN ;RESTORE FUNCTION
8689 052674 022737 000001 002130 CMP #1,@RSFUN ;WHAT IS IT?
8690 052702 001467 BEQ RS41 ;BRANCH IF WC
8691 052704 002546 BLT RS43 ;BRANCH IF WRITE
8692 052706 000137 050362 JMP @RSWTRY
8693 052712 005237 002130 RSSRV: INC @RSFUN ;FIND OUT WHAT FUNCTION
```

```

8694 052716 022737 000002 002130      CMP      #2,@R$RFUN      ;WAS JUST EXECUTED
8695 052724 001472                      BEQ      RSWCK
8696 052726 100554                      BMI      RSREAD
8697
8698
8699 052730 032737 000400 001750      ;WRITE FUNCTION WAS JUST EXECUTED
8700 052736 001036                      BIT      #BIT8,@R$SHSTAT ;REPEAT FLAG SET?
8701 052740 032777 040000 127312      BNE      RSLOOP      ;BRANCH IF YES
8702 052746 001445                      BIT      #BIT14,@R$SDS ;ANY ERRORS?
8703 052750 105737 002136                      BEQ      RS41      ;BRANCH IF NO
8704 052754 001414                      TSTB    @R$STRY      ;TRIED 3 TIMES?
8705 052756 052777 000040 127270      BEQ      RSERR      ;BRANCH IF YES
8706 052764 004737 050422                      BIS      #BIT5,@R$SCS2 ;CLEAR ALL ERRORS
8707 052770 105237 002136                      JSR      PC,@R$LDRS  ;LOAD UNIT #
8708 052774 013746 177776                      INCB    @R$STRY      ;INCREMENT TRY COUNT
8709 053000 012746 050362                      MOV     @R$PSW,-(SP) ;SETUP THE STACK TO
8710 053004 000002                      MOV     @R$WTRY,-(SP) ;TRY THE WRITE AGAIN
8711 053006 012737 100200 001750      RSERR:  MOV     #100200,@R$SHSTAT ;SET ERROR AND DONE BIT
8712 053014 010046                      MOV     R0,-(SP)      ;SAVE R0
8713 053016 013700 001624                      MOV     @R$R11,R0    ;GET RUN TBL INDEX
8714 053022 052760 100000 001700      BIS     #BIT15,RUNTRAK(R0) ;SET ERROR BIT
8715 053030 012600                      MOV     (SP)+,R0     ;RESTORE R0
8716 053032 000002                      RTI
8717
8718 053034 012737 100200 001750      RSLOOP: MOV     #100200,@R$SHSTAT ;SET DONE AND ERROR BITS
8719 053042 032777 040000 127210      BIT     #BIT14,@R$SDS ;ANY ERRORS?
8720 053050 001003                      BNE     1$           ;BRANCH IF YES
8721 053052 042737 100000 001750      BIC     #BIT15,@R$SHSTAT ;CLEAR ERROR BIT
8722 053060 000002                      RTI           ;RETURN
8723
8724 053062 112737 177775 002136      ;WRITE OK...NOW DO A WRITE CHECK
8725 053070 105777 127164      RS41:  MOV     #-3,@R$STRY ;INIT TRY COUNT
8726 053074 001775                      RS42:  TSTB    @R$SDS  ;IS DRIVE READY?
8727 053076 004737 050422                      BEQ     RS42      ;BRANCH IF NO
8728 053102 112777 000151 127132      JSR     PC,@R$LDRS  ;LOAD RS REGISTERS
8729 053110 000002                      MOV     #151,@R$SCS1 ;LOAD FUNCTION AND GO
8730
8731
8732 053112 032737 000400 001750      ;FUNCTION JUST EXECUTED WAS A WRITE CHECK
8733 053120 001345      RSWCK:  BIT     #BIT8,@R$SHSTAT ;REPEAT FLAG SET?
8734 053122 032777 040000 127130      BNE     RSLOOP      ;BRANCH IF YES
8735 053130 001421                      BIT     #BIT14,@R$SDS ;ANY ERRORS?
8736 053132 105737 002136                      BEQ     1$           ;BRANCH IF NO
8737 053136 001723                      TSTB    @R$STRY      ;TRIED 3 TIMES?
8738 053140 005337 002130                      BEQ     RSERR      ;BRANCH IF YES
8739 053144 052777 000040 127102      DEC     @R$RFUN      ;SET FUNCTION BACK TO WC
8740 053152 004737 050422                      BIS     #BIT5,@R$SCS2 ;CLEAR THE ERROR
8741 053156 105237 002136                      JSR     PC,@R$LDRS
8742 053162 013746 177776                      INCB    @R$STRY      ;INCREMENT THE TRY COUNT
8743 053166 012746 053070                      MOV     @R$PSW,-(SP)
8744 053172 000002                      MOV     @R$R42,-(SP)
8745
8746 053174 032777 040000 127052      1$:    RTI           ;TRY AGAIN
8747 053202 001404                      BIT     #BIT14,@R$SCS2 ;WRITE CHECK ERROR?
8748 053204 005737 001624                      BEQ     2$           ;BRANCH IF NO
8749 053210 001401                      TST     @R$R11      ;FIRST 2K?
8749 053210 001401                      BEQ     2$           ;BRANCH IF YES
    
```

```

8750 053212 000747          BR      3$
8751
8752          ;WRITE CHECK WAS OK...NOW DO A READ.
8753 053214 112737 177775 002136 2$:  MOVB  #-3,@RSTRY          ;INIT TRY COUNT
8754 053222 105777 127032 RS43: TSTB  @RSDS          ;IS DRIVE READY?
8755 053226 001775          BEQ    RS43          ;BRANCH IF NO
8756 053230 004737 050422          JSR    PC,@#LDRS          ;LOAD RS REGISTERS
8757 053234 013777 002050 127006          MOV    @#RSNEWH,@RSBAE          ;LOAD BAE
8758 053242 013777 002046 126776          MOV    @#RSNEWL,@RSBA          ;LOAD BUS ADR
8759 053250 112777 000171 126764          MOVB  #171,@RSCS1          ;LOAD FUNCTION AND GO
8760 053256 000002          RTI          ;RETURN
8761
8762          ;FUNCTION JUST EXECUTED WAS A READ.
8763 053260 032737 000400 001750 RSREAD: BIT  #BIT8,@#RSHSTAT          ;REPEAT FLAG SET?
8764 053266 001262          BNE   RSLOOP          ;BRANCH IF YES
8765 053270 032777 040000 126762          BIT  #BIT14,@RSDS          ;ANY ERRORS?
8766 053276 001421          BEQ    1$          ;BRANCH IF NO
8767 053300 105737 002136          TSTB  @RSTRY          ;TRIED 3 TIMES?
8768 053304 001640          BEQ    RSERR          ;BRANCH IF YES
8769 053306 005337 002130          DEC   @#RSFUN          ;RESTORE FUN TO READ
8770 053312 052777 000040 126734          BIS  #BITS,@RSCS2          ;CLEAR ALL ERRORS
8771 053320 004737 050422          JSR    PC,@#LDRS          ;LOAD UNIT #
8772 053324 105237 002136          INCB  @RSTRY          ;INCREMENT TRY COUNT
8773 053330 013746 177776          MOV    @#PSW,-(SP)
8774 053334 012746 053222          MOV    #RS43,-(SP)
8775 053340 000002          RTI          ;TRY AGAIN
8776 053342 112737 000200 001750 1$:  MOVB  #200,@#RSHSTAT          ;SET DONE FLAG
8777 053350 000002          RTI          ;RETURN
8778
8779          ;*****
8800          ;SBTTL UNIBUS EXERCISER SERVICE ROUTINE
8801          ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8802          ;*****
8803          ;UBESRV: SAVREG
8804          JSR    PC,@#LDKT          ;GO TO LOW CORE
8805          MOV    #UBETBL+6,R4          ;GET ADDRESS OF UBECR1
8806          TST   @R4          ;WAS THERE AN ERROR?
8807          BMI   UBE2          ;BRANCH IF YES
8808          MOV    #3,-(SP)          ;PUT DEVICE ID IN STACK
8809          JSR    PC,@#GIVEMAP          ;GIVE UP MAP REG
8810          MOV    #SERRTB,UBESAV          ;INITIALIZE UBE
8811          CLR   UBESAV+2
8812          MOV    #SERRTB,UBEADR
8813          CLR   UBEADR+2
8814          MOV    #UBEADR,-(SP)
8815          JSR    PC,@#GETMAP
8816          MOV    @#UBEADR+2,@-(R4)          ;LOAD UBECR2
8817          MOV    @#UBEADR,@-(R4)          ;LOAD UBEBA
8818          MOV    #172400,@-(R4)          ;LOAD UBECC
8819          JSR    PC,@#RESKT          ;GO BACK TO ORIGINAL CORE
8820          RESREG
8821          MOV    #64545,@UBETBL+6          ;RESTART UBE
8822          RTI          ;RETURN
8823
8824          ;UBE ERROR-IS IT LAST MEMORY?
8825          UBE2: CLR   @#STMP10
8826          SUB   #4,R4          ;ADJUST R4

```

```
8806 053500 017403 000002      MOV      @2(R4),R3      ;GET BECR2
8807 053504 042703 020003      BIC      #20003,R3     ;GET RID OF ADDRESS BITS
8808 053510 022703 000400      CMP      #400,R3      ;WAS ERROR A TIMEOUT?
8809 053514 001052              BNE      UBEERR       ;BRANCH IF NO
8810 053516 017437 000000 001732  MOV      @2(R4),@#ERRBA ;SAVE BUS ADR OF ERROR
8811 053524 017437 000002 001734  MOV      @2(R4),@#ERRBA+2
8812 053532 042737 177774 001734  BIC      #177774,@#ERRBA+2
8813 053540 004737 062324              JSR      PC,@#PHYMAP   ;GET PHYSICAL ADDRESS THAT TIMED OUT
8814 053544 162737 000004 001524  SUB      #4,@#PA1500   ;ADJUST PHYSICAL ADR THAT FAILED
8815 053552 005637 001526              SBC      @#PA2116     ;UBE STOPS AT ADR+4
8816 053556 023737 001526 001606  CMP      @#PA2116,@#MXMMHI ;AT MAX MEM HIGH?
8817 053564 101006              BHI      1$           ;BRANCH IF HIGHER
8818 053566 103423              BLO      MHOLE        ;BRANCH IF LOWER
8819 053570 023737 001524 001610  CMP      @#PA1500,@#MXMMLO ;AT MAX MEM LO?
8820 053576 101001              BHI      1$           ;BRANCH IF HIGHER
8821 053600 103416              BLO      MHOLE        ;BRANCH IF LOWER
8822 053602 012746 000002 1$:      MOV      #3,-(SP)      ;PUT DEVICE ID ON STACK
8823 053606 004737 063142              JSR      PC,@#GIVEMAP
8824 053612 005726              TST      (SP)+
8825 053614 004737 062214              JSR      PC,@#UBEINIT
8826 053620 004737 064050              JSR      PC,@#RESKT
8827 053624 104422              RESREG
8828 053626 012777 064545 126442  MOV      #64545,@UBETBL+6
8829 053634 000002              RTI
8830
8831 053636 010637 001322  MHOLE:  MOV      SP,@#STMP10
8832 053642 013737 001212 001324  UBEERR: MOV      @#SLPERR,@#STMP11 ;SAVE LOOP ERROR ADR
8833 053650 012737 053712 001212  MOV      #UBE3,@#SLPERR ;SET LOOP ADR
8834 053656 012703 000022              MOV      #22,R3
8835 053662 005737 001322              TST      @#STMP10
8836 053666 001002              BNE      1$
8837 053670 104007              ERROR   7
8838 053672 000407              BR      UBE3
8839 053674 013737 001524 001226 1$:      MOV      @#PA1500,@#SGDDAT
8840 053702 013737 001526 001230  MOV      @#PA2116,@#SBDDAT
8841 053710 104012              ERROR   12
8842
8843 ;RESTART UBE IN SAME MEMORY
8844 053712 013737 001324 001212  UBE3:  MOV      @#STMP11,@#SLPERR ;RESTORE ERROR LOOP ADR
8845 053720 010446              MOV      R4,-(SP)      ;SAVE R4
8846 053722 012704 002270              MOV      #UBETBL,R4   ;GET ADDRESS OF UBE TABLE
8847 053726 012734 172400              MOV      #172400,@(R4)+ ;SET UBECC
8848 053732 013734 001726              MOV      @#UBEADR,@(R4)+ ;SET UBEBA <15:00>
8849 053736 005074 000004              CLR      @4(R4)       ;CLEAR ALL ERRORS
8850 053742 013734 001730              MOV      @#UBEADR+2,@(R4)+ ;SET EXT ADR BITS
8851 053746 012774 064545 000000  MOV      #64545,@(R4) ;START UBE
8852 053754 012604              MOV      (SP)+,R4     ;RESTORE R4
8853 053756 004737 064050              JSR      PC,@#RESKT
8854 053762 104422              RESREG
8855 053764 000002              RTI ;RETURN
8856
8857 ;*****
8858 ;SBTTL MASS BUS TESTER SERVICE ROUTINE
8859 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8860 ;*****
8861 053766 104420  MBTSRV: SAVREG
```

8862	053770	004737	063752		JSR	PC,@WLDKT	:GO TO LOW CORE	
8863	053774	005037	001322		CLR	@WSTMP10		
8864	054000	012704	002306		MOV	#MBTTBL,R4	:GET ADDRESS OF ADDRESS OF CS1 REG	
8865	054004	032734	040000		BIT	#BIT14,@(R4)+	:ANY ERRORS?	
8866	054010	001007			BNE	1\$:BRANCH IF YES	
8867	054012	004737	064050	2\$:	JSR	PC,@WRESKT	:GO BACK TO ORIGINAL CORE	
8868	054016	104422			RESREG			
8869	054020	112777	000161	126260	MOVB	#161,@MBTTBL		
8870	054026	000002			RTI		:RESTART MBT AND RETURN	
8871	054030	062704	000010		ADD	#10,R4	:ADJUST R4	
8872	054034	032774	004000	000000	BIT	#BIT11,@(R4)	:NON-EXISTANT MEMORY ERROR?	
8873	054042	001436			BEQ	MBTERR	:BRANCH IF NO	
8874	054044	162704	000006		SUB	#6,R4	:ADJUST R4	
8875	054050	013437	001524		MOV	@(R4)+,@WPA1500	:GET BUS ADR	
8876	054054	013437	001526		MOV	@(R4)+,@WPA2116	:GET BUS ADR EXT	
8877	054060	162737	000004	001524	SUB	#4,@WPA1500	:ADJUST BUS ADR	
8878	054066	005637	001526		SBC	@WPA2116		
8879	054072	023737	001524	001610	CMP	@WPA1500,@WXMML0	:IS IT LAST MEMORY?	
8880	054100	001015			BNE	MEMHOLE	:BRANCH IF NO	
8881	054102	023737	001526	001606	CMP	@WPA2116,@WXMMLHI	:CHECK EXT ADR BITS	
8882	054110	001011			BNE	MEMHOLE		
8883	054112	005724			TST	(R4)+	:INCREMENT R4	
8884	054114	052774	000047	000000	BIS	#47,@(R4)	:CLEAR THE ERROR	
8885	054122	012734	000007		MOV	#7,@(R4)+	:SELECT UNIT 7	
8886	054126	005074	177766		CLR	@-12(R4)	:CLEAR WORD COUNT	
8887	054132	000727			BR	2\$:CONTINUE	
8888								
8889	054134	010637	001322		MEMHOLE:MOV	SP,@WSTMP10		
8890	054140	013737	001212	001324	MBTERR:MOV	@W\$LPERR,@WSTMP11	:SAVE LOOP ADDRESS	
8891	054146	012737	054210	001212	MOV	#1\$,@W\$LPERR	:SET NEW LOOP ADR	
8892	054154	012703	000020		MOV	#20,R3	:PUT DEVICE ID IN R3	
8893	054160	005737	001322		TST	@WSTMP10		
8894	054164	001002			BNE	2\$		
8895	054166	104007			ERROR	7		
8896	054170	000407			BR	1\$		
8897	054172	013737	001524	001226	2\$:	MOV	@WPA1500,@W\$GDDAT	
8898	054200	013737	001526	001230	MOV	@WPA2116,@W\$BDDAT		
8899	054206	104013			ERROR	13		
8900	054210	013737	001324	001212	1\$:	MOV	@WSTMP11,@W\$LPERR	:RESTORE LOOP ADR
8901	054216	012704	002316		MOV	#MBTTBL+10,R4	:GET ADR OF MBTTBL+10	
8902	054222	015400			MOV	@-(R4),R0	:GET BUS ADR EXTENDED	
8903	054224	015401			MOV	@-(R4),R1	:GET BUS ADR	
8904	054226	015402			MOV	@-(R4),R2	:GET WORD COUNT	
8905	054230	006302			ASL	R2	:ADJUST WORD COUNT	
8906	054232	160201			SUB	R2,R1	:FORM START ADR OF THIS XFER	
8907	054234	005600			SBC	R0		
8908	054236	052774	000047	000010	BIS	#47,@10(R4)	:CLEAR THE WORLD	
8909	054244	012774	000007	000010	MOV	#7,@10(R4)	:SELECT UNIT 7	
8910	054252	005724			TST	(R4)+	:ADJUST R4	
8911	054254	010134			MOV	R1,@(R4)+	:RESTORE BUS ADR	
8912	054256	010074	000000		MOV	R0,@(R4)		
8913	054262	004737	064050		JSR	PC,@WRESKT	:GO BACK TO ORIGINAL CORE	
8914	054266	104422			RESREG			
8915	054270	112777	000161	126010	MOVB	#161,@MBTTBL	:START MBT AGAIN	
8916	054276	000002			RTI		:RETURN	
8917								

.....

8918
8919
8920
8921
8922
8923
8924 054300 104420
8925 054302 004737 063752
8926 054306 105237 001630
8927 054312 122737 000074 001630
8928 054320 001014
8929 054322 105237 001631
8930 054326 105037 001630
8931 054332 122737 000074 001631
8932 054340 001004
8933 054342 105037 001631
8934 054346 005237 001626
8935 054352 004737 064050
8936 054356 104422
8937 054360 012737 000100 177546
8938 054366 000002
8939
8940
8941
8942
8943
8944
8945
8946
8947
8948
8949
8950
8951
8952
8953 054370
8954 054370 032737 040000 177570
8955 054376 001077
8956
8957 054400 000416
8958
8959 054402 013746 000004
8960 054406 012737 054426 000004
8961 054414 005737 177060
8962 054420 012637 000004
8963 054424 000453
8964 054426 022626
8965 054430 012637 000004
8966 054434 000413
8967 054436
8968 054436 105767 124542
8969 054442 001421
8970 054444 126767 124547 124532
8971 054452 101015
8972 054454 032737 001000 177570
8973 054462 001404

```
.SBTTL LINE CLOCK SERVICE ROUTINE
;* THIS ROUTINE FIRST REMAPS PROGRAM EXECUTION TO LOW
;* MEMORY. IT THEN INCREMENTS AND KEEPS TRACK OF THE
;* SECOND AND MINUTE COUNTS KEPT IN LOCATIONS 'LTICKS'
;* AND 'MTICKS' RESPECTIVELY.
;*****
KSRV: SAVREG
      JSR    PC,@WLDKT          ;GO TO LOW CORE
      INCB  @WLTICKS          ;INCREMENT TICK COUNT
      CMPB  #60.,@WLTICKS     ;ONE SECOND YET?
      BNE   1$                ;BRANCH IF NO
      INCB  @WLTICKS+1        ;INCREMENT SECOND COUNT
      CLRB  @WLTICKS          ;CLEAR SECOND COUNT
      CMPB  #60.,@WLTICKS+1   ;ONE MINUTE YET?
      BNE   1$                ;BRANCH IF NO
      CLRB  @WLTICKS+1        ;INCREMENT MINUTE COUNT
      INC   @MTICKS           ;RESTORE THE KT
1$:   JSR    PC,@WRESKT
      RESREG
      MOV   #BIT6,@WCLKS      ;CLEAR READY BIT IN CLOCK
      RTI                      ;RETURN
;*****
```

```
.SBTTL SCOPE HANDLER ROUTINE
;* THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;* AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;* THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;* SW14=1 LOOP ON TEST
;* SW11=1 INHIBIT ITERATIONS
;* SW09=1 LOOP ON ERROR
;* CALL SCOPE ;;SCOPE-IOT
$SCOPE:
      BIT   #SW14,@SWR        ;;LOOP ON PRESENT TEST?
      BNE   $OVER             ;;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR    6$              ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                                      ;;THIS INSTRUCTION TO A 'NOP' (NOP-240)
      MOV   @WERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
      MOV   #5$,@WERRVEC     ;;SET FOR TIMEOUT
      *ST   @W177060         ;;TIME OUT ON XOR?
      MOV   (SP)+,@WERRVEC   ;;RESTORE THE ERROR VECTOR
      BR    $SVLAD           ;;GO TO THE NEXT TEST
5$:   CMP   (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
      MOV   (SP)+,@WERRVEC   ;;RESTORE THE ERROR VECTOR
      BR    7$              ;;LOOP ON THE PRESENT TEST
6$:   ;*****END OF CODE FOR THE XOR TESTER*****
2$:   TSTB  $ERFLG           ;;HAS AN ERROR OCCURRED?
      BEQ   3$              ;;BR IF NO
      CMPB  $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
      BHI   3$              ;;BR IF NO
      BIT   #BIT09,@SWR     ;;LOOP ON ERROR?
      BEQ   4$              ;;BR IF NO
```

```
8974 054464 016767 124522 124516 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
8975 054472 000441 BR $OVER
8976 054474 105067 124504 4$: CLR $ERFLG ;;ZERO THE ERROR FLAG
8977 054500 005067 124622 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
8978 054504 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
8979 054506 032737 004000 177570 3$: BIT #BIT11,@$SWR ;;INHIBIT ITERATIONS?
8980 054514 001011 BNE 1$ ;;BR IF YES
8981 054516 105767 124456 TSTB $PASS ;;IF FIRST PASS OF PROGRAM
8982 054522 001406 BEQ 1$ ;; INHIBIT ITERATIONS
8983 054524 005267 124456 INC $ICNT ;;INCREMENT ITERATION COUNT
8984 054530 026767 124572 124450 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
8985 054536 002017 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
8986 054540 012767 000001 124440 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
8987 054546 016767 000046 124552 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
8988 054554 011667 124430 $SVLAD: MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
8989 054560 011667 124426 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
8990 054564 005067 124540 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
8991 054570 112767 000001 124421 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
8992 054576 105767 124402 $OVER: TSTB $ERFLG ;;ANY ERRORS?
8993 054602 001403 BEQ 1$ ;;BRANCH IF NO
8994 054604 116737 124374 001203 MOVB $ERFLG,@$STSTNM+1
8995 054612 016716 124372 1$: MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
8996 054616 000002 RTI ;;FIXES PS
8997 054620 000010 $MXCNT: 10 ;;MAX. NUMBER OF ITERATIONS
8998
8999
9000
9001
9002
9003
9004
9005
9006
9007
9008
9009
9010
9011
9012
9013
9014
9015
9016
9017
9018
9019
9020
9021
9022
9023
9024
9025
9026
9027
9028
9029
```

```
.SBTTL ERROR HANDLER ROUTINE

;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;*AND GO TO $ERRTYP ON ERROR
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW15=1 HALT ON ERROR
;* HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
;*SW13=1 INHIBIT ERROR TYPEOUTS
;*SW10=1 BELL ON ERROR
;*SW09=1 LOOP ON ERROR
;*CALL
;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
7$: MOVB $ERFLG,@$STSTNM+1
INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $STSTNM,@$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
TST @$SWR ;;HALT ON ERROR - 1?
BPL 8$ ;;BRANCH IF NO
HALT ;;YES--HALT
8$: BIT #BIT10,@$SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE ,SBELL ;;RING BELL
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB @$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,@$SWR ;;SKIP TYPEOUT IF SET
```

```
9030 054722 001004      BNE      2$          ;;SKIP TYPEOUTS
9031 054724 004767 000056 JSR      PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
9032 054730 104400 001337      TYPE    ,SCLRF
9033 054734 005737 177570      TST     @PSWR      ;;HALT ON ERROR
9034 054740 100001      BPL     9$          ;;SKIP IF CONTINUE
9035 054742 000000      HALT
9036 054744 022767 046570 123070 9$:      CMP     #SENDAD,42 ;;HALT ON ERROR'
9037 054752 001001      BNE     3$          ;;ACT-11?
9038 054754 000000      HALT              ;;BRANCH IF NO
9039 054756 032737 001000 177570 3$:      BIT     #BIT09,@PSWR ;;YES
9040 054764 001402      BEQ     4$          ;;LOOP ON ERROR SWITCH SET?
9041 054766 016716 124220      MOV     $LPERR,(SP) ;;BR IF NO
9042 054772 005767 124332      TST     $ESCAPE    ;;FUDGE RETURN FOR LOOPING
9043 054776 001402      BEQ     5$          ;;CHECK FOR AN ESCAPE ADDRESS
9044 055000 016716 124324      MOV     $ESCAPE,(SP) ;;BR IF NONE
9045 055004      5$:      ;;FUDGE RETURN ADDRESS FOR ESCAPE
9046 055004 000002      RTI              ;;RETURN
```

:SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```
9047  
9048  
9049  
9050  
9051  
9052  
9053  
9054  
9055  
9056  
9057  
9058  
9059  
9060  
9061  
9062  
9063  
9064  
9065  
9066  
9067  
9068  
9069  
9070  
9071  
9072  
9073  
9074  
9075
```

;*THIS ROUTINE FIRST TYPES A STANDARD MESSAGE CONSISTING OF THE
;*VIRTUAL PC, THE PHYSICAL PC, THE PSW AT THE TIME OF THE ERROR CALL,
;*AND THE SUB-PASS COUNT. THE SUB-PASS COUNT CONSISTS OF THE SUB PASS COUNT IN THE
;*HIGH BYTE AND THE PASS COUNT IN THE LOW BYTE.
;*IT THEN USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE"
;*THE ERROR MESSAGE POINTER AND TYPES THE ERROR MESSAGE. THE DATA
;*HEADER POINTER IS THEN OBTAINED AND A DATA HEADER IS TYPED.
;*THE DATA POINTER AND DATA FORMAT ARE THEN OBTAINED. THERE ARE
;*FOUR TYPES OF DATA FORMAT, AS FOLLOWS:

- 0 TYPE THE CONTENTS OF THE DATA TABLE WORD IN 6 DIGIT OCTAL FORMAT
- 1 CONVERT THE CONTENTS OF THE DATA TABLE WORD TO 22 BITS AND TYPE AN 8 DIGIT OCTAL NUMBER
- 2 TYPE THE CONTENTS OF THE DATA TABLE WORD AND THE WORD+2 IN 8 DIGIT OCTAL FORMAT
- 3 USE THE CONTENTS OF THE DATA TABLE WORD AS A DEVICE ID AND TYPE THE DEVICES NAME
- 4 CONVERT THE TWO WORDS POINTED TO BY THE DATA TABLE TO FLOATING POINT FORMAT AND TYPE.
- 5 CONVERT THE FOUR WORDS POINTED TO BY THE DATA TABLE TO FLOATING DOUBLE FORMAT AND TYPE

```
9076 055006 104420      $ERRTYP:SAVREG
9077 055010 104400 001337      TYPE    ,SCLRF      ;;'"CARRIAGE RETURN"' & 'LINE FEED'
9078 055014 004737 056506      JSR     PC,@TYPTIME ;;GO TYPE THE TIME
9079 055020 104400 064510      TYPE    ,MSG3
9080 055024 104400 001337      TYPE    ,SCLRF
9081 055030 016746 124164      MOV     $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
9082      ;;TYPE THE VIRTUAL PC
9083 055034 104402      TYPOC   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
9084 055036 104400 056260      TYPE    ,8$
9085 055042 013700 001522      MOV     @VADR,R0    ;SAVE VADR
```



```
9086 055046 013737 001220 001522      MOV      @SERRPC,@VADR      ;SAVE THE VIR PC FOR CONVERSION
9087 055054 122737 000014 001216      CMPB     #14,@SITEMB
9088 055062 003403                      BLE      51$
9089 055064 105737 001216      TSTB     @SITEMB          ;ERROR ZERO?
9090 055070 001005                      BNE      42$              ;BRANCH IF NO
9091 055072 004737 062432 51$:    JSR      PC,@CNVADR        ;CONVERT TO 22 BITS
9092 055076 010037 001522      MOV      R0,@VADR
9093 055102 000407                      BR       41$
9094 055104 013737 001522 001524 42$:    MOV      @VADR,@PA1500
9095 055112 005037 001526      CLR      @PA2116
9096 055116 010037 001522      MOV      R0,@VADR
9097 055122 012746 001524 41$:    MOV      #PA1500,-(SP)    ;PUT ADDRESS OFPC ON STACK
9098 055126 004737 057544      JSR      PC,@SDB20        ;CONVERT TO ASCII
9099 055132 062716 000003      ADD      #3,(SP)          ;GET RID OF 3 MS DIGITS
9100 055136 012667 000002      MOV      (SP)+,30$        ;SAVE POINTER TO ASCII
9101 055142 104400                      TYPE
9102 055144 000000 30$:    .WORD
9103 055146 104400 056260      TYPE     ,8$
9104 055152 016646 000030      MOV      30(SP),-(SP)    ;GET PSW AT TIME OF ERROR
9105 055156 104402                      TYPOC    ;TYPE IT
9106 055160 104400 056260      TYPE     ,8$
9107 055164 016746 124450      MOV      $MAINT,-(SP)    ;;SAVE $MAINT FOR TYPEOUT
9108                                ;;TYPE THE MAINTENANCE REG
9109                                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
9110 055170 104402                      TYPOC
9111 055172 104400 056260      TYPE     ,8$
9112 055176 116746 124000      MOV      $TSTNM,-(SP)
9113 055202 105066 000001      CLRB     1(SP)
9114 055206 104402                      TYPOC    ;TYPE THE TEST NUMBER
9115 055210 104400 056260      TYPE     ,8$
9116 055214 013746 001560      MOV      @SUBPASS,-(SP)
9117 055220 162716 000060      SUB      #60,(SP)
9118 055224 104402                      TYPOC
9119 055226 104400 056260      TYPE     ,8$
9120 055232 016746 123742      MOV      $PASS,-(SP)    ;;SAVE $PASS FOR TYPEOUT
9121                                ;;TYPE THE PASS COUNT
9122                                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
9123 055236 104402                      TYPOC
9124 055240 104400 001337      TYPE     ,$CRLF
9125 055244 005000                      CLR      R0
9126 055246 153700 001216      BISB     @SITEMB,R0      ;PICK UP THE INDEX
9127 055252 001431                      BEQ      6$              ;EXIT IF ZERO
9128 055254 022700 000007 1$:    CMP      #7,R0          ;IS THIS ERROR ?
9129 055260 001551                      BEQ      15$            ;BRANCH IF YES
9130 055262 005300                      DEC      R0              ;;ADJUST THE INDEX SO THAT IT WILL
9131 055264 006300                      ASL      R0              ;; WORK FOR THE ERROR TABLE
9132 055266 006300                      ASL      R0
9133 055270 006300                      ASL      R0
9134 055272 062700 002344      ADD      #SERRTB,R0     ;;FORM TABLE POINTER
9135 055276 012067 000004      MOV      (R0)+,2$      ;;PICKUP 'ERROR MESSAGE' PCINTER
9136 055302 001404                      BEQ      3$              ;;SKIP TYPEOUT IF NO POINTER
9137 055304 104400                      TYPE     ;TYPE THE 'ERROR MESSAGE'
9138 055306 000000 2$:    .WORD     0              ;;'ERROR MESSAGE' POINTER GOES HERE
9139 055310 104400 001337      TYPE     , $CRLF        ;;'CARRIAGE RETURN' & 'LINE FEED'
9140 055314 012067 000004 3$:    MOV      (R0)+,4$      ;;PICKUP 'DATA HEADER' POINTER
9141 055320 001404                      BEQ      5$              ;;SKIP TYPEOUT IF 0
9142 055322 104400                      TYPE     ;TYPE THE 'DATA HEADER'
9143 055324 000000 4$:    .WORD     0              ;;'DATA HEADER' POINTER GOES HERE
```

```
9142 055326 104400 001337          TYPE      ,SCLF          ;:'CARRIAGE RETURN' & 'LINE FEED'  
9143 055332 012001          5$:  MOV      (R0)+,R1          ;:PICKUP 'DATA TABLE' POINTER  
9144 055334 001004          BNE      7$                  ;:GO TYPE THE DATA  
9145 055336 104422          6$:  RESREG  
9146 055340 104400 001337          TYPE      ,SCLF          ;:'CARRIAGE RETURN' & 'LINE FEED'  
9147 055344 000207          RTS      PC                  ;:RETURN  
9148 055346 011002          7$:  MOV      (R0),R2          ;:GET 'DATA FORMAT' POINTER  
9149 055350 122712 000001          10$: CMPB     #1,(R2)          ;:DATA FORMAT 1?  
9150 055354 001424          BEQ     9$                  ;:BRANCH IF YES  
9151 055356 122712 000002          CMPB     #2,(R2)          ;:DATA FORMAT 2?  
9152 055362 001441          BEQ     11$                 ;:BRANCH IF YES  
9153 055364 122712 000003          CMPB     #3,(R2)          ;:DATA FORMAT 3?  
9154 055370 001445          BEQ     24$                 ;:BRANCH IF YES  
9155 055372 122712 000004          CMPB     #4,(R2)          ;:DATA FORMAT 4?  
9156 055376 001456          BEQ     40$                 ;:BRANCH IF YES  
9157 055400 122712 000005          CMPB     #5,(R2)          ;:DATA FORMAT 5?  
9158 055404 001465          BEQ     60$                 ;:BRANCH IF YES  
9159  
9160          ;:*****  
9161 055406 005202          ;:DATA FORMAT 0  
9162 055410 013146          INC      R2                  ;:INCREMENT FORMAT POINTER  
9163 055412 104402          MOV      @ (R1)+, - SP      ;:PUSH DATA TO BE TYPED  
9164 055414 005711          TST      (R1)                ;:ANY MORE DATA?  
9165 055416 001747          BEQ     6$                  ;:BRANCH IF NO  
9166 055420 104400 056260          TYPE     ,8$                ;:TYPE TWO SPACES  
9167 055424 000757          BR      10$                 ;:*****  
9168  
9169          ;:DATA FORMAT 1  
9170 055426 005202          9$:  INC      R2                  ;:INCREMENT FORMAT POINTER  
9171 055430 004737 062432          JSR     PC,@#CNVADR          ;:GET 22 BIT ADR  
9172 055434 012746 001524          14$: MOV      #PA1500,-(SP)    ;:PUSH ADR OF 22 BIT ADR  
9173 055440 004737 057544          JSR     PC,@#SDB20          ;:CONVERT TO ASCII  
9174 055444 062716 000003          ADD     #3,(SP)             ;:DELETE LEADING ZEROS  
9175 055450 012667 000002          MOV     (SP)+,12$           ;:GET ADR OF ASCII STRING  
9176 055454 104400          TYPE  
9177 055456 000000          12$: .WORD  
9178 055460 062701 000002          ADD     #2,R1                ;:INCREMENT R1  
9179 055464 000753          BR      13$                 ;:*****  
9180  
9181          ;:DATA FORMAT 2  
9182 055466 005202          11$: INC      R2                  ;:INCREMENT FORMAT POINTER  
9183 055470 011100          MOV     (R1),R0              ;:  
9184 055472 012037 001524          MOV     (R0)+,@#PA1500      ;:  
9185 055476 011037 001526          MOV     (R0),@#PA2116       ;:  
9186 055502 000754          BR      14$                 ;:*****  
9187  
9188          ;:DATA FORMAT 3  
9189 055504 005202          24$: INC      R2                  ;:INCREMENT FORMAT POINTER  
9190 055506 013167 000016          MOV     @ (R1)+,25$         ;:GET DEVICE ID  
9191 055512 062767 064440 000010          ADD     #MSGINX,25$         ;:FORM ADR OF ASCII7 ADR  
9192 055520 017767 000004 000002          MOV     @25$,25$           ;:GET ADR OF ASCII7  
9193 055526 104400          TYPE  
9194 055530 000000          25$: .WORD  
9195 055532 000730          BR      13$                 ;:CONTINUE  
9196  
9197          ;:*****  
          ;:DATA FORMAT 4
```

9198 055534 005202
9199 055536 012167 000002
9200 055542 104424
9201 055544 000000
9202 055546 012667 000002
9203 055552 104400
9204 055554 000000
9205 055556 000716
9206
9207
9208 055560 005202
9209 055562 012167 000002
9210 055566 104426
9211 055570 000000
9212 055572 012667 000002
9213 055576 104400
9214 055600 000000
9215 055602 000704
9216
9217
9218 055604 010300
9219 055606 062700 064440
9220 055612 011067 000002
9221 055616 104400
9222 055620 000000
9223 055622 104400 055630
9224 055626 000404
9225
9226 055640
9227 055640 010300
9228 055642 022700 000010
9229 055646 003403
9230 055650 104400 064627
9231 055654 000411
9232
9233
9234 055656 022703 000020
9235 055662 001426
9236 055664 002435
9237 055666 104400 064742
9238 055672 022700 000012
9239 055676 001140
9240
9241
9242 055700 062700 064414
9243 055704 011000
9244 055706 022703 000002
9245 055712 001404
9246 055714 100406
9247 055716 012704 000007
9248 055722 000423
9249 055724 012704 000006
9250 055730 000420
9251 055732 012704 000011
9252 055736 000415
9253

```
40$: INC R2
      MOV (R1)+,44$ :GET ADDRESS OF DATA
      FL20 :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$: MOV R3,R0 :SAVE R3
      ADD #MSGINX,R0 :GEN ADRS OF ASCIZ
      MOV (R0),16$
      TYPE
16$: .WORD
      TYPE ,65$ :TYPE ASCIZ STRING
      BR 64$ :GET OVER THE ASCIZ
:65$: .ASCIZ /FAILED/<CRLF>
64$: MOV R3,R0 :SAVE DEVICE ID
      CMP #10,R0 :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,R0 :WAS IT RS?
      BNE 29$ :BRANCH IF NO
:*****
:UNIBUS ERROR OR RS04 ERROR
18$: ADD #REGINX,R0 :FORM ADR OF REG TABLE
      MOV (R0),R0 :GET ADR OF REG TABLE
      CMP #2,R3 :RP3 OR RK?
      BEQ 20$ :BRANCH IF RK
      BMI 21$ :BRANCH IF NOT RP03
      MOV #7,R4 :SET RP03 SOB COUNT
      BR 22$
20$: MOV #6,R4 :SET RK05 SOB COUNT
      BR 22$
21$: MOV #11,R4 :SET RS04 SOB COUNT
      BR 22$
:*****
```

9254
9255 055740 104400 065131
9256 055744 012704 000011
9257 055750 062700 064414
9258 055754 011000
9259 055756 000405
9260
9261 055760 104400 065240
9262 055764 012704 000004
9263 055770 000767
9264 055772 013046
9265 055774 104402
9266 055776 104400 056260
9267 056002 077405
9268
9269
9270 056004 022703 000022
9271 056010 001454
9272 056012 022703 000002
9273 056016 002445
9274 056020 001005
9275
9276 056022 104400 065434
9277 056026 012700 002166
9278 056032 000404
9279
9280 056034 012700 002144
9281 056040 104400 065444
9282
9283 056044 013001
9284 056046 005720
9285 056050 013037 001732
9286 056054 072127 177774
9287 056060 042701 177774
9288 056064 010137 001734
9289 056070 162737 000002 001732
9290 056076 005637 001734
9291 056102 004737 062324
9292 056106 012746 001524
9293 056112 004737 057544
9294 056116 062716 000003
9295 056122 012667 000002
9296 056126 104400
9297 056130 000000
9298 056132 104400 001337
9299 056136 000167 177174
9300
9301 056142 012700 002272
9302 056146 013037 001732
9303 056152 013037 001734
9304 056156 042737 177774 001734
9305 056164 162737 000002 001732
9306 056172 005637 001734
9307 056176 000734
9308
9309

```
:MBT ERROR
26$:  TYPE      ,MSG16
      MOV      #11,R4          :SET MBT SOB COUNT
28$:  ADD      #REGINX,R0
      MOV      (R0),R0        :GET ADR OF MBT TABLE
      BR      22$            :GO TYPE REGISTERS

:UN!BUS EXERCISER ERROR
27$:  TYPE      ,MSG17
      MOV      #4,R4          :SET UBE SOB COUNT
      BR      28$            :GO TYPE UBE REGISTERS
22$:  MOV      @ (R0)+,-(SP)  :GET DATA IN REG
      TYPCC
      TYPE     ,8$           :TYPE IT
      SOB     R4,22$        :TYPE TWO SPACES
      SOB     R4,22$        :CONTINUE

:*****
:THIS CODE TYPES A PHYSICAL BUS ADDRESS IF THE ERROR WAS AN RP03, RK05, OR UBE
      CMP     #22,R3          :UBE ERROR?
      BEQ     73$            :BRANCH IF YES
      CMP     #2,R3          :RK05?
      BLT     32$            :BRANCH IF NOT RK OR RP03
      BNE     70$            :BRANCH IF RP03

:RK05 ERROR
      TYPE     ,MSG22
      MOV     #RKCS,R0       :GET ADR OF ADR OF RKCS REG
      BR     71$

:RP03 ERROR
70$:  MOV     #RP3CS,R0      :GET ADR OF ADR OF RP3CS REG
      TYPE     ,MSG23

:GET, CALCULATE, & TYPE PHYSICAL BUS ADDRESS
71$:  MOV     @ (R0)+,R1      :GET BUS ADR EXTENDED BITS
      TST     (R0)+         :ADJUST R0
      MOV     @ (R0)+,@ERRBA :GET BUS ADDRESS THAT FAILED
      ASH     #-4,R1        :GET BITS 4&5 INTO BITS 0&1
      BIC     #177774,R1    :GET RID OF UNUSED BITS
      MOV     R1,@ERRBA+2   :SAVE EXTENDED BITS
74$:  SUB     #2,@ERRBA      :DECREMENT BUS ADR
      SBC     @ERRBA+2
      JSR     PC,@PHYMAP    :GO CONVERT TO 22 BIT PHYSICAL
      MOV     #PA1500,-(SP)
      JSR     PC,@SDB20     :CONVERT TO ASCIZ STRING
      ADD     #3,(SP)       :GET RID OF LEADING ZEROS
      MOV     (SP)+,72$

72$:  .WORD
32$:  TYPE     ,SCLF
      JMP     6$           :EXIT

:GET UBE VIRTUAL ADDRESS
73$:  MOV     #UBETBL+2,R0   :GET ADR OF UBE TABLE +2
      MOV     @ (R0)+,@ERRBA :GET BUS ADR THAT FAILED
      MOV     @ (R0)+,@ERRBA+2:GET BAE BITS
      BIC     #177774,@ERRBA+2:MASK OFF ADR BITS
      SUB     #2,@ERRBA
      SBC     @ERRBA+2
      BR     74$          :GO CONVERT & TYPE PHYSICAL ADR

:*****
:RP04 ERROR
```

```

9310 056200 062700 064414
9311 056204 011000
9312 056206 012704 000011
9313 056212 013046
9314 056214 104402
9315 056216 104400 056260
9316 056222 077405
9317 056224 104400 001337
9318 056230 104400 001337
9319 056234 012704 000004
9320 056240 104400 065052
9321 056244 013046
9322 056246 104402
9323 056250 104400 056260
9324 056254 077405
9325 056256 000725
9326 056260 020040 000
9327 056264
9328
9329
9330
9331
9332
9333
9334
9335
9336
9337
9338
9339
9340
9341
9342
9343
9344
9345
9346
9347
9348
9349
9350 056264 05767 122763
9351 056270 100002
9352 056272 000000
9353 056274 000407
9354 056276 010046
9355 056300 017600 000002
9356 056304 112046
9357 056306 001005
9358 056310 005726
9359 056312 012600
9360 056314 062716 000002
9361 056320 000002
9362 056322 122716 000011
9363 056326 001426
9364 056330 122716 000200
9365 056334 001004

```

```

29$: ADD #REGINX,R0
MOV (R0),R0 ;FORM ADR OF RPO4 TABLE
MOV #11,R4 ;SET SOB COUNT
31$: MOV @ (R0)+,-(SP) ;GET DATA TO BE TYPED
TPOC ;TYPE DATA
TYPE ,8$
SOB R4,31$ ;CONTINUE
TYPE ,SCLRF
TYPE ,SCLRF
MOV #4,R4 ;SET SOB COUNT
TYPE ,MSG14
50$: MOV @ (R0)+,-(SP) ;GET DTA TO BE TYPED
TPOC ;TYPE IT
TYPE ,8$
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 ADRESS OF MESSAGE
$TYPE: TSTB $TPFLG ;:IS THERE A TERMINAL?
BPL 1$ ;:BR IF YES
HALT ;:HALT HERE IF NO TERMINAL
BR 3$ ;:LEAVE
1$: MOV R0,-(SP) ;:SAVE R0
MOV @2(SP),R0 ;:GET ADDRESS OF ASCIZ STRING
2$: MOVB (R0)+,-(SP) ;:PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;:BR IF IT ISN'T THE TERMINATOR
ST (SP)+ ;:IF TERMINATOR POP IT OFF THE STACK
MOV (SP)+,R0 ;:RESTORE R0
3$: ADD #2,(SP) ;:ADJUST RETURN PC
RTI ;:RETURN
4$: CMPB #HT,(SP) ;:BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;:BRANCH IF NLT
BNE 5$

```

```
9366 056336 005726          TST      (SP)+          ;;POP <CR><LF> EQUIV
9367 056340 104400 001337    TYPE     ,SCLRF
9368 056344 000757          BR       2$            ;;GET NEXT CHARACTER
9369 056346 004767 000056    5$:     JSR      PC,$TYPEC ;;GO TYPE THIS CHARACTER
9370 056352 126726 122674    6$:     CMPB     $FI LC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
9371 056356 001352          BNE     2$            ;;IF NO GO GET NEXT CHAR.
9372 056360 016746 122664          MOV     $NULL,-(SP)    ;;GET # OF FILLER CHARS. NEEDED
9373                                ;;AND THE NULL CHAR.
9374 056364 105366 000001    7$:     DECB     1(SP)   ;;DOES A NULL NEED TO BE TYPED?
9375 056370 002770          BLT     6$            ;;BR IF NO--GO POP THE NULL OFF OF STACK
9376 056372 004767 000032    JSR     PC,$TYPEC    ;;GO TYPE A NULL
9377 056376 105367 000100    DECB     $CHARCNT    ;;DON'T COUNT THE NULL AS A CHARACTER
9378 056402 000770          BR      7$            ;;LOOP
9379
9380                                ;;HORIZONTAL TAB PROCESSOR
9381
9382 056404 112716 000040    8$:     MOVB     #' ,(SP) ;;REPLACE TAB WITH SPACE
9383 056410 004767 000014    9$:     JSR     PC,$TYPEC ;;TYPE A SPACE
9384 056414 132767 000007 000060    BITB     #7,$CHARCNT ;;BRANCH IF NOT AT
9385 056422 001372          BNE     9$            ;;TAB STOP
9386 056424 005726          TST     (SP)+          ;;POP SPACE OFF STACK
9387 056426 000726          BR      2$            ;;GET NEXT CHARACTER
9388 056430 005737 001476    $TYPEC: TST     @NOTYPE    ;;INHIBIT TYPING?
9389 056434 100423          BMI     $TYPEX        ;;BRANCH IF YES
9390 056436 105777 122602    TSTB     @STPS        ;;WAIT UNTIL PRINTER IS READY
9391 056442 100372          BPL     $TYPEC
9392 056444 116677 000002 122574    MOVB     2(SP),@STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
9393 056452 122766 000015 000002    CMPB     #CR,2(SP)   ;;BRANCH IF
9394 056460 001003          BNE     1$            ;;NOT <CR>
9395 056462 105067 000014    CLRB     $CHARCNT    ;;
9396 056466 000406          BR      $TYPEX        ;;EXIT
9397 056470 122766 000012 000002    1$:     CMPB     #LF,2(SP) ;;BRANCH IF
9398 056476 001402          BEQ     $TYPEX        ;;<LF>
9399 056500 105227          INCB     (PC)+        ;;INC SPACE
9400 056502 000000    $CHARCNT: .WORD 0    ;;COUNT
9401 056504 000207    $TYPEX: RTS     PC
9402
9403
9404                                ;*****
9405                                ;SBTTL ROUTINE TO TYPE THE ELAPSED RUN TIME OF THE PROGRAM
9406                                ;* THIS ROUTINE CONVERTS THE CONTENTS OF LOCATIONS 'LTICKS'
9407                                ;* AND 'MTICKS' TO SECONDS AND MINUTES/HOURS RESPECTIVELY
9408                                ;* AND TYPES THEM IN THE FOLLOWING FORMAT:
9409                                ;* HHH:MM:SS
9410                                ;*****
9411 056506 104420    *YPTIME.SAVREG
9412 056510 004737 063752    JSR     PC,@LDKT      ;GO BACK TO LOW CORE
9413 056514 113701 001631    MOVB     @LTICKS+1,R1 ;GET SECOND COUNT
9414 056520 005000          CLR     R0
9415 056522 071027 000012    DIV     #10,R0
9416 056526 062701 000060    ADD     #60,R1
9417 056532 110137 056744    MOVB     R1,@TIMEBUF+10
9418 056536 010001          MOV     R0,R1
9419 056540 005000          CLR     R0
9420 056542 071027 000006    DIV     #6,R0
9421 056546 062701 000060    ADD     #60,R1
9422 056552 110137 056743    MOVB     R1,@TIMEBUF+7
```

9622	056556	013701	001626
9623	056562	005000	
9624	056564	071027	000012
9625	056570	062701	000060
9626	056574	110167	000141
9627	056600	010001	

MOV	20MTICKS,R1	:GET MINUTE COUNT
CLR	R0	
DIV	#10,R0	:GET HOURS AND MINUTES
ADD	#60,R1	:MAKE REMAINDER ASCII
MOVB	R1,TIMEBUF+5	:PUT IN BUFFER
MOV	R0,R1	

9428	056602	005000		CLR	R0
9429	056604	071027	000006	DIV	#6,R0
9430	056610	062701	000060	ADD	#60,R1
9431	056614	110167	000120	MOVB	R1,TIMEBUF+4
9432	056620	005700		TST	R0
9433	056622	001434		BEQ	2\$
9434	056624	010001		MOV	R0,R1

DEQKC-D PDP 11/70-74MP CPU EXERCISER
CEQKCD.P11 04-OCT-79 08:55

M 16
MACY11 30A(1052) 04-OCT-79 09:00 PAGE 185
ROUTINE TO TYPE THE ELAPSED RUN TIME OF THE PROGRAM

SEQ 0207

9435 056626 005000

CLR RC

9436	056630	071027	000012
9437	056634	062701	000060
9438	056640	110167	000072
9439	056644	005700	
9440	056646	001422	
9441	056650	010001	
9442	056652	005000	
9443	056654	071027	000010
9444	056660	062701	000060
9445	056664	110167	000045
9446	056670	005700	
9447	056672	001410	
9448	056674	010001	
9449	056676	005000	
9450	056700	071027	000012
9451	056704	062701	000060
9452	056710	110167	000020
9453	056714	104400	056734
9454	056720	104400	001337
9455	056724	004737	064050
9456	056730	104422	
9457	056732	000207	
9458	056734	001	001
9459	056737	072	001
9460	056742	072	060
9461	056745	000	060
9462			
9463			
9464			
9465			
9466			
9467			
9468			
9469			
9470	056746	104400	067176
9471	056752	104400	070517
9472	056756	104400	070364
9473	056762	012700	000010
9474	056766	005001	
9475	056770	105761	001642
9476	056774	001004	
9477	056776	062701	000002
9478	057002	077006	
9479	057004	000207	
9480	057006	010102	
9481	057010	062702	064440
9482	057014	011267	000002
9483	057020	104400	
9484	057022	000000	
9485	057024	112767	000060
9486	057032	116102	001642
9487	057036	012703	000010
9488	057042	006002	
9489	057044	103002	
9490	057046	104400	057066
9491	057052	005267	000010

```

DIV #10,R0
ADD #60,R1
MOVB R1,TIMEBUF+2
TST R0
BEQ 2$
MOV R0,R1
CLR R0
DIV #10,R0
ADD #60,R1
MOVB R1,TIMEBUF+1
TST R0
BEQ 2$
MOV R0,R1
CLR R0
DIV #10,R0
ADD #60,R1
MOVB R1,TIMEBUF
2$: TYPE ,TIMEBUF
TYPE ,SCLF
JSR PC,@RESK1 ;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
* DEVICE AND THE UNIT NUMBERS THAT WERE FOUND TO BE
* AVAILABLE FOR THAT DEVICE.
*****
TYPsiz: TYPE ,SWITCH
TYPE ,MSG30 ;NOTE SWITCH REG BIT 8 REVERSAL
TYPE ,MSG4
MOV #10,R0 ;SET SOB COUNT
CLR R1
1$: TSTB SYSSIZE(R1) ;DEVICE AVAILABLE?
BNE 2$ ;BRANCH IF YES
7$: ADD #2,R1 ;INCREMENT INDEX
SOB R0,1$ ;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$: INC 4$
    
```

```
9492 057056 077307          SOB      R3,6$          ;CONTINUE
9493 057060 104400 001337    TYPE    $CRLF
9494 057064 000744          BR      7$
9495 057066          000      054      040 4$: .BYTE 0,54,40,0 ;NUMBER,COMMA,SPACE,TERMINATOR
9496 057071          000
9497
9498
9499
9500
9501
9502
9503
9504
9505
9506
9507
9508
9509
9510
9511
9512
9513
9514
9515
9516
9517
9518
9519
9520
9521
9522
9523 057072 017646 000000          $TYPOS: MOV    @ (SP),-(SP) ;:PICKUP THE MODE
9524 057076 116667 000001 000211    MOV    1(SP), $OFILL ;:LOAD ZERO FILL SWITCH
9525 057104 112667 000207          MOV    (SP)+, $OMODE+1 ;:NUMBER OF DIGITS TO TYPE
9526 057110 062716 000002          ADD    #2, (SP) ;:ADJUST RETURN ADDRESS
9527 057114 000406          BR     $TYPON
9528 057116 112767 000001 000171    $TYPOC: MOV    #1, $OFILL ;:SET THE ZERO FILL SWITCH
9529 057124 112767 000006 000165    MOV    #6, $OMODE+1 ;:SET FOR SIX(6) DIGITS
9530 057132 112767 000005 000154    $TYPON: MOV    #5, $OCNT ;:SET THE ITERATION COUNT
9531 057140 010346          MOV    R3, -(SP) ;:SAVE R3
9532 057142 010446          MOV    R4, -(SP) ;:SAVE R4
9533 057144 010546          MOV    R5, -(SP) ;:SAVE R5
9534 057146 116704 000145          MOV    $OMODE+1, R4 ;:GET THE NUMBER OF DIGITS TO TYPE
9535 057152 005404          NEG    R4
9536 057154 062704 000006          ADD    #6, R4 ;:SUBTRACT IT FOR MAX. ALLOWED
9537 057160 110467 000132          MOV    R4, $OMODE ;:SAVE IT FOR USE
9538 057164 116704 000125          MOV    $OFILL, R4 ;:GET THE ZERO FILL SWITCH
9539 057170 016605 000012          MOV    12(SP), R5 ;:PICKUP THE INPUT NUMBER
9540 057174 005003          CLR    R3 ;:CLEAR THE OUTPUT WORD
9541 057176 006105          1$: ROL    R5 ;:ROTATE MSB INTO 'C'
9542 057200 000404          BR     3$ ;:GO DO MSB
9543 057202 006105          2$: ROL    R5 ;:FORM THIS DIGIT
9544 057204 006105          ROL    R5
9545 057206 006105          ROL    R5
9546 057210 010503          MOV    R5, R3
9547 057212 006103          3$: ROL    R3 ;:GET LSB OF THIS DIGIT
```

```

9548 057214 1 5367 000076      DECB  $OMODE      ;;TYPE THIS DIGIT?
9549 057220 100016             BPL    7$         ;;BR IF NO
9550 057222 042703 177770     BIC    #177770,R3 ;;GET RID OF JUNK
9551 057226 001002             BNE    4$         ;;TEST FOR 0
9552 057230 005704             TST    R4         ;;SUPPRESS THIS 0?
9553 057232 001403             BEQ    5$         ;;BR IF YES
9554 057234 005204             4$: INC    R4         ;;DON'T SUPPRESS ANYMORE 0'S
9555 057236 052703 000060     BIS    #'0,R3    ;;MAKE THIS DIGIT ASCII
9556 057242 052703 000040     5$: BIS    #' ,R3  ;;MAKE ASCII IF NOT ALREADY
9557 057246 110367 000040     MOVB   R3,8$     ;;SAVE FOR TYPING
9558 057252 104400 057312     TYPE   ,8$      ;;GO TYPE THIS DIGIT
9559 057256 105367 000032     7$: DECB  $OCNT   ;;COUNT BY 1
9560 057262 003347             BGT    2$         ;;BR IF MORE TO DO
9561 057264 002402             BLT    6$         ;;BR IF DONE
9562 057266 005204             INC    R4         ;;INSURE LAST DIGIT ISN'T A BLANK
9563 057270 000744             BR     2$         ;;GO DO THE LAST DIGIT
9564 057272 012605             6$: MOV    (SP)+,R5 ;;RESTORE R5
9565 057274 012604             MOV    (SP)+,R4  ;;RESTORE R4
9566 057276 012603             MOV    (SP)+,R3  ;;RESTORE R3
9567 057300 016666 000002 000004 MOV    2(SP),4(SP) ;;SET THE STACK FOR RETURNING
9568 057306 012616             MOV    (SP)+,(SP)
9569 057310 000002             RTI                    ;;RETURN
9570 057312 000             8$: .BYTE 0        ;;STORAGE FOR ASCII DIGIT
9571 057313 000             .BYTE 0        ;;TERMINATOR FOR TYPE ROUTINE
9572 057314 000             $OCNT: .BYTE 0   ;;OCTAL DIGIT COUNTER
9573 057315 000             $OFILL: .BYTE 0  ;;ZERO FILL SWITCH
9574 057316 000000             $OMODE: .WORD 0   ;;NUMBER OF DIGITS TO TYPE

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 ;*REPLACED WITH SPACES.

;*CALL:
 ;* MOV NUM,-(SP) ;;PUT THE BINARY NUMBER ON THE STACK
 ;* TYPDS ;;GO TO THE ROUTINE

```

9588 057320      $TYPDS:
9589 057320 010046      MOV    R0,-(SP)   ;;PUSH R0 ON STACK
9590 057322 010146      MOV    R1,-(SP)   ;;PUSH R1 ON STACK
9591 057324 010246      MOV    R2,-(SP)   ;;PUSH R2 ON STACK
9592 057326 010346      MOV    R3,-(SP)   ;;PUSH R3 ON STACK
9593 057330 010546      MOV    R5,-(SP)   ;;PUSH R5 ON STACK
9594 057332 012746 020200 MOV    #2020,-(SP) ;;SET BLANK SWITCH AND SIGN
9595 057336 016605 000020 MOV    20(SP),R5  ;;GET THE INPUT NUMBER
9596 057342 100004             BPL    1$         ;;BR IF INPUT IS POS.
9597 057344 005405             NEG    R5         ;;MAKE THE BINARY NUMBER POS.
9598 057346 112766 000055 000001 MOVB   #'-,1(SP)  ;;MAKE THE ASCII NUMBER NEG.
9599 057354 005000             1$: CLR    R0         ;;ZERO THE CONSTANTS INDEX
9600 057356 012703 057534      MOV    #$DBLK,R3 ;;SETUP THE OUTPUT POINTER
9601 057362 112723 000040      MOVB   #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
9602 057366 005002             2$: CLR    R2         ;;CLEAR THE BCD NUMBER
9603 057370 016001 057524      MOV    $DTBL(R0),R1 ;;GET THE CONSTANT

```

9604	057374	160105			3\$:	SUB	R1,R5	::FORM THIS BCD DIGIT
9605	057376	002402				BLT	4\$::BR IF DONE
9606	057400	005202				INC	R2	::INCREASE THE BCD DIGIT BY 1
9607	057402	000774				BR	3\$	
9608	057404	060105			4\$:	ADD	R1,R5	::ADD BACK THE CONSTANT
9609	057406	005702				TST	R2	::CHECK IF BCD DIGIT=0
9610	057410	001002				BNE	5\$::FALL THROUGH IF 0
9611	057412	105716				TSTB	(SP)	::STILL DOING LEADING 0'S?
9612	057414	100407				SMI	7\$::BR IF YES
9613	057416	106316			5\$:	ASLB	(SP)	::MSD?
9614	057420	103003				BCC	6\$::BR IF NO
9615	057422	116663	000J01	177777		MOVB	1(SP),-1(R3)	::YES--SET THE SIGN
9616	057430	052702	000060		6\$:	BIS	#'0,R2	::MAKE THE BCD DIGIT ASCII
9617	057434	052702	000040		7\$:	BIS	#',R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
9618	057440	110223				MOVB	R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
9619	057442	005720				TST	(R0)+	::JUST INCREMENTING
9620	057444	020027	000010			CMP	R0,#10	::CHECK THE TABLE INDEX
9621	057450	002746				BLT	2\$::GO DO THE NEXT DIGIT
9622	057452	003002				BGT	8\$::GO TO EXIT
9623	057454	010502				MOV	R5,R2	::GET THE LSD
9624	057456	000764				BR	6\$::GO CHANGE TO ASCII
9625	057460	105726			8\$:	TSTB	(SP)+	::WAS THE LSD THE FIRST NON-ZERO?
9626	057462	100003				BPL	9\$::BR IF NO
9627	057464	116663	177777	177776		MOVB	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
9628	057472	105013			9\$:	CLRB	(R3)	::SET THE TERMINATOR
9629	057474	012605				MOV	(SP)+,R5	::POP STACK INTO R5
9630	057476	012603				MOV	(SP)+,R3	::POP STACK INTO R3
9631	057500	012602				MOV	(SP)+,R2	::POP STACK INTO R2
9632	057502	012601				MOV	(SP)+,R1	::POP STACK INTO R1
9633	057504	012600				MOV	(SP)+,R0	::POP STACK INTO R0
9634	057506	104400	057534			TYPE	\$DBLK	::NOW TYPE THE NUMBER
9635	057512	016666	000002	000004		MOV	2(SP),4(SP)	::ADJUST THE STACK
9636	057520	012616				MOV	(SP)+,(SP)	
9637	057522	000002				RTI		::RETURN TO USER
9638	057524	023420			\$DTBL:	10000.		
9639	057526	001750				1000.		
9640	057530	000144				100.		
9641	057532	000012				10.		
9642	057534	000004			\$DBLK:	.BLKW 4		
9643						*****		
9644						*****		
9645						*****		
9646						*****		
9647						*****		
9648						*****		
9649						*****		
9650						*****		
9651						*****		
9652						*****		
9653						*****		
9654						*****		
9655	057544	104420				*****		
9656	057546	016601	000002			*****		
9657	057552	012705	057701			*****		
9658	057556	012704	000014			*****		
9659	057562	012703	177770			*****		

.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,@#\$DB20 :: CALL THE ROUTINE
;* RETURN :: THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

\$DR20: SAVREG :: SAVE ALL REGISTERS
 MOV 2(SP),R1 :: PICKUP THE POINTER TO LOW WORD
 MOV #\$OCTVL+13.,R5 :: POINTER TO DATA TABLE
 MOV #12.,R4 :: DO ELEVEN CHARACTERS
 MOV #^C7,R3 :: MASK

```
9660 057566 012100      MOV      (R1)+,R0      ;; LOWER WORD
9661 057570 012101      MOV      (R1)+,R1      ;; HIGH WORD
9662 057572 005002      CLR      R2            ;; TERMINATOR
9663 057574 110245      1$:     MOVVB   R2,-(R5)  ;; PUT CHARACTER IN DATA TABLE
9664 057576 010002      MOV      R0,R2        ;; GET THIS DIGIT
9665 057600 005304      DEC      R4            ;; COUNT THIS CHARACTER
9666 057602 003016      BGT     3$            ;; BR IF NOT THE LAST DIGIT
9667 057604 001414      SEQ     2$            ;; BR IF IT IS THE LAST DIGIT
9668 057606 005205      INC     R5            ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
9669 057610 010566 000002  MOV     R5,2(SP)      ;; ASCII CHAR. & PUT IT ON THE STACK
9670 057614 122765 000061 000003  CMPB   #61,3(R5)    ;; LAST NUBER LEGAL?
9671 057622 002003      BGE     4$            ;; BRANCH IF YES
9672 057624 112765 000060 000003  MOVVB  #60,3(R5)    ;; MAKE IT ZERO
9673 057632 104422      4$:     RESREG          ;; RESTORE ALL REGISTERS
9674 057634 000207      RTS     PC            ;; RETURN TO USER
9675 057636 006203      2$:     ASR     R3            ;; POSITION THE MASK FOR THE LAST DIGIT
9676 057640 006001      3$:     ROR     R1            ;; POSITION THE BINARY NUMBER FOR
9677 057642 006000      ROR     R0            ;; THE NEXT OCTAL DIGIT
9678 057644 006001      ROR     R1
9679 057646 006000      ROR     R0
9680 057650 006001      ROR     R1
9681 057652 006000      ROR     R0
9682 057654 040302      BIC     R3,R2        ;; MASK OUT ALL JUNK
9683 057656 062702 000060  ADD     #'0,R2        ;; MAKE THIS CHAR. ASCII
9684 057662 000744      BR      1$            ;; GO PUT IT IN THE DATA TABLE
9685 057664 000016      $OCTVL: .BLKB 14.    ;; RESERVE DATA TABLE
9686
9687
9688      .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
9689
9690      ;*SAVE R0-R5
9691      ;*CALL:
9692      ;*   SAVREG
9693      ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
9694      ;*
9695      ;*TOP---(+16)
9696      ;* +2---(+18)
9697      ;* +4---R5
9698      ;* +6---R4
9699      ;* +8---R3
9700      ;*+10---R2
9701      ;*+12---R1
9702      ;*+14---R0
9703
9704      $SAVREG:
9705      MOV     R0,-(SP)    ;; PUSH R0 ON STACK
9706      MOV     R1,-(SP)    ;; PUSH R1 ON STACK
9707      MOV     R2,-(SP)    ;; PUSH R2 ON STACK
9708      MOV     R3,-(SP)    ;; PUSH R3 ON STACK
9709      MOV     R4,-(SP)    ;; PUSH R4 ON STACK
9710      MOV     R5,-(SP)    ;; PUSH R5 ON STACK
9711      MOV     22(SP),-(SP) ;; SAVE PS OF MAIN FLOW
9712      MOV     22(SP),-(SP) ;; SAVE PC OF MAIN FLOW
9713      MOV     22(SP),-(SP) ;; SAVE PS OF CALL
9714      MOV     22(SP),-(SP) ;; SAVE PC OF CALL
9715      RTI
```

```

9716
9717
9718
9719
9720 057740
9721 057740 012666 000022
9722 057744 012666 000022
9723 057750 012666 000022
9724 057754 012666 000022
9725 057760 012605
9726 057762 012604
9727 057764 012603
9728 057766 012602
9729 057770 012601
9730 057772 012600
9731 057774 000002
9732
9733
9734
9735
9736
9737
9738
9739
9740
9741
9742
9743
9744
9745
9746
9747
9748
9749 057776 104420
9750 060000 017600 000000
9751 060004 062716 000002
9752 060010 016001 000002
9753 060014 011000
9754 060016 012704 001367
9755 060022 112744 000000
9756 060026 012705 000005
9757 060032 010103
9758 060034 042703 177770
9759 060040 062703 000060
9760 060044 110344
9761 060046 073027 177775
9762 060052 077511
9763 060054 010103
9764 060056 042703 177776
9765 060062 062703 000060
9766 060066 110344
9767 060070 112744 000040
9768 060074 073027 177777
9769 060100 012705 000002
9770 060104 010103
9771 060106 042703 177770

```

```

;*RESTORE R0-R5
;*CALL:
;* RESREG
$RESREG:
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
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

;*****
;SBTL 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.
;*****
$FL20: SAVREG
MOV @ (SP),R0 ;;GET ADDRESS OF DATA
ADD #2,(SP) ;;ADJUST RETURN PC
MOV 2(R0),R1 ;;PUT SECOND DATA WORD IN R1
MOV (R0),R0 ;;PUT FIRST DATA WORD IN R0
MOV #$FLBUFF+23,R4 ;;GET ADDRESS OF BUFFER END IN R4
MOVB #0,-(R4) ;;PUT TERMINATOR IN BUFFER
MOV #5,R5 ;;SET SOB COUNT FOR FRACTION DIGITS
1$: MOV R1,R3 ;;GET LSB'S OF FRACTION
BIC #^C7,R3 ;;SAVE LS 3 BITS
ADD #60,R3 ;;MAKE THEM ASCII
MOVB R3,-(R4) ;;STORE IN BUFFER
ASHC #-3,R0 ;;SHIFT NUMBER TO NEXT 3 BITS
SOB R5,1$ ;;CONTINUE FOR 7 DIGITS
MOV R1,R3 ;;GET NEXT DIGITS
BIC #^C1,R3 ;;ONLY WANT 1 BIT
ADD #60,R3 ;;MAKE THEM ASCII
MOVB R3,-(R4) ;;STORE IN BUFFER
MOVB #40,-(R4) ;;PUT SPACE IN BUFFER
ASHC #-1,R0
MOV #2,R5 ;;SET SOB COUNT
3$: MOV R1,R3 ;;GET LOW WORD
BIC #^C7,R3 ;;MASK 3 BITS

```

```
9772 060112 062703 000060      ADD    #60,R3      ;MAKE THEM ASCII
9773 060116 110344      MOVB   R3,-(R4)    ;PUT IN BUFFER
9774 060120 073027 177775      ASHC  #3,R0      ;GET NEXT 3 BITS
9775 060124 077511      SOB   R5,3$      ;CONVERT THEM
9776 060126 010103      MOV   R1,R3      ;
9777 060130 042703 177776      BIC  #^C1,R3     ;ONLY WANT 1 BIT
9778 060134 062703 000060      ADD  #60,R3      ;MAKE IT ASCII
9779 060140 110344      MOVB  R3,-(R4)    ;PUT IN BUFFER
9780 060142 112744 000040      MOVB  #40,-(R4)   ;PUT SPACE IN BUFFER
9781 060146 112744 000040      MOVB  #40,-(R4)
9782 060152 072127 177777      ASH  #-1,R1     ;GET FIRST 3 BITS OF EXPONENT
9783 060156 012705 000002      MOV  #2,R5     ;SET SOB COUNT FOR 2 DIGITS
9784 060162 010103      2$: MOV  R1,R3     ;GET LSB'S OF EXPONENT
9785 060164 042703 177770      BIC  #^C7,R3     ;SAVE 3 BITS
9786 060170 062703 000060      ADD  #60,R3      ;MAKE THEM ASCII
9787 060174 110344      MOVB  R3,-(R4)    ;STORE IN BUFFER
9788 060176 072127 177775      ASH  #-3,R1     ;GET NEXT 3 BITS
9789 060202 077511      SOB  R5,2$      ;CONTINUE
9790 060204 010103      MOV  R1,R3     ;GET LAST 2 BITS OF EXPONENT
9791 060206 042703 177774      BIC  #^C3,R3     ;MAKE SURE ONLY 2 BITS
9792 060212 062703 000060      ADD  #60,R3      ;MAKE THEM ASCII
9793 060216 110344      MOVB  R3,-(R4)    ;STORE IN BUFFER
9794 060220 112744 000040      MOVB  #40,-(R4)   ;PUT SPACE IN BUFFER
9795 060224 112744 000040      MOVB  #40,-(R4)
9796 060230 042700 177776      BIC  #^C1,R0     ;GET SIGN BIT (IT WAS EXTENDED)
9797 060234 062700 000060      ADD  #60,R0      ;MAKE IT ASCII
9798 060240 110044      MOVB  R0,-(R4)    ;PUT IT IN THE BUFFER
9799 060242 104422      RESREG
9800 060244 011646      MOV  (SP),-(SP)  ;SAVE RETURN PC
9801 060246 016666 000004 000002  MOV  4(SP),2(SP) ;AND RETURN PSW
9802 060254 012766 001344 000004  MOV  #$FLBUFF,4(SP) ;PUT BUFFER ADDRESS ON STACK
9803 060262 000006      RTT             ;RETURN
9804 .EVEN
```

```
*****
;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.
*****
```

```
9824 060264 104420      $FLD20: SAVREG
9825 060266 017667 000000 000006  MOV  @2(SP),1$   ;GET ADDRESS OF DATA TO CONVERT
9826 060274 062716 000002      ADD  #2,(SP)     ;ADJUST RETURN PC
9827 060300 104424      FL20            ;CONVERT MS 32 BITS
```


9828	060302	000000		1\$:	.WORD		
9829	060304	012600			MOV	(SP)+,R0	:GET ADDRESS OF CONVERTED DATA
9830	060306	010067	121076		MOV	R0,\$BUFF	:SAVE IT
9831	060312	062700	000041		ADD	#41,R0	:ADJUST TO END OF BUFFER
9832	060316	105040			CLRB	-(R0)	:PUT TERMINATOR IN BUFFER
9833	060320	016701	177756		MOV	1\$,R1	:GET ADDRESS OF DATA TO CONVERT
9834	060324	062701	000004		ADD	#4,R1	:ADJUST TO LOWER 32 BITS
9835	060330	012102			MOV	(R1)+,R2	:SAVE THE DATA
9836	060332	012103			MOV	(R1)+,R3	
9837	060334	012701	000002		MUV	#2,R1	:SET LOOP COUNT
9838	060340	012704	000005	3\$:	MOV	#5,R4	:SET LOOP COUNT
9839	060344	010305		4\$:	MOV	R3,R5	:GET LS 32 BITS OF DATA
9840	060346	042705	177770		BIC	#^C7,R5	:MASK 3 BITS
9841	060352	062705	000060		ADD	#60,R5	:MAKE THEM ASCII
9842	060356	110540			MOVB	R5,-(R0)	:PUT IN BUFFER
9843	060360	073227	177775		ASHC	#-3,R2	:GET NEXT 3 BITS
9844	060364	077411			SOB	R4,4\$:CONTINUE
9845	060366	010305			MOV	R3,R5	:GET LS 32 BITS
9846	060370	042705	177776		BIC	#^C1,R5	:ONLY WANT 1 BIT
9847	060374	062705	000060		ADD	#60,R5	:MAKE IT ASCII
9848	060400	110540			MOVB	R5,-(R0)	:PUT IN TABLE
9849	060402	112740	000040		MOVB	#40,-(R0)	:PUT SPACE IN TABLE
9850	060406	073227	177777		ASHC	#-1,R2	
9851	060412	077126			SOB	R1,3\$:CONVERT NEXT 16 BITS
9852	060414	104422			RESREG		
9853	060416	011646			MOV	(SP),-(SP)	:ADJUST STACK
9854	060420	016666	000004 000002		MOV	4(SP),2(SP)	:TO RETURN WITH ADDRESS
9855	060426	016766	120756 000004		MOV	\$BUFF,4(SP)	:OF BUFFER ON STACK
9856	060434	000006			RTT		:RETURN

::*****

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

:*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 R0,-(SP) ::PUSH R0 ON STACK
 MOV R1,-(SP) ::PUSH R1 ON STACK
 MOV R2,-(SP) ::PUSH R2 ON STACK
 MOV \$LONUM,R0 ::SET R0 WITH LOW
 MOV \$HINUM,R1 ::SET R1 WITH HIGH
 MOV #-7,R2 ::SET SHIFT COUNT
 1\$: ASL R0 ::SHIFT R0 LEFT AND
 ROL R1 ::ROTATE CARRY INTO R1 AND
 INC R2 ::CHECK FOR DONE
 BNE 1\$::CONTINUE SHIFT LOOP
 ADD \$LONUM,R0 ::ADD NUMBER TO MAKE X 129
 ADC R1 ::PROPOGATE CARRY
 ADD \$HINUM,R1 ::ADD NUMBER TO MAKE X 129

9870	060436		
9871	060436	010046	
9872	060440	010146	
9873	060442	010246	
9874	060444	016700	121100
9875	060450	016701	121076
9876	060454	012702	177771
9877	060460	006300	
9878	060462	006101	
9879	060464	005202	
9880	060466	001374	
9881	060470	066700	121054
9882	060474	005501	
9883	060476	066701	121050

```

9884 060502 062700 001057      ADD    #1057,R0      ;;ADD LOW CONSTANT
9885 060506 005501      ADC    R1            ;;PROPUGATE CARRY
9886 060510 062701 047401      ADD    #47401,R1    ;;ADD HIGH CONSTANT
9887 060514 010067 121030      MOV    R0,$LONUM    ;;SAVE R0
9888 060520 010167 121026      MOV    R1,$HINUM    ;;SAVE R1
9889 060524 012602      MOV    (SP)+,R2     ;;POP STACK INTO R2
9890 060526 012601      MOV    (SP)+,R1     ;;POP STACK INTO R1
9891 060530 012600      MOV    (SP)+,R0     ;;POP STACK INTO R0
9892 060532 000207      RTS    PC           ;;RETURN
9893
9894      ;*****
9895      ;SBTTL FLOATING POINT NUMBER GENERATOR
9896      ;*
9897      ;* THIS ROUTINE GENERATES TWO RANDOM FLOATING POINT NUMBERS
9898      ;* IN EITHER SINGLE OR DOUBLE PRECISION. FOR SINGLE PRECISION
9899      ;* THE NUMBERS ARE STORED IN $TMP0 AND $TMP2. DOUBLE PRECISION
9900      ;* NUMBERS ARE STORED IN $TMP0 AND $TMP4.
9901      ;* IN EITHER SINGLE OR DOUBLE THE EXTENDED EXPONENT IS STORED
9902      ;* IN $REG0 AND $REG1.
9903      ;*****
9902 060534 012767 000002 000130  FLTDBL: MOV    #2,SOBDBL    ;SET LOOP FOR 2, FOUR WORD NUMBERS
9903 060542 016700 000124      FLTSG: MOV    SOBDBL,R0    ;SET WORD LENGTH LOOP
9904 060546 012702 001302      MOV    #2,$TMP0,R2     ;GET ADDRESS TO STORE WORDS IN
9905 060552 012701 000002      2$:  MOV    #2,R1         ;SET NUMBER OF WORDS TO 2
9906 060556 004767 177654      1$:  JSR    PC,$RAND      ;GET RANDOM NUMBER
9907 060562 022701 000002      CMP    #2,R1          ;FIRST TIME?
9908 060566 001404      BEQ    3$            ;BRANCH IF YES
9909 060570 022767 000002 000074  CMP    #2,SOBDBL      ;DOUBLE PRECISION?
9910 060576 001407      BEQ    4$            ;BRANCH IF YES
9911 060600 016703 120746      3$:  MOV    $HINUM,R3     ;GET EXPONENT PART
9912 060604 042703 000177      BIC    #177,R3        ;CHECK FOR MINUS ZERO
9913 060610 022703 100000      CMP    #BIT15,R3     ;
9914 060614 001760      BEQ    1$            ;BRANCH IF MINUS ZERO
9915 060616 016722 120730      4$:  MOV    $HINUM,(R2)+  ;SAVE HINUM
9916 060622 016722 120722      MOV    $LONUM,(R2)+  ;SAVE LONUM
9917 060626 077125      SOB    R1,1$         ;CONTINUE
9918 060630 077030      SOB    R0,2$         ;CONTINUE FOR DOUBLE PREC
9919 060632 012746 001302      MOV    #2,$TMP0,-(SP) ;PUT ADDRESS OF NUMBER ON STACK
9920 060636 012746 001002      MOV    #1002,-(SP)   ;PUT CONTROL WORD ON STACK
9921 060642 022767 000002 000022  CMP    #2,SOBDBL      ;DOUBLE PREC?
9922 060650 001002      BNE    5$            ;BRANCH IF NO
9923 060652 012716 001004      MOV    #1004,(SP)    ;CHANGE CONTROL WORD
9924 060656 004767 000012      5$:  JSR    PC,EXPEXT     ;CALCULATE EXT EXPONENTS
9925 060662 012767 000001 000002  MOV    #1,SOBDBL     ;INIT SOBDBL FOR SINGLE PREC
9926 060670 000207      RTS    PC           ;RETURN
9927 060672 000001      SOBDBL: .WORD 1
9928
9929      ;*****
9930      ;SBTTL FLOATING POINT EXPONENT EXTENSION
9931      ;*
9932      ;* THIS ROUTINE CONVERTS THE ACTUAL EXPONENT OF A FLOATING POINT
9933      ;* NUMBER INTO AN ACTUAL EXPONENT OF 200 AND AN EXTENDED
9934      ;* EXPONENT EQUAL TO THE DIFFERENCE BETWEEN THE ORIGINAL
9935      ;* ACTUAL EXPONENT AND 200.
9936      ;*
9937      ;* THE ROUTINE IS ENTERED WITH A CONTROL WORD ON THE STACK.
9938      ;* BIT 15 OF THE CONTROL WORD INDICATES WHETHER THE NUMBER
9939      ;* IS IN MEMORY (<15> 0) OR IN AN ACCUMULATOR (<15>=1).

```

9940
9941
9942
9943
9944
9945
9946 060674 012605
9947 060676 012600
9948 060700 100437
9949 060702 012601
9950 060704 162700 000400
9951 060710 012702 001302
9952 060714 160102
9953 060716 005402
9954 060720 006202
9955 060722 062702 001256
9956 060726 011103
9957 060730 042703 100177
9958 060734 072327 177771
9959 060740 162703 000200
9960 060744 010312
9961 060746 042711 077600
9962 060752 052711 040000
9963 060756 162700 000400
9964 060762 100435
9965 060764 110003
9966 060766 006303
9967 060770 060301
9968 060772 062702 000002
9969 060776 000753
9970
9971 061000 072027 177776
9972 061004 042700 177477
9973 061010 010002
9974 061012 072227 177773
9975 061016 062702 001412
9976 061022 042767 000300 000004
9977 061030 050067 000000
9978 061034 175003
9979 061036 060312
9980 061040 005003
9981 061042 042767 000300 000004
9982 061050 050067 000000
9983 061054 176403
9984 061056 010546
9985 061060 000207
9986
9987
9988
9989
9990
9991 061062 012737 061210 000024
9992 061070 012737 000340 000026
9993 061076 010046
9994 061100 010146
9995 061102 010246

```
;; BITS <9:8> INDICATE THE NUMBER OF NUMBERS TO CONVERT AND  
;; BITS <2:0> INDICATE THE WORD LENGTH OF THE NUMBER(S).  
;; IN THE CASE OF A MEMORY CONVERSION, THE ADDRESS OF THE  
;; FIRST WORD TO CONVERT IS ALSO ON THE STACK (PRECEDING  
;; THE CONTROL WORD).  
;*****  
EXPEXT: MOV (SP)+,R5 ;SAVE RETURN PC  
MOV (SP)+,R0 ;GET CONTROL WORD  
BMI 1$ ;BRANCH IF ACC CONVERSION  
MOV (SP)+,R1 ;GET START ADDRESS  
SUB #400,R0  
MOV #STMP0,R2 ;GET OFFSET FROM $TMP0  
SUB R1,R2  
NEG R2  
ASR R2  
ADD #SREG0,R2 ;GEN ADDRESS OF EXT WORD  
3$: MOV (R1),R3 ;GET DATA  
BIC #100177,R3 ;GET EXPONENT  
ASH #-7,R3 ;RIGHT JUSTIFY EXPONENT  
SUB #200,R3 ;CONVERT TO 2'S COMPLIMENT  
MOV R3,(R2) ;ADD TO EXTENDED EXPONENT  
BIC #77600,(R1) ;MAKE ACTUAL  
BIS #BIT14,(R1) ;EXPONENT 200  
SUB #400,R0 ;ANY MORE WORDS?  
BMI 2$ ;BRANCH IF NO  
MOVB R0,R3 ;GET WORD LENGTH  
ASL R3  
ADD R3,R1 ;SELECT NEXT NUMBER ADDRESS  
ADD #2,R2 ;SELECT NEXT EXTENDED ADDRESS  
BR 3$ ;CONTINUE  
;ACCUMULATOR CONVERSION  
1$: ASH #-2,R0 ;GET ACCUMULATOR NUMBER  
BIC #177477,R0  
MOV R0,R2 ;GENERATE  
ASH #-5,R2 ;ADDRESS OF  
ADD #SAC0,R2 ;EXTENDED EXPONENT  
BIC #300,5$ ;GENERATE INSTRUCTION  
BIS R0,5$ ;TO GET EXPONENT  
5$: STEXP AC0,R3 ;GET EXPONENT  
ADD R3,(R2) ;ADD TO EXTENDED EXPONENT  
CLR R3  
BIC #300,4$ ;GENERATE INSTRUCTION  
BIS R0,4$ ;TO LOAD EXPONENT BACK TO ACC  
4$: LDEXP R3,AC0 ;LOAD EXPONENT OF 200  
2$: MOV R5,-(SP) ;RESTORE RETURN PC  
RTS PC ;RETURN  
;*****  
;SBTTL POWER DOWN AND UP ROUTINES  
;POWER DOWN ROUTINE  
$PWRDN: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST UP  
MOV #340,@#PWRVEC+2 ;;PRIO:7  
MOV R0,-(SP) ;;PUSH R0 ON STACK  
MOV R1,-(SP) ;;PUSH R1 ON STACK  
MOV R2,-(SP) ;;PUSH R2 ON STACK
```



```

10052
10053 061262 010346          $RDLIN: MOV      R3,-(SP)          ;;SAVE R3
10054 061264 012703 061370  1$:  MOV      #$TTYIN,R3          ;;GET ADDRESS
10055 061270 022703 061400  2$:  CMP      #$TTYIN+8.,R3        ;;BUFFER FULL?
10056 061274 101405          BLOS     4$                      ;;BR IF YES
10057 061276 104412          RDCHR                    ;;GO READ ONE CHARACTER FROM THE TTY
10058 061300 112613          MOVB    (SP)+,(R3)            ;;GET CHARACTER
10059 061302 122713 000177  10$: CMPB    #177,(R3)           ;;IS IT A RUBOUT
10060 061306 001003          BNE     3$                      ;;SKIP IF NOT
10061 061310 104400 001336  4$:  TYPE    ,SQUES              ;;TYPE A '?'
10062 061314 000763          BR      1$                      ;;CLEAR THE BUFFER AND LOOP
10063 061316 111367 000044  3$:  MOVB    (R3),9$              ;;ECHO THE CHARACTER
10064 061322 104400 061366          TYPE    ,9$
10065 061326 122723 000015          CMPB    #15,(R3)+            ;;CHECK FOR RETURN
10066 061332 001356          BNE     2$                      ;;LOOP IF NOT RETURN
10067 061334 105063 177777          CLRB    -1(R3)                ;;CLEAR RETURN (THE 15)
10068 061340 104400 001340          TYPE    ,SLF                  ;;TYPE A LINE FEED
10069 061344 012603          MOV     (SP)+,R3              ;;RESTORE R3
10070 061346 011646          MOV     (SP),-(SP)            ;;ADJUST THE STACK AND PUT ADDRESS OF THE
10071 061350 016666 000004 000002  MOV     4(SP),2(SP)           ;;FIRST ASCII CHARACTER ON IT
10072 061354 012766 061370 000004  MOV     #$TTYIN,4(SP)
10073 061364 000002          RTI                          ;;RETURN
10074 061366 000          9$:  .BYTE   0                      ;;STORAGE FOR ASCII CHAR. TO TYPE
10075 061367 000          .BYTE   0                      ;;TERMINATOR
10076 061370 000010  $TTYIN: .BLKB  8.                ;;RESERVE 8 BYTES FOR TTY INPUT
10077
10078
10079          .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
10080
10081          ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
10082          ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
10083          ;*ARE READ A '?' FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
10084          ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
10085          ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
10086          ;*POSITIVE 32767 TO NEGATIVE 32768.
10087          ;*CALL:
10088          ;*      RDDEC                      ;;READ A DECIMAL NUMBER
10089          ;*      RETURN HERE                ;;NUMBER IS ON TOP OF THE STACK
10090
10091
10092 061400 011646          $RDDEC: MOV     (SP),-(SP)        ;;PROVIDE SPACE FOR
10093 061402 016666 000004 000002  MOV     4(SP),2(SP)            ;;THE INPUT NUMBER
10094 061410 010046          MOV     R0,-(SP)              ;;PUSH R0 ON STACK
10095 061412 010146          MOV     R1,-(SP)              ;;PUSH R1 ON STACK
10096 061414 010246          MOV     R2,-(SP)              ;;PUSH R2 ON STACK
10097 061416 104414          1$:  RDLIN                    ;;READ AN ASCII LINE
10098 061420 012600          MOV     (SP)+,R0              ;;ADDRESS OF 1ST CHAR.
10099 061422 010067 000120          MOV     R0,6$                 ;;SAVE INCASE OF BAD INPUT
10100 061426 005046          CLR     -(SP)                  ;;CLEAR DATA WORD
10101 061430 005002          CLR     R2                      ;;SIGN SET POSITIVE
10102 061432 122710 000055          CMPB    #'-',(R0)             ;;SEE IF A MINUS SIGN WAS TYPED
10103 061436 001001          BNE     2$                      ;;BR IF NO MINUS SIGN
10104 061440 112002          MOVB    (R0)+,R2              ;;SAVE FOR LATER USE
10105 061442 112001          2$:  MOVB    (R0)+,R1              ;;PICKUP THIS CHARACTER
10106 061444 001424          BEQ     3$                      ;;GET OUT IF ZERO
10107 061446 122701 000060          CMPB    #'0,R1                ;;MAKE SURE THIS CHARACTER
    
```

```
10108 061452 003032          BGT      5$          ;; IS A DIGIT BETWEEN 0 & 9
10109 061454 122701 000071  CMPB    #'9,R1
10110 061460 002427          BLT      5$
10111 061462 032716 170000  BIT     #'^(7777,(SP) ;; DON'T LET NUMBER GET TO BIG
10112 061466 001024          BNE     5$          ;; BR IF NUMBER WOULD OVERFLOW
10113 061470 006316          ASL     (SP)        ;; *2
10114 061472 011646          MOV     (SP),-(SP)  ;; SAVE FOR LATER
10115 061474 006316          ASL     (SP)        ;; *4
10116 061476 006316          ASL     (SP)        ;; *8
10117 061500 062616          ADD     (SP)+,(SP) ;; *10
10118 061502 102416          BVS     5$          ;; OVERFLOW ISN'T ALLOWED
10119 061504 162701 000060  SUB     #'0,R1      ;; STRIP AWAY THE ASCII JUNK
10120 061510 060116          ADD     R1,(SP)    ;; ADD IN THIS DIGIT
10121 061512 102412          BVS     5$          ;; OVERFLOW ISN'T ALLOWED
10122 061514 000752          BR      2$          ;; LOOP
10123 061516 005702          3$:    TST     R2      ;; CHECK IF NUMBER IS NEG
10124 061520 001401          BEQ     4$          ;; BR IF NO
10125 061522 005416          NEG     (SP)        ;; YES--NEGATE THE NUMBER
10126 061524 012666 000012  4$:    MOV     (SP)+,12(SP) ;; SAVE THE RESULT
10127 061530 012602          MOV     (SP)+,R2   ;; POP STACK INTO R2
10128 061532 012601          MOV     (SP)+,R1   ;; POP STACK INTO R1
10129 061534 012600          MOV     (SP)+,R0   ;; POP STACK INTO R0
10130 061536 000002          RTI
10131
10132 061540 005726          5$:    TST     (SP)+    ;; CLEAN PARTIAL NUMBER FROM STACK
10133 061542 105010          CLRB   (R0)        ;; SET A TERMINATOR
10134 061544 104400          TYPE   ;; TYPE THE INPUT UP TO BAD CHAR.
10135 061546 000000          6$:    .WORD  0        ;; POINTER GOES HERE
10136 061550 104400 001336  TYPE   , $QUES     ;; '?' 'CR' & 'LF'
10137 061554 000720          BR      1$          ;; TRY AGAIN
10138
10139
10140          .SBTTL  ROUTINE TO SIZE MEMORY
10141
10142          ;*CALL:
10143          ;*      JSR      PL,$SIZE
10144          ;*      RETURN
10145          ;*$LSTAD WILL CONTAIN:
10146          ;*      WITH KT11 OPTION      -- LAST VIRTUAL ADDRESS OF THE LAST BANK
10147          ;*      WITHOUT KT11 OPTION   -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
10148          ;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
10149          ;*$KT11 IS THE MEMORY MANAGEMENT KEY
10150          ;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
10151          ;*      MUST BE SETUP BEFORE THE CALL
10152          ;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
10153          ;*      DETERMINED BY ROUTINE
10154          ;*  --NOTE--
10155          ;*THIS ROUTINE SUPPORTS PDP 11/74.
10156          ;*IF ACTUAL MEMORY IS LESS THAN THAT INDICATED BY SIZE REGISTER
10157          ;*AND A REFERENCE IS MADE TO A MEMORY ADDRESS THAT IS GREATER THAN
10158          ;*ACTUAL MEMORY BUT LESS THAN SIZE REGISTER (INDICATED), THEN A
10159          ;*MEMORY REFERENCE TIMEOUT TO VECTOR 114 WILL OCCUR.
10160
10161 061556 010046          $SIZE: MOV     R0,-(SP)  ;; SAVE R0 ON THE STACK
10162 061560 010146          MOV     R1,-(SP)  ;; SAVE R1 ON THE STACK
10163 061562 010246          MOV     R2,-(SP)  ;; SAVE R2 ON THE STACK
```

10164	061564	010346				MOV	R3,-(SP)	::SAVE R3 ON THE STACK
10165	061566	013746	000004			MOV	@ERRVEC,-(SP)	::SAVE PRESENT ERROR VECTOR PS & P
10166	061572	013746	000006			MOV	@ERRVEC+2,-(SP)	::SAVE PRESENT PARITY VECTOR PS & P
10167	061576	013746	000114			MOV	@114,-(SP)	::SAVE PRESENT PARITY VECTOR PS & P
10168	061602	013746	000116			MOV	@116,-(SP)	::SAVE PRESENT PARITY VECTOR PS & P
10169	061606	010600				MOV	SP,R0	::SAVE THE STACK POINTER
10170	061610	013737	177776	000006		MOV	@PS,@ERRVEC+2	::SET ERRVEC PS TO PRESENT PS
10171	061616	012701	003776			MOV	#3776,R1	::SETUP ADDRESS
10172	061622	105727				TSTB	(PC)+	::USE MEMORY MANAGEMENT?
10173	061624	000200			\$K*11:	.WORD	200	::SET TO USE MEMORY MANAGEMENT
10174	061626	100065				BPL	SCORE	::BR IF NO
10175	061630	012737	061774	000004		MOV	#SKTNEX,@ERRVEC	::SET FOR TIMEOUT
10176	061636	005737	177572			TST	@SR0	::KT11 ARE YOU THERE?
10177	061642	052767	100000	177754		BIS	#100000,\$K*11	::YES--SET KT11 KEY
10178	061650	005046				CLR	-(SP)	::INITIALIZE FOR 'PAR' LOADING
10179	061652	012702	172340			MOV	#KIPAR0,R2	::ADDRESS OF FIRST 'PAR'
10180	061656	012703	000010			MOV	#^D8,R3	::LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
10181	061662	012762	077406	177740	1\$:	MOV	#77406,-40(R2)	::PDR = 4K, UP, READ/WRITE
10182	061670	011622				MOV	(SP),(R2)+	::LOAD 'PAR'
10183	061672	062716	000200			ADD	#200,(SP)	::UPDATE FOR NEXT 'PAR'
10184	061676	077307				SQB	R3,1\$::LOOP UNTIL ALL EIGHT ARE LOADED
10185	061700	012742	177600			MOV	#177600,-(R2)	::SETUP KIPAR7 FOR I/O
10186	061704	005042				CLR	-(R2)	::SETUP KIPAR6 FOR TESTING
10187	061706	012737	061724	000004		MOV	#2\$,@ERRVEC	::CATCH TIMEOUT IF NO SR3
10188	061714	012737	000020	172516		MOV	#20,@SR3	::ENABLE 22-BIT ADDRESSING
10189	061722	000401				BR	3\$::THIS PDP-11 HAS A SR3 RFG.
10190	061724	022626			2\$:	CMP	(SP)+,(SP)+	::CLEAN OFF THE STACK--NO SR3.
10191	061726	005237	177572		3\$:	INC	@SR0	::TURN ON MEMORY MANAGEMENT
10192	061732	012737	061764	000004		MOV	#SKTOUT,@ERRVEC	::SET FOR TIME OUT
10193	061740	012737	062106	000114		MOV	#SMTOUT,@114	::SET FOR MEMORY REF TIMEOUT TO 114
10194	061746	005737	143776		4\$:	TST	@143776	::TRAP ON NON-EX-MEM
10195	061752	062712	000040			ADD	#40,(R2)	::MAKE A 1K STEP
10196	061756	023712	172356			CMP	@KIPAR7,(R2)	::LAST ONE?
10197	061762	101371				BHI	4\$::NO--TRY IT
10198	061764	011202			\$KTOUT:	MOV	(R2),R2	::GET LAST BANK+1
10199	061766	005037	177572			CLR	@SR0	::TURN OFF MEMORY MANAGEMENT
10200	061772	000421				BR	\$SIZEX	
10201	061774	042767	100000	177622	\$KTNEX:	BIC	#100000,\$KT11	::KT11 NON-EXISTENT
10202	062002	012737	062032	000004	\$SCORE:	MOV	#SCROUT,@ERRVEC	::SET FOR TIMEOUT
10203	062010	005002				CLR	R2	::SET UP BANK
10204	062012	062701	004000		1\$:	ADD	#4000,R1	::INCREMENT BY 1K
10205	062016	062702	000040			ADD	#40,R2	::1K STEP
10206	062022	005711				TST	(R1)	::TRAP ON TIME OUT
10207	062024	022701	177776			CMP	#177776,R1	::LAST ONE
10208	062030	001370				BNE	1\$::NO--TRY AGAIN
10209	062032	162701	004000		\$CROUT:	SUB	#4000,R1	
10210	062036	162702	000040		\$SIZEX:	SUB	#40,R2	::DROP BACK
10211	062042	010006				MOV	R0,SP	::RESTORE THE STACK
10212	062044	012637	000116			MOV	(SP)+,@116	::RESTORE PARITY VECTOR
10213	062050	012637	000114			MOV	(SP)+,@114	
10214	062054	012637	000006			MOV	(SP)+,@ERRVEC+2	::RESTORE ERROR VECTOR
10215	062060	012637	000004			MOV	(SP)+,@ERRVEC	
10216	062064	010167	000050			MOV	R1,\$LSTAD	::LAST ADDRESS
10217	062070	010267	000046			MOV	R2,\$LSTBK	::LAST BANK
10218	062074	012603				MOV	(SP)+,R3	::RESTORE R3
10219	062076	012602				MOV	(SP)+,R2	::RESTORE R2

```
10220 062100 012601      MOV      (SP)+,R1      ;;RESTORE R1
10221 062102 012600      MOV      (SP)+,R0      ;;RESTORE R0
10222 062104 000207      RTS      PL
10223 062106 032737 000001 177744 $MTMOUT: BIT  #BIT0,@MEMERR ;;MAKE SURE TRAP TO 114 IS
10224 062114 001005      BNE      1$           ;;DUE TO MEMORY REF TIMEOUT
10225                                     ;;IF NOT, IS IT AN ABORT?
10226 062116 032737 100000 177744      BIT  #BIT15,@MEMERR ;;CPU ABORT?
10227 062124 001001      BNE      1$           ;;IF YES, EXIT
10228 062126 000002      RTI
10229 062130 012737 177777 177744 1$:  MOV      #-1,@MEMERR ;;CLEAR THE MEM ERROR REG
10230 062136 000712      BR       $KTOUT
10231 062140 000000      $LSTAD: .WORD 0      ;;CONTAINS THE LAST ADDRESS
10232 062142 000000      $LSTBK: .WORD 0      ;;CONTAINS THE LAST BANK
10233                                     ;;*****
10234                                     .SBTTL TRAP DECODER
10235                                     ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
10236                                     ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
10237                                     ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
10238                                     ;*GO TO THAT ROUTINE.
10239                                     $TRAP: MOV      R0,-(SP)      ;;SAVE R0
10240                                     MOV      2(SP),R0      ;;GET TRAP ADDRESS
10241                                     TST      -(R0)        ;;BACKUP BY 2
10242 062144 010046 000002      MOV#B   (R0),R0      ;;GET RIGHT BYTE OF TRAP
10243 062146 016600      MOV      $TRPAD(R0),R0 ;;INDEX TO TABLE
10244 062152 005740      RTS      R0          ;;GO TO ROUTINE
10245 062154 111000
10246 062156 016000 062164
10247 062162 000200
10248
10249                                     .SBTTL TRAP TABLE
10250                                     ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
10251                                     ;*BY THE 'TRAP' INSTRUCTION.
10252                                     :          ROUTINE
10253                                     :          -----
10254                                     $TRPAD:
10255                                     $TYPE   ;;CALL=TYPE      TRAP+0(104400) TTY TYPEOUT ROUTINE
10256 062164 056264      $TYPOC  ;;CALL=TYPOC   TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
10257 062166 057116      $TYPOS  ;;CALL=TYPOS   TRAP+4(104404) TYPE OCTAL NUMBER (NO LEADING ZEROS)
10258 062170 057072      $TYPON  ;;CALL=TYPON   TRAP+6(104406) TYPE OCTAL NUMBER (AS PER LAST CALL)
10259 062172 057132      $TYPDS  ;;CALL=TYPDS   TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
10260 062174 057320      $RDCHR  ;;CALL=RDCHR   TRAP+12(104412) TTY TYPEIN CHARACTER ROUTINE
10261 062176 061226      $RDLIN  ;;CALL=RDLIN   TRAP+14(104414) TTY TYPEIN STRING ROUTINE
10262 062200 061262      $RDDEC  ;;CALL=RDDEC   TRAP+16(104416) READ A DECIMAL NUMBER FROM TTY
10263 062202 061400      $SAVREG ;;CALL=SAVREG   TRAP+20(104420) SAVE R0-R5 ROUTINE
10264 062204 057702      $RESREG ;;CALL=RESREG   TRAP+22(104422) RESTORE R0-R5 ROUTINE
10265 062206 057740      $FL20   ;;CALL=FL20   TRAP+24(104424)
10266 062210 057776      $FLD20  ;;CALL=FLD20   TRAP+26(104426)
10267 062212 060264
10268
10269                                     ;;*****
10270                                     .SBTTL UNIBUS EXERCISER INITIALIZATION ROUTINE
10271                                     ;*THIS ROUTINE INITIALIZES THE BASE ADDRESS FOR THE
10272                                     ;*UNIBUS EXERCISER AND LOADS UP THE EXERCISER REGISTERS.
10273                                     ;;*****
10274 062214 012767 002344 117500 UBEINIT:MOV  #SERRTB,UBESAV ;BASE ADDRESS OF UBE TRANSFER
```



```
10276 062222 005067 117476          CLR    UBESAV+2
10277 062226 012767 002344 117472    MOV    #SERRTB,UBEADR          ;BASE ADDRESS OF UBE TRANSFER
10278 062234 005067 117470          CLR    UBEADR+2
10279
10280
10281          ;SET UP THE UBE AND START IT
10282 062240 012702 002270          MOV    #UBETBL,R2             ;GET ADDRESS OF UBE TABLE
10283 062244 005072 000010          CLR    @10(R2)               ;CLEAR ALL ERRORS
10284 062250 012772 053352 000012    MOV    #UBESRV,@12(R2)       ;SET UP UBE VECTOR
10285 062256 012772 000340 000014    MOV    #PR7,@14(R2)          ;SET UP UBE VECTOR PSW
10286 062264 012732 172400          MOV    #172400,@(R2)+        ;SET CC FOR 1.3K WORD TRANSFER
10287          ;UBE IS DOING BYTE TRANSFERS
10288 062270 012746 000003          MOV    #3,-(SP)              ;PUT DEVICE ID IN STACK
10289 062274 012746 001726          MOV    #UBEADR,-(SP)         ;PUT ADDRESS OF PHYSICAL BA ON STACK
10290 062300 004737 062676          JSR    PC,@GETMAP            ;GO GET MAP REGISTER
10291 062304 013732 001726          MOV    @UBEADR,@(R2)+        ;LOAD UBE BUS ADDRESS
10292 062310 013732 001730          MOV    @UBEADR+2,@(R2)+      ;LOAD ADR BITS 16 & 17
10293 062314 052737 000040 172516    BIS    #40,@SR3              ;ENABLE MAP
10294 062322 000207          RTS    PC                    ;RETURN
10295          ;*****
10296          ;SBTTL CONVERT UNIBUS VIRTUAL ADDRESS TO PHYSICAL ADDRESS
10297          ;* THIS ROUTINE CONVERTS THE CONTENTS OF LOCATIONS
10298          ;* 'ERRBA' AND 'ERRBA+2' FROM A VIRTUAL 18-BIT ADDRESS
10299          ;* TO A PHYSICAL 22-BIT ADDRESS AS MAPPED BY THE APPROPRIATE
10300          ;* MAP REGISTER. THE 22-BIT ADDRESS IS STORED IN LOCATIONS
10301          ;* 'PA2116' AND 'PA1500'.
10302          ;*****
10303          PHYMAP: SAVREG
10304 062324 104420          MOV    @ERRBA,R3             ;GET BUS ADDRESS <15:00>
10305 062326 013703 001732          MOV    @ERRBA+2,R2          ;GET BUS ADDRESS <17:16>
10306 062332 013702 001734          BIC    #177774,R2           ;
10307 062336 042702 177774          BIT    #BIT5,@MMR3          ;MAP ON?
10308 062342 032737 000040 172516    BNE    1$                   ;BRANCH IF YES
10309 062350 001005          MOV    R3,@PA1500           ;PHY ADR=BUS ADR
10310 062352 010337 001524          MOV    R2,@PA2116
10311 062356 010237 001526          BR     MAPEND
10312 062362 000421          1$: MOV    R3,R5               ;SAVE ADR BITS <15:00>
10313 062364 010305          ASHC   #5,R2                ;GET MAP REG SELECT BITS
10314 062366 073227 000005          BIC    #3,R2                ;
10315 062372 042702 000003          ADD    #MAPLO,R2            ;FORM ADDRESS OF MAP REG
10316 062376 062702 170200          MOV    (R2)+,@PA1500        ;GET CONTENTS OF MAP REG LO
10317 062402 012237 001524          MOV    (R2),@PA2116         ;GET CONTENTS OF MAP REG HI
10318 062406 011237 001526          BIC    #160000,R5           ;FORM PHYSICAL ADDRESS
10319 062412 042705 160000          ADD    R5,@PA1500           ;THAT TIMED OUT
10320 062416 060537 001524          ADC    @PA2116              ;
10321 062422 005537 001526          MAPEND: RESREG
10322 062426 104422          RTS    PC
10323          ;*****
10324          ;SBTTL CONVERT A VIRTUAL ADDRESS TO A PHYSICAL ADDRESS
10325          ;* THIS ROUTINE CONVERTS A 16-BIT VIRTUAL ADDRESS TO A
10326          ;* 22-BIT PHYSICAL ADDRESS. THE VIRTUAL ADDRESS IS
10327          ;* ASSUMED TO BE IN LOCATION 'VADR' AND THE PHYSICAL
10328          ;* ADDRESS IS PLACED IN LOCATIONS 'PA2116' AND 'PA1500'.
10329          ;*
10330          ;* IF MEMORY MANAGEMENT IS OFF THE PHYSICAL ADDRESS IS
10331          ;*
```

10332
10333
10334
10335
10336
10337
10338
10339
10340
10341 062432 104420
10342 062434 013703 001522
10343 062440 105737 001531
10344 062444 001426
10345 062446 005002
10346 062450 073227 000003
10347 062454 072327 177775
10348 062460 042703 160000
10349 062464 006102
10350 062466 062702 172340
10351 062472 011205
10352 062474 005004
10353 062476 073427 000006
10354 062502 060305
10355 062504 005504
10356 062506 010437 001526
10357 062512 010537 001524
10358 062516 104422
10359 062520 000207
10360 062522 163703 001534
10361 062526 005004
10362 062530 010305
10363 062532 000765
10364
10365
10366
10367
10368
10369
10370
10371
10372
10373
10374
10375
10376
10377 062534 012703 004000
10378 062540 012737 062632 000114
10379 062546 024042
10380 062550 001026
10381 062552 005112
10382 062554 005112
10383 062556 021210
10384 062560 001022
10385 062562 020005
10386 062564 001414
10387 062566 105737 001531

```

: * 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 @#VADR,R3 ;GET VIRTUAL ADDRESS TO CONVERT
TSTB @#MMON ;IS MEMCRY MGMT ON?
BEQ 1$ ;BRANCH IF NO
CLR R2
ASHC #3,R2 ;GET PAR SELECT BITS
ASH #-3,R3 ;RETURN VIR ADDR TO ORIGINAL
BIC #160000,R3 ;MAKE SURE SIGN DIDN'T EXTEND
ROL R2 ;MAKE R2 EVEN FOR WORD ADDRESSING
ADD #KIPAR0,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$: MOV R4,@#PA21'6 ;SAVE PHYSICAL
MOV R5,@#PA1500 ;ADDRESS
RESREG
RTS PC ;RETURN
1$: SUB @#FACTOR,R3 ;FORM PHYSICAL ADDRESS
CLR R4
MOV R3,R5
BR 2$ ;RETURN
: *****
: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 #2048,R3 ;COUNTER
MOV #2$,@#CACHVEC ;SETUP PARITY VECTOR
7$: CMP -(R0),-(R2) ;CHECK DATA
BNE 99$
COM (R2) ;COMPLEMENT DEST DATA
COM (R2) ;TWICE
CMP (R2),(R0) ;CHECK DATA
BNE 99$
1$: CMP R0,R5 ;BRANCH IF ALL DATA CHECKED
BEQ 3$
TSTB @#MMON ;MEMORY MANAGMENT ON?

```

10388 062572 001765
10389 062574 077314
10390 062576 012703 010000
10391 062602 162737 000200 172350
10392 062610 012702 120000
10393 062614 000754
10394 062616 012737 063464 000114 3\$:
10395 062624 000207
10396 062626 000262 99\$:
10397 062630 000207
10398 062632 013737 177744 001306 2\$:
10399 062640 013737 177740 001310
10400 062646 013737 177742 001312
10401 062654 010237 001522
10402 062660 010037 001302
10403 062664 104005
10404 062666 012737 177777 177744
10405 062674 000754

```

BEQ 7$ ;BR BACK IF NOT
SOB R3,7$ ;REPEAT 4096 TIMES
MOV #4096.,R3 ;SET UP COUNTER AGAIN
SUB #200,@#KIPAR4 ;MAP TO NEXT LOWER 4K OF SPACE
MOV #120000,R2 ;START AT TOP OF 4K SPACE + 2
BR 7$ ;CHECK MORE
MOV #.PARSRV,@#CACHVEC ;RESTORE CACHVEC
RTS PC ;RETURN

99$: SEV
RTS PC

2$: MOV @#MEMFRR,@#STMP2 ;SAVE ERROR REG
MOV @#LOADRS,@#STMP3 ;SAVE ERROR ADR
MOV @#HIADRS,@#STMP4
MOV R2,@#VADR
MOV R0,@#STMP0
ERROR 5
MOV #-1,@#MEMERR ;CLEAR ERROR REG
BR 99$ ;RETURN
    
```

10406
10407
10408
10409
10410
10411
10412
10413
10414
10415
10416
10417
10418
10419
10420
10421
10422
10423
10424
10425
10426
10427
10428
10429

```

*****
SRTTL 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
*REGISTERS WITH THE PHYSICAL ADDRESS MINUS THE RANDOM NUMBER
*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
*****
    
```

10430 062676 016600 000004
10431 062702 016601 000002
10432 062706 013746 177776
10433 062712 005116
10434 062714 042716 177437
10435 062720 000237
10436 062722 104420
10437 062724 012137 001226
10438 062730 012137 001230
10439 062734 004737 060436 2\$:
10440 062740 013702 001552
10441 062744 013703 001550
10442 062750 073227 177764
10443 062754 042702 177760

```

*****
GETMAP: MOV 4(SP),R0 ;GET DIVICE ID
MOV 2(SP),R1 ;GET ADR OF PHY ADR
MOV @#PSW,-(SP) ;SAVE CURRENT PRIORITY
COM (SP) ;MAKE IT READY FOR RESTORE
BIC #^CPR7,(SP)
SPL 7 ;IF THIS IS RK CALL, LOCK OUT UBE
SAVREG
MOV (R1)+,@#SGDDAT ;SAVE PHYSICAL
MOV (R1)+,@#SBDDAT ;ADDRESS
JSR PC,@#SRAND ;GET RANDOM NUMBER
MOV @#SHNUM,R2 ;GET HIGH RANDOM NUMBER
MOV @#SLGNUM,R3 ;GET LOW RANDOM NUMBER
ASHC #-14,R2 ;CONVERT TO 20 BIT NUMBER
BIC #177760,R2 ;GET RID OF 11 BITS OF SIGN EXT
    
```

```
10444 062760 022702 000016      CMP      #16,R2      ;LEGAL MAP REG SELECT?
10445 062764 100001      BP_      3$          ;BRANCH IF YES
10446 062766 000762      BR       2$          ;TRY AGAIN
10447 062770 005737 001532      3$:  TST      @#QV      ;ACT11 (QV OR AUTO)?
10448 062774 001403      BEQ      4$          ;BRANCH IF NO
10449 062776 122702 000000      CMPB     #0,R2      ;MAP SELECT 0?
10450 063002 001754      BEQ      2$          ;BRANCH IF YES.(ACT MUST
10451                                ;USE THIS MAP REG)
10452 063004 010204      4$:  MOV      R2,R4      ;SAVE MAP SELECT BITS
10453 063006 042703 100000      BIC      #BIT15,R3   ;CLEAR SELECT BIT 0
10454 063012 073227 177776      ASHC     #-2,R2      ;FORM 18 BIT ADDRESS
10455 063016 042703 000001      BIC      #BIT0,R3    ;MAKE SURE ITS EVEN
10456 063022 010241      MOV      R2,-(R1)    ;RETURN NEW BUS ADDRESS
10457 063024 010341      MOV      R3,-(R1)    ;TO THE APPROPRIATE HANDLER
10458 063026 012705 000004      MOV      #4,R5       ;SET SOB COUNT
10459 063032 120465 001715      1$:  CMPB     R4,MAPTBL-1(R5) ;IS THIS MAP IN USE?
10460 063036 001435      BEQ      5$          ;BRANCH IF YES
10461 063040 077504      SOB      R5,1$       ;CONTINUE
10462 063042 110460 001716      MOVB     R4,MAPTBL(R0) ;PUT MAP SELECT BITS IN TABLE
10463 063046 072427 000003      ASH      #3,R4       ;FORM INDEX TO GET MAP REG ADDR
10464 063052 062704 170200      ADD      #MAPLO,R4    ;GENERATE MAP ADDRESS
10465 063056 042703 160000      BIC      #160000,R3   ;GET LS 13 BITS OF RAND NO.
10466 063062 013701 001226      MOV      @#SGDDAT,R1 ;GET PHYSICAL
10467 063066 013702 001230      MOV      @#SBDDAT,R2 ;ADDRESS
10468 063072 160301      SUB      R3,R1       ;GENERATE MAP
10469 063074 005602      SBC      R2          ;REGISTER DATA
10470 063076 010124      MOV      R1,(R4)+    ;LOAD THE
10471 063100 010224      MOV      R2,(R4)+    ;FIRST MAP REGISTER
10472 063102 062701 020000      ADD      #20000,R1   ;ADD 4K
10473 063106 005502      ADC      R2          ;TO MAP DATA
10474 063110 010124      MOV      R1,(R4)+    ;LOAD THE
10475 063112 010224      MOV      R2,(R4)+    ;SECOND MAP REGISTER
10476 063114 104422      RESREG
10477 063116 042637 177776      BIC      (SP)+,@#PSW ;RETURN PRIORITY TO ORIGINAL VALUE
10478 063122 011666 000004      MOV      (SP),4(SP)  ;SETUP RETURN PC
10479 063126 022626      CMP      (SP)+,(SP)+ ;CLEAN UP THE STACK
10480 063130 000207      RTS      PC          ;RETURN
10481                                ;REGISTER PAIR IS IN USE, TRY ANOTHER RANDOM NUMBER
10482 063132 062701 000004      5$:  ADD      #4,R1      ;RESTORE R1
10483 063136 000137 062734      JMP      @#2$        ;GET ANOTHER RANDOM NUMBER
10484                                ;*****
10485                                ;SBTTL GIVE MAP SUBROUTINE
10486                                ;* THIS ROUTINE TAKES THE MAP ADDRESS OUT OF THE MAP TABLE
10487                                ;* FOR THE REQUESTING DEVICE AND REPLACES IT WITH 377.
10488                                ;*****
10489 063142 010046      GIVEMAP:MOV R0,-(SP) ;SAVE R0
10490 063144 016600 000004      MOV      4(SP),R0    ;GET DEVICE ID
10491 063150 112760 000377 001716      MOVB     #377,MAPTBL(R0) ;TAKE IT OUT OF THE TABLE
10492 063156 012600      MOV      (SP)+,R0    ;RESTORE R0
10493 063160 000207      RTS      PC          ;RETURN
10494                                ;*****
10495                                ;SBTTL ROUTINE TO CLEAR '-' BIT
10496                                ;*****
10497                                ;*****
10498 063162 013746 177776      CLRTBIT:MOV @#PSW,-(SP) ;PUSH PSW ONTO STACK
10499 063166 011627      MOV      (SP),(PC)+ ;SAVE IN RETPSW BELOW
```

```
10500 063170 000000 RE*PSW: .WORD 0
10501 063172 042716 000020 BIC #20,(SP) ;CLEAR T BIT IN PSW ON STACK
10502 .....
10503 .SBTTL ROUTINE TO RESTORE THE T BIT
10504 .....
10505 063176 012746 063204 RESPSW: MOV #1$,-(SP) ;SET RETURN PC FOR RTI
10506 063202 000002 RTI ;CLEAR 'T' BIT IN PSW
10507 063204 000207 1$: RTS PC ;RETURN
10508
10509 063206 042737 177400 177776 RESTPS: BIC #177400,@#PSW ;SET KERNEL MGDE
10510 063214 016746 177750 MOV RETPSW,-(SP) ;PUSH ORIG PSW ONTO STACK
10511 063220 000766 BR RESPSW
10512 .....
10513 .SBTTL KEYBOARD INT SERV ROUTINE
10514 *THIS ROUTINE HANDLES INTERRUPTS FROM THE KEYBOARD
10515 *
10516 *TYPING A CONTROL 'C' WILL CAUSE THE PROCFOSSOR TO HALT
10517 *
10518 *TYPING A CARRAGE RETURN WILL CAUSE A CARRIAGE RETURN-LINE FEED
10519 *TO BE TYPED.
10520 *
10521 *TYPING A CONTROL 'O' WILL INHIBIT ANY FURTHER TYPEOUT. THE SECOND CONTROL 'O'
10522 *WILL ENABLE TYPEOUT AGAIN AND ECHO A CR-LF.
10523 *
10524 *ANY OTHER CHARACTER WILL JUST BE ECHOED.
10525 .....
10526
10527
10528 000003 CNTRLC=3
10529 000017 CNTRLO=17
10530
10531 063222 017746 116014 TKISR: MOV @STKB,-(SP) ;GET CHARACTER
10532 063226 042716 177600 BIC #177600,(SP) ;STRIP UNUSED BITS
10533 063232 022716 000003 CMP #CNTRLC,(SP) ;BRANCH IF NOT CONTROL C (^C)
10534 063236 001010 BNE 1$
10535 063240 012737 001336 001270 MOV #SCRLF-1,@#SREG5 ;ECHO CR LF
10536 063246 106277 115772 ASRB @STPS
10537 063252 005726 TST (SP)+ ;POP CHARACTER OFF THE STACK
10538 063254 000000 HALT
10539 063256 000002 RTI ;RETURN
10540
10541 063260 122716 000015 1$: CMPB #1$,(SP) ;BRANCH IF NOT <CR>
10542 063264 001007 BNE 2$
10543 063266 012737 001336 001270 MOV #SCRLF-1,@#SREG5 ;ECHO CR LF
10544 063274 106277 115744 ASRB @STPS
10545 063300 005726 TST (SP)+ ;POP CHARACTER OFF STACK
10546 063302 000002 RTI ;RETURN
10547
10548 063304 122716 000017 2$: CMPB #CNTRLO,(SP) ;BRANCH IF NOT CONTROL O (^O)
10549 063310 001012 BNE 3$
10550 063312 005726 TST (SP)+
10551 063314 005167 116156 COM NOTYPE
10552 063320 100405 BMI 7$
10553 063322 012737 001336 001270 MOV #SCRLF-1,@#SREG5 ;ECHO CR LF
10554 063330 106277 115710 ASRB @STPS
10555 063334 000002 7$: RTI
```

10556					
10557	063336	104420		3\$: SAVREG	
10558	063340	011605		MOV (SP),R5	;RETRIEVE CHARACTER
10559	063342	004737	063752	JSR PC,@#LDKT	;GO TO LOW CORE
10560	063346	013700	001452	MOV @#TKBFRP,R0	;GET BUFFER PTR
10561	063352	110520		4\$: MOVB R5,(R0)+	;LOAD CHAR INTO BFR
10562	063354	105010		CLRB (R0)	;CLEAR NEXT LOC
10563	063356	022700	001474	5\$: CMP #TKBFR+20,R0	;BRANCH IF NOT END OF BFR
10564	063362	001002		BNE 6\$	
10565	063364	012700	001454	MOV #TKBFR,R0	;RESET BUFFER PTR
10566	063370	010037	001452	6\$: MOV R0,@#TKBFRP	;RESTORE BFR PTR
10567	063374	004737	064050	JSR PC,@#RESKT	;GO BACK TO ORIGINAL MEMORY
10568	063400	104422		RESREG	
10569	063402	005737	001476	ECHO: TST @#NOTYPE	;TYPEOUT DISABLED?
10570	063406	100004		BPL 1\$;BRANCH IF NO
10571	063410	005726		TST (SP)+	;FIX UP STACK
10572	063412	105077	115626	CLRB @#STPS	;CLEAR IE BIT
10573	063416	000002		RTI	;RE JRN
10574	063420	105777	115620	1\$: TSTB @#STFS	;PRINTER READY?
10575	063424	100375		BPL -4	;BRANCH IF NO
10576	063426	112677	115614	MOVB (SP)+,@#STPB	;MOVE CHAR TO PRINTER
10577	063432	000002		RTI	;RETURN
10578					
10579					
10580					
10581					
10582					
10583					
10584					
10585	063434	005237	001270		
10586	063440	117746	115624		
10587	063444	001356			
10588	063446	005726			
10589	063450	005077	115570		
10590	063454	012737	001554		
10591	063462	000002	001270		
10592					
10593					
10594					
10595					
10596					
10597					
10598					
10599					
10600					
10601					
10602					
10603					
10604					
10605					
10606					
10607					
10608					
10609	063464	012737	063744		
10610	063472	016637	000002		
10611	063500	011637	001522		

:SBTTL TELETYPE INTERRUPT SERVICE ROUTINE
:*THIS ROUTINE TYPES A MESSAGE POINTED TO BY THE ADR STORED
:*IN LOCATION \$REG5. THIS ROUTINE IS INTERRUPT DRIVEN.

TPIISR: INC @#\$REG5 ;STEP MESSAGE ADDRESS PTR
MOVB @#\$REG5,-(SP) ;GET CHAR TO BE TYPED
BNE ECHO ;GO TYPE CHAR IF NOT '0'
TST (SP)+ ;POP STACK
CLR @#STPS ;CLEAR IE BIT
MOV #NULLS,@#\$REG5
RTI ;RETURN

:SBTTL PARITY ERROR SERVICE
:* THIS ROUTINE FIELDS UNEXPECTED TRAPS TO 114. IT IS ASSUMED
:* THAT THE ERROR WAS IN CACHE AND WAS CAUSED BY THE 'OTHER
:* WORD' RATHER THAN THE 'WANTED WORD' WHICH MEANS THAT THE
:* BAD DATA IS STILL IN THE CACHE. SO, TO CLEAR THE BAD DATA
:* THE ERROR ADDRESS IS REFERENCED CAUSING THE CACHE TO GO
:* TO MAIN MEMORY TO GET THE DATA. THIS PREVENTS AN
:* ARBITRARY REFERENCE TO THE BAD WORD FROM TRAPPING.

:* AFTER THE ERROR IS REPORTED, BITS 2 AND 3 OF THE MEMORY
:* ERROR REGISTER ARE TESTED TO SEE IF THE BAD DATA IS IN
:* MAIN MEMORY. IF IT IS, THE PROGRAM RESTARTS SINCE THE
:* GOOD DATA IS NOW LOST FOREVER. OTHERWISE THE PROGRAM
:* RETURNS TO THE ADDRESS POINTED TO BY '\$LPER?'.

PARSRV:MOV #RT1,@#CAHVEC ;PUT NEW ADDRESS IN PARITY VECTOR
MOV 2(SP),@#STMP0 ;SAVER ERROR PSW
MOV (SP),@#VADR ;SAVE PC

```
10612 063504 162737 000002 001522 SUB #2,@#VADR ;ADJUST ERROR PC
10613 063512 013702 177744 MOV @#MEMERR,R2 ;GET ERROR REGISTER
10614 063516 013703 177740 MOV @#LOADRS,R3 ;GET LO ADDRESS ERROR REG
10615 063522 010337 001310 MOV R3,@#STMP3 ;PUT LOW ADR IN MEMORY
10616 063526 013737 177742 001312 MOV @#HIADRS,@#STMP4 ;GET HI ADDRESS ERROR REG
10617 063534 042703 176000 BIC #176000,R3 ;MASK OFF LOWER TEN BITS
10618 063540 013704 172354 MOV @#KIPAR6,R4 ;SAVE PAR6
10619 063544 105737 001531 TSTB @#MMON ;IS MEMORY MGMT ON?
10620 063550 001407 BEQ 1$ ;BRANCH IF NO
10621 063552 005037 172354 CLR @#KIPAR6 ;CLEAR PAR6
10622 063556 012737 077406 172314 MOV #77406,@#KIPDR6 ;ENSURE PDR 6 RESIDENT
10623 063564 052703 140000 BIS #140000,R3 ;SETUP R3 TO REFERENCE THRU PAR6
10624 063570 105713 1$: TSTB (R3) ;REFERENCE ADDRESS THAT TRAPPED
10625 ;SHOULD CAUSE ABORT
10626 063572 005102 2$: COM R2 ;GET ORIGINAL MEMORY
10627 063574 010237 177744 MOV R2,@#MEMERR ;ERROR REG DATA
10628 063600 013737 177744 001306 PERET: MOV @#MEMERR,@#STMP2 ;SAVE ERROR REG FOR TYPEOUT
10629 063606 013737 001212 001266 MOV @#SLPERR,@#SREG4 ;SAVE LOOP ADDRESS
10630 063614 012737 063624 001212 MOV #2$,@#SLPERR ;SET RETURN ADDRESS IF LOOPING
10631 063622 104004 ERROR 4
10632 063624 013737 001266 001212 2$: MOV @#SREG4,@#SLPERR ;RESTORE LOOP ADDRESS
10633 063632 010437 172354 MOV R4,@#KIPAR6 ;RESTORE PAR6
10634 063636 013704 177744 MOV @#MEMERR,R4 ;GET MEM ERR REG
10635 063642 012737 177777 177744 MOV #-1,@#MEMERR ;CLEAR ERR REG
10636 063650 012737 063464 000114 MOV #.PARSRV,@#CACHVEC ;RESTORE PARITY VECTOR
10637 063656 042704 177763 BIC #177763,R4 ;CLEAR ALL BUT BITS 2 & 3
10638 063662 001426 BEQ 1$ ;BRANCH IF NOT MAIN MEMORY ERROR
10639 063664 104400 063672 TYPE ,65$ ;:TYPE ASCIZ STRING
10640 063670 000420 BR 64$ ;:GET OVER THE ASCIZ
10641 ;:65$: .ASCIZ /FATAL PARITY ERROR-RESTARTING/<CRLF>
10642 64$:
10643 063732 000005 RESET ;CLEAR THE WORLD
10644 063734 000137 003542 JMP @#START
10645 063740 012716 063746 1$: MOV #X,(SP) ;PUT ADDRESS ON STACK TO GET ORIGINAL
10646 ;PSW BACK
10647 063744 000002 RTI: RTI ;GET OLD PSW
10648 063746 000177 115240 X: JMP @#SLPERR ;JUMP TO START OF TEST THAT HAD THE PE
10649 ;*****
10650 ;SBTTL CONTEXT SWITCH DOWN SUBROUTINE
10651 ;* SUBROUTINE TO SAVE & LOAD KIPAR'S 0,1,2 AND 3 (IF MEM MGMT ENABLED)
10652 ;* THIS ROUTINE IS CALLED BY THE KEYBOARD INTERRUPT, LINE CLOCK
10653 ;* INTERRUPT, UBE SERVICE ROUTINE, MBT SERVICE ROUTINE, AND TYPE TIME ROUTINE.
10654 ;*****
10655 063752 105737 001531 LDKT: TSTB @#MMON ;BRANCH IF MEM MGMT DISABLED
10656 063756 001433 BEQ 1$
10657 063760 012604 MOV (SP)+,R4 ;SAVE RETURN PC
10658 063762 013737 177776 001516 MOV @#PSW,@#SSAVPSW ;SAVE THE CURRENT PSW
10659 063770 042737 140000 177776 BIC #140000,@#PSW ;GO TO KERNEL MODE
10660 063776 012700 172340 MOV #KIPAR0,R0 ;GET ADDRESS OF PAR0
10661 064002 012001 MOV (R0)+,R1 ;GET PAR0
10662 064004 012002 MOV (R0)+,R2 ;GET PAR1
10663 064006 012003 MOV (R0)+,R3 ;GET PAR2
10664 064010 012005 MOV (R0)+,R5 ;GET PAR3
10665 064012 012740 000600 MOV #600,-(R0) ;BACK TO LOW CORE
10666 064016 012740 000400 MOV #400,-(R0) ;RELOC BACK TO LOW CORE
10667 064022 012740 000200 MOV #200,-(R0)
```

```
10668 064026 005040          CLR      -(R0)
10669 064030 012700 001506    MOV      #$$SAVPAR,R0      ;GET ADDRESS OF SAVE BUFFER
10670 064034 010120          MOV      R1,(R0)+         ;PUT PAR DATA IN MEMORY
10671 064036 010220          MOV      R2,(R0)+
10672 064040 010320          MOV      R3,(R0)+
10673 064042 010510          MOV      R5,(R0)
10674 064044 010446          MOV      R4,-(SP)         ;PUT RETURN PC ON STACK
10675 064046 000207 1$:     RTS      PC
10676
10677      ;*****
10678      .SBTTL  CONTEXT SWITCH UP SUBROUTINE
10679      ;SUBROUTINE TO RESTORE KIPAR0, 1,2 AND 3 (IF MGMT ENABLED)
10680      ;*****
10681 064050 005737 001531    RESKT:  TSTB   @MMON          ;BRANCH IF MEM MGMT DISABLED
10682 064054 001421          BEQ     1$
10683 064056 012604          MOV      (SP)+,R4         ;GET RETURN PC
10684 064060 012700 001506    MOV      #$$SAVPAR,R0      ;GET ADDRESS OF SAVE BUFF
10685 064064 012001          MOV      (R0)+,R1         ;GET CLD PAR DATA
10686 064066 012002          MOV      (R0)+,R2
10687 064070 012003          MOV      (R0)+,R3
10688 064072 012005          MOV      (R0)+,R5
10689 064074 012700 172340    MOV      #KIPAR0,R0        ;GET ADDRESS OF PAR0
10690 064100 010120          MOV      R1,(R0)+         ;RELOCATE BACK
10691 064102 010220          MOV      R2,(R0)-
10692 064104 010320          MOV      R3,(R0)+
10693 064106 010510          MOV      R5,(R0)
10694 064110 013737 001516 177776  MOV      @$$SAVPSW,@PSW
10695 064116 010446          MOV      R4,-(SP)
10696 064120 000207 1$:     RTS      PC
10697
10698      ;*****
10699      .SBTTL  KT ABORT SUBROUTINE
10700      ;*****
10700 064122 016637 000002 001302  KTABRT: MOV      2(SP),@STMP0      ;SAVE ERROR PSW
10701 064130 011637 001522          MOV      (SP),@VADR       ;SAVE ERROR PC
10702 064134 162737 000002 001522  SUB      #2,@VADR
10703 064142 013737 177572 001306  MOV      @MMR0,@STMP2      ;SAVE MMR0
10704 064150 013737 177576 001310  MOV      @MMR2,@STMP3      ;SAVE MMR2
10705 064156 013737 001212 001266  MOV      @SLPERR,@SREG4    ;SAVE LOOP ADDRESS
10706 064164 012737 064174 001212  MOV      #1$,@SLPERR      ;SET RETURN ADR IF LOOPING
10707 064172 104003          ERROR   3
10708 064174 013737 001266 001212 1$:     MOV      @SREG4,@SLPERR    ;RESTORE LOOP ADR
10709 064202 042737 170000 177572  BIC     #170000,@MMR0     ;CLEAR ERRORS
10710 064210 013716 001212          MOV      @SLPERR,(SP)    ;GET LOOP ADDRESS
10711 064214 000002          RTI
10712
10713      ;*****
10714      .SBTTL  RESERVED INSTRUCTION ROUTINE
10715      ;*****
10716 064216 016637 000002 001302  RESERR: MOV      2(SP),@STMP0      ;SAVE PSW
10717 064224 011637 001522          MOV      (SP),@VADR       ;SAVE ERROR PC
10718 064230 162737 000002 001522  SUB      #2,@VADR
10719 064236 013737 001212 001266  MOV      @SLPERR,@SREG4    ;SAVE LOOP ADR
10720 064244 012737 064254 001212  MOV      #1$,@SLPERR      ;SET RETURN ADR IF LOOPING
10721 064252 104002          ERROR   2
10722 064254 013737 001266 001212 1$:     MOV      @SREG4,@SLPERR    ;RESTORE LOOP ADR
10723 064262 013716 001212          MOV      @SLPERR,(SP)    ;GET LOOP ADDRESS
```


10855 065246 020040 052502 040523
10856 065254 051104 020040 020040
10857 065262 051103 020062 020040
10858 065270 020040 051103 020061
10859 065276 050040 054510 020123
10860 065304 052502 040523 051104
10861 065312 000200
10862 065314 044124 020105 052521
10863 065322 041511 020113 051102
10864 065330 053517 020116 047506
10865 065336 020130 052512 050115
10866 065344 042105 047440 042526
10867 065352 020122 044124 020105
10868 065360 040514 054532 042040
10869 065366 043517 020123 040502
10870 065374 045503 030040 031061
10871 065402 032063 033065 034067
10872 065410 006471 000012
10873 065414 046111 042514 040507
10874 065422 020114 042504 044526
10875 065430 042503 000200
10876 065434 020040 020040 020040
10877 065442 020040
10878 065444 020040 020040 020040
10879 065452 000
10880 065453 125 044516 052502
10881 065460 020123 054105 051105
10882 065466 044503 042523 020122
10883 065474 000
10884 065475 117 052120 041456
10885 065502 036520 000
10886
10887
10888
10889 065505 125 042516 050130
10890 065512 041505 042524 020104
10891 065520 051124 050101 052040
10892 065526 020117 000064
10893 065532 041520 043117 050124
10894 065540 020040 044120 051531
10895 065546 041520 020040 020040
10896 065554 051520 020127 020040
10897 065562 050103 042525 051122
10898 065570 000
10899 065571 000 001 000
10900 065574 000 000
10901
10902 065576 001522 001522 001302
10903 065604 001306 000000
10904 065610 047125 054105 042520
10905 065616 052103 042105 052040
10906 065624 040522 020120 047524
10907 065632 030440 000060
10908 065636 041520 043117 050124
10909 065644 020040 044120 051531
10910 065652 041520 020040 020040

MSG1: .ASCII THE QUIR BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789/<15><12>

MSG2: .ASCII /ILLEGAL DEVICE/<CR LF>

MSG22: .ASCII / / /

MSG23: .ASCII / / /

MSG24: .ASCII /UNIBUS EXERCISER /

MSG25: .ASCII /OPT.CP=/

;MSG30 THROUGH MSG38 ARE AT END OF LISTING

EM1: .ASCII /UNEXPECTED TRAP TO 4/

DH1: .ASCII /PCOFTP PHYSPC PSW CPUERR/

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

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

EM2: .ASCII /UNEXPECTED TRAP TO 10/

DH2: .ASCII /PCOFTP PHYSPC PSW/

Address	Hex	Hex	Hex	Hex	Label	Text
10967	066306	040527	020123	054502		
10968	066314	041440	000120			
10969	066320	051123	040503	051104	DH6:	.ASCIZ /SRCADR DSTADR/
10970	066326	020040	051504	040524		
10971	066334	051104	000			
10972		066340				
10973	066340	001302	001522	000000	DT6:	.EVEN \$TMP0,VADR,0
10974	066346	051105	047522	020122	EM10:	.WORD ?ERROR DURING DATA CHECK-RELOC WAS BY I/O?
10975	066354	052504	044522	043516		
10976	066362	042040	052101	020101		
10977	066370	044103	041505	026513		
10978	066376	042522	047514	020103		
10979	066404	040527	020123	054502		
10980	066412	044440	047457	000		
10981	066417	123	041522	042101	DH10:	.ASCIZ /SRCADR DSTADR DEVICE THAT DID XFER/
10982	066424	020122	020040	051504		
10983	066432	040524	051104	020040		
10984	066440	042040	053105	041511		
10985	066446	020105	044124	052101		
10986	066454	042040	042111	054040		
10987	066462	042506	000122			
10988	066466	000	001	003	DF10:	.BYTE 0,1,3,0
10989	066471	000				
10990						
10991	066472	001302	001522	001306	DT10:	.EVEN \$TMP0,VADR,\$TMP2,\$TMP3,0
10992	066500	001310	000000			
10993	066504	044502	024124	024523	EM11:	.ASCIZ /BIT(S) STUCK IN MICRO-BREAK REGISTER/
10994	066512	051440	052524	045503		
10995	066520	044440	020116	044515		
10996	066526	051103	026517	051102		
10997	066534	040505	020113	042522		
10998	066542	044507	052123	051105		
10999	066550	000				
11000	066551	107	047517	042104	DH11:	.ASCIZ /GOODDAT BAD DATA/
11001	066556	052101	041040	042101		
11002	066564	042040	052101	000101		
11003	066572	000	000		DF11:	.BYTE 0,0
11004						
11005	066574	001302	001304	000000	DT11:	.EVEN \$TMP0,\$TMP1,0
11006	066602	041125	020105	047516	EM12:	.ASCIZ /UBE NON-EXISTANT MEMORY ERROR/
11007	066610	026516	054105	051511		
11008	066616	040524	052116	046440		
11009	066624	046505	051117	020131		
11010	066632	051105	047522	000122		
11011	066640	044120	051531	041040	DH12:	.ASCIZ /PHYS BUSADR/
11012	066646	051525	042101	000122		
11013	066654	002			DF12:	.BYTE 2
11014		066656				
11015	066656	001226	000000		DT12:	.EVEN \$GDDAT,0
11016	066662	041115	020124	047516	EM13:	.ASCIZ /MBT NON-EXISTANT MEMORY ERROR/
11017	066670	026516	054105	051511		
11018	066676	040524	052116	046440		
11019	066704	046505	051117	020131		
11020	066712	051105	047522	000122		
11021	066720	044120	051531	040440	DH13:	.ASCIZ /PHYS ADDRESS/
11022	066726	042104	042522	051523		

```
11023 066734 000
11024 066735 106 047514 052101 EM14: .ASCIZ /FLOATING POINT ERROR/
11025 066742 047111 020107 047520
11026 066750 047111 020124 051105
11027 066756 047522 000122
11028 066762 042011 040524 030524 DH14: .ASCIZ / DTAT1 DATA2/
11029 066770 004411 004411 040504
11030 066776 040524 000062
11031
11032 067002 001312 001262 001316 DT14: .EVEN
      067010 001264 000000 .WORD $TMP4,$REG2,$TMP6,$REG3,0
11033
11034 067014 004 000 004 DF14: .BYTE 4,0,4,0
11035 067017 000
11036 067020 042504 044526 042502 EM15: .ASCIZ /DEVICE HUNG/
11037 067026 044040 047125 000107
11038 067034 004411 040504 040524 DH16: .ASCIZ / DATA1 DATA2/
11039 067042 004461 004411 020011
11040 067050 020040 042040 052101
11041 067056 031101 000
11042 067061 005 000 005 DF16: .BYTE 5,0,5,0
11043 067064 000
11044 067066
11045 067066 001432 001262 001442 DT16: .EVEN
      067074 001264 000000 .WORD FLTMP0,$REG2,FLTMP1,$REG3,0
11046
11047 067100 030122 043040 044501 EM17: .ASCIZ /RO FAILED TO LOAD CORRECTLY ON MFPT/
11048 067106 042514 020104 047524
11049 067114 046040 040517 020104
11050 067122 047503 051122 041505
11051 067130 046124 020131 047117
11052 067136 046440 050106 000124
11053 067144 044503 020123 047111 EM20: .ASCIZ /CIS INSTRUCTION FAILURE/
11054 067152 052123 052522 052103
11055 067160 047511 020116 040506
11056 067166 046111 051125 000105
11057 067174 000000
11058
11059
11060 067176 050117 051105 052101
11061 067204 047511 040516 020114
11062 067212 053523 052111 044103
11063 067220 051440 052105 044524
11064 067226 043516 100123
11065 067232 053523 052111 044103 .ASCII /SWITCH USE/<CRLF>
11066 067240 004411 052411 042523
11067 067246 200
11068 067247 040 030440 004465 .ASCII / 15 HALT ON ERROR/<CRLF>
11069 067254 044011 046101 020124
11070 067262 047117 042440 051122
11071 067270 051117 200
11072 067273 040 030440 004464 .ASCII / 14 LOOP ON TEST/<CRLF>
11073 067300 046011 047517 020120
11074 067306 047117 052040 051505
11075 067314 100124
11076 067316 020040 031461 004411 .ASCII / 13 INHIBIT ERROR TYPEOUTS/<CRLF>
11077 067324 047111 044510 044502
11078 067332 020124 051105 047522
```

```
ENDTAG: .WORD 0
:*****
:THE FOLLOWING ASCII GETS OVERLAYED WHEN THE PROGRAM RUNS.
SWITCH: .ASCII /OPERATIONAL SWITCH SETTINGS/<CRLF>
```

11079	067340	020122	054524	042520		
11080	067346	052517	051524	200		
11081	067353	040	030440	004462	.ASCII / 12	INHIBIT USE/<CRLF>
11082	067360	044411	044116	041111		
11083	067366	052111	052440	042502		
11084	067374	200				
11085	067375	040	030440	004461	.ASCII / 11	INHIBIT ITERATIONS/<CRLF>
11086	067402	044411	044116	041111		
11087	067410	052111	044440	052124		
11088	067416	051105	052101	047511		
11089	067424	051516	200			
11090	067427	040	030440	004460	.ASCII / 10	BELL ON ERROR/<CRLF>
11091	067434	041011	046105	020114		
11092	067442	047117	042440	051122		
11093	067450	051117	200			
11094	067453	040	020040	004471	.ASCII / 9	LOOP ON ERROR/<CRLF>
11095	067460	046011	047517	020120		
11096	067466	047117	042440	051122		
11097	067474	051117	200			
11098	067477	040	020040	004470	.ASCII ? 8	ALLOW RELOCATION VIA I/O DEVICE (NOTE CHANGE)?<CRLF>
11099	067504	040411	046114	053517		
11100	067512	051040	046105	041517		
11101	067520	052101	047511	020116		
11102	067526	044526	020101	027511		
11103	067534	020117	042504	044526		
11104	067542	042503	024040	047516		
11105	067550	042524	041440	040510		
11106	067556	043516	024505	200		
11107	067563	040	020040	004467	.ASCII / 7	INHIBIT TYPEOUT OF THIS TEXT AND SYS SIZE/<CRLF>
11108	067570	044411	044116	041111		
11109	067576	052111	052040	050131		
11110	067604	047505	052125	047440		
11111	067612	020106	044124	051511		
11112	067620	052040	054105	020124		
11113	067626	047101	020104	054523		
11114	067634	020123	044523	042532		
11115	067642	200				
11116	067643	040	020040	004466	.ASCII / 6	INHIBIT RELOCATION/<CRLF>
11117	067650	044411	044116	041111		
11118	067656	052111	051040	046105		
11119	067664	041517	052101	047511		
11120	067672	100116				
11121	067674	020040	032440	004411	.ASCII / 5	INHIBIT ROUND ROBIN RELOCATION/<CRLF>
11122	067702	047111	044510	044502		
11123	067710	020124	047522	047125		
11124	067716	020104	047522	044502		
11125	067724	020116	042522	047514		
11126	067732	040503	044524	047117		
11127	067740	200				
11128	067741	040	020040	004464	.ASCII / 4	INHIBIT RANDOM DISK ADDRESS/<CRLF>
11129	067746	044411	044116	041111		
11130	067754	052111	051040	047101		
11131	067762	047504	020115	044504		
11132	067770	045523	040440	042104		
11133	067776	042522	051523	200		
11134	070003	040	020040	004463	.ASCII / 3	INHIBIT MBT/<CRLF>

11135	070010	044411	044116	041111
11136	070016	052111	046440	052102
11137	070024	200		
11138	070025	040	020040	004462
11139	070032	052011	042510	042523
11140	070040	052040	051110	042505
11141	070046	051440	044527	041524
11142	070054	042510	100123	
11143	070060	020040	030440	004411
11144	070066	051101	020105	047105
11145	070074	047503	042504	020104
11146	070102	047524	051440	046105
11147	070110	041505	020124	042522
11148	070116	047514	040503	044524
11149	070124	047117	200	
11150	070127	040	020040	004460
11151	070134	047411	020116	044124
11152	070142	020105	047506	046114
11153	070150	053517	047111	020107
11154	070156	042504	044526	042503
11155	070164	035123	200	
11156	070167	011	027060	027056
11157	070174	050122	030461	051057
11158	070202	030120	100063	
11159	070206	030411	027056	051056
11160	070214	030513	027461	045522
11161	070222	032460	200	
11162	070225	011	027062	027056
11163	070232	047516	020124	051525
11164	070240	042105	200	
11165	070243	011	027063	027056
11166	070250	047516	020124	051525
11167	070256	042105	200	
11168	070261	011	027064	027056
11169	070266	044122	030067	051057
11170	070274	030120	100064	
11171	070300	032411	027056	051056
11172	070306	033510	027460	051522
11173	070314	032060	047440	020122
11174	070322	051522	031460	200
11175	070327	011	027066	027056
11176	070334	047516	020124	051525
11177	070342	042105	200	
11178	070345	011	027067	027056
11179	070352	047516	020124	051525
11180	070360	042105	000200	
11181	070364	052200	042510	043040
11182	070372	046117	047514	044527
11183	070400	043516	042040	053105
11184	070406	041511	051505	040440
11185	070414	042116	042040	044522
11186	070422	042526	020123	044527
11187	070430	046114	041040	020105
11188	070436	051525	042105	043040
11189	070444	051117	051040	046105
11190	070452	041517	052101	047511

.ASCII / 2

THESE THREE SWITCHES/<CRLF>

.ASCII / 1

ARE ENCODED TO SELECT RELOCATION/<CRLF>

.ASCII / 0

ON THE FOLLOWING DEVICES:/<CRLF>

.ASCII ?

0...RP11/RP03?<CRLF>

.ASCII ?

1...RK11/RK05?<CRLF>

.ASCII ?

2...NOT USED?<CRLF>

.ASCII ?

3...NOT USED?<CRLF>

.ASCII ?

4...RH70/RP04?<CRLF>

.ASCII ?

5...RH70/RS04 OR RS03?<CRLF>

.ASCII ?

6...NOT USED?<CRLF>

.ASCII ?

7...NOT USED?<CRLF>

MSG4:

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

11191 070460 020116 043111 041040
11192 070466 052111 034040 051440
11193 070474 052105 100072
11194 070500 042504 044526 042503
11195 070506 042011 044522 042526
11196 070514 100123 000
11197 070517 015 025012 047052
11198 070524 052117 025105 020052
11199 070532 053523 052111 044103
11200 070540 051040 043505 041040
11201 070546 052111 034040 044040
11202 070554 051501 041040 042505
11203 070562 020116 042522 042526
11204 070570 051522 042105 044440
11205 070576 020116 042522 020126
11206 070604 100104
11207 070606 047516 042524 052040
11208 070614 040510 020124 053523
11209 070622 020122 044502 020124
11210 070630 020070 042523 020124
11211 070636 047516 020127 046101
11212 070644 047514 051527 044440
11213 070652 047457 051040 046105
11214 070660 041517 052101 047511
11215 070666 100116 200
11216 070671 124 044510 020123
11217 070676 051120 043517 040522
11218 070704 020115 052523 050120
11219 070712 051117 051524 044440
11220 070720 047457 051040 046105
11221 070726 041517 052101 047511
11222 070734 020116 047117 054514
11223 070742 053440 052111 020110
11224 070750 044124 020105 047506
11225 070756 046114 053517 047111
11226 070764 020107 042504 044526
11227 070772 042503 035123
11228 070776 051200 030120 026063
11229 071004 045522 032460 051054
11230 071012 030120 027464 027465
11231 071020 026066 051522 031460
11232 071026 032057 000
11233 071031 113 030502 026461
11234 071036 046505 005015 000
11235 071043 061 027461 032067
11236 071050 020040 020040 020040
11237 071056 020040 020040 045450
11238 071064 030502 041461 024515
11239 071072 005015 000
11240 071075 015 041412 052520
11241 071102 052440 042116 051105
11242 071110 052040 051505 020124
11243 071116 047506 047125 020104
11244 071124 047524 041040 020105
11245 071132 020101 000
11246 071135 113 030502 026461

.ASCIZ /DEVICE DRIVES/<CRLF>

MSG30: .ASCII <15><12>/**NOTE** SWITCH REG BIT 8 HAS BEEN REVERSED IN REV D/<CRLF>

.ASCII 'NOTE THAT SWR BIT 8 SET NOW ALLOWS I/O RELOCATION'<CRLF><CRLF>

.ASCII 'THIS PROGRAM SUPPORTS I/O RELOCATION ONLY WITH THE FOLLOWING DEVICES:'

.ASCIZ <CRLF>'RP03,RK05,RP04/5/6,RS03/4'

MSG31: .ASCIZ 'KB11-EM'<15><12>

MSG32: .ASCIZ '11/74 (KB11CM)'<15><12>

MSG34: .ASCIZ <15><12>'CPU UNDER TEST FOUND TO BE A ''

MSG35: .ASCIZ 'KB11-B/C'<15><12>

DEQKC-D PDP 11/70-74MP CPU EXERCISER
CEQKCD.P11 04-OCT-79 08:55

MACY11 30A(1052) 04-OCT-79^{1 3} 09:00 PAGE 219
TRAP TO 4 SERVICE ROUTINE

SEQ 0241

11247	071142	027502	006503	000012	
11248	071150	041113	030461	042455	MSG36: .ASCIZ 'KB11-E'<15><12>
11249	071156	005015	000		
11250	071161	103	051511	020120	MSG37: .ASCIZ /CISP OPTION NOT FOUND/<15><12>
11251	071166	050117	044524	047117	
11252	071174	047040	052117	043040	
11253	071202	052517	042116	005015	
11254	071210	000			
11255	071211	103	051511	020120	MSG38: .ASCIZ /CISP OPTION FOUND/<'5><12>
11256	071216	050117	044524	047117	
11257	071224	043040	052517	042116	
11258	071232	005015	000		
11259		000001			.END

A	041210	7125*	7231	7232	7320#	7391
AA	001533	734#	1236*			
ADCB2	013656	2624	2626#			
ADCB5	014516	2856	2858#			
ADCB6	015234	3014	3015	3017#		
ADCB7	016172	3239	3240	3241	3243#	
ADCO	011512	2018	2019	2020	2022#	
ADC1	012426	2249	2250	2251	2253#	
ADC2	013452	2553	2555#			
ADC5	014310	2783	2784	2786#		
ADC6	015030	2956	2957	2959#		
ADC7	016032	3196	3197	3199#		
ADDN	076050	547#	7069			
ADDNI =	076150	579#	7047			
ADDP =	076070	562#	7178			
ADDP1 =	076170	587#				
ADDC	016674	3407	3408	3409	3411#	
ADD1	017006	3445	3446	3448#		
ADD1A	017232	3528	3529	3530	3532#	
ADD1B	017250	3537	3538	3540#		
ADD2	017706	3684	3685	3687#		
ADD3	020500	3878	3880#			
ADD6	021072	3981	3982	3984#		
ADD7	021564	4099	4100	4101	4103#	
ARBEX	043032	7440	7483#			
ARBF IN	043020	7425	7447	7468	7470	7481#
ASHCLO	030402	5582#				
ASHCRO	030460	5603#				
ASHLO	030172	5532#				
ASHL1	031004	5716#				
ASHN =	076056	553#	7074			
ASHNI -	076156	585#	7039	7043	7051	7105
ASHP	076076	568#	7172	7183		
ASHP1 -	076176	593#	7155			
ASHRO	030306	5559#				
ASHR1	031072	5739#				
ASLB1	013004	2384	2385	2387#		
ASLB1A	013230	2471	2472	2474#		
ASLB3	014506	2850	2851	2853#		
ASLB4	013762	2661	2662	2663	2665#	
ASLB6	015216	3006	3007	3008	3010#	
ASLB7	016270	3271	3272	3274#		
ASLO	011634	2062	2063	2064	2065	2067#
ASL1	012602	2312	2313	2314	2316#	
ASL3	014224	2752	2753	2755#		
ASL4	013544	2585	2586	2587	2589#	
ASL6	015000	2944	2945	2947#		
ASL7	015660	3143	3144	3146#		
ASRB1	013100	2420	2422#			
ASRB1A	013114	2426	2427	2429#		
ASRB2	013726	2646	2647	2649#		
ASRB2A	013744	2654	2655	2657#		
ASRB5	014446	2831	2832	2834#		
ASRB6	015334	3046	3047	3049#		
ASRB7	016306	3278	3279	3281#		
ACRO	011662	2076	2077	2078	2080#	

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	001716	774#	1325*	1326*	10459	10462*	10491*
MAPTST	033510	6260#					
MAPTWO	033636	6259	6295	6300#			
MARKEX	027402	5369	5373	5377#			
MARK1	027356	5365	5367#				
MATC =	076045	546#	6908				
MATCH	037270	6900	6904#				
MATCI =	076145	578#					
MBRK	051664	5892#					
MBTAS =	160116	493#					
MBTBA =	160104	488#	974				
MBTBAE=	160174	497#	975				
MBTCS1=	160100	486#	972				
MBTCS2=	160110	490#	977				
MBTCS3=	160176	498#	980				
MBTDB =	160120	494#					
MBTDT =	160126	496#	983				
MBTER =	160114	492#	979				
MBTERR	054140	8873	8890#				
MBTMR1=	160124	495#					
MBTMR2=	160106	489#	976				
MBTN2	002336	984#	1428				
MBTN3	002340	985#	1432				
MBTN4	002342	986#	1436				
MBTOPT=	002000	507#	1426	1452	7511		
MBTPSW-	000776	500#	982				
MBTSET	043150	7501	7503	7511#			
MBTSRV	053766	7522	8861#				

RP3HDA	002102	864#	8139	8428										
RP3HDC	002104	865#	8112*	8140	8429									
RP3HST	001736	784#	1328	7760	7765	7811	7885*	7899*	7902	7904	7931	8073*	8074	8358
		8371*	8378*	8381*	8393	8436	8455*							
RP3HWC	001756	797#	7777*	8141	8430									
RP3LOO	050640	8359	8378#	8394	8437									
RP3NWH	002024	826#	7791*	8421	8423*	8426								
RP3NWL	002022	825#	7790*	8418	8431									
RP3OLD	001772	808#	7789*	7792*	8118	8121	8124*	8142	8386					
RP3PSW	002160	911#	8135*											
RP3REA	051166	8355	8436#											
RP3RPT	050462	8077	8344#											
RP3SRV	050512	8134	8351#											
RP3TRY	002132	888#	8114*	8362	8367*	8384*	8399	8405*	8413*	8440	8446*			
RP3UNI	002052	842#	7785*	8125	8128*	8132	8387	8427						
RP3VEC	002156	910#	8134*											
RP3WC	002146	906#	8141*	8430*										
RP3WCK	050730	8353	8393#											
RP3WTR	047126	8130#	8131	8349	8369									
RP31	050664	8347	8361	8384#										
RP310	001612	758#	8120*	8123*	8127*	8137	8385*	8386*	8387*	8389	8425*	8426*	8427*	8432
RP311	001614	759#	8078*	8093	8373	8397								
RP32	050714	8388#	8407											
RP33	051126	8350	8428#	8448										
RP4BA	002206	926#	8282*	8663*										
RP4BAE	002210	927#	8281*	8662*										
RP4CS1	002202	924#	1652	1682*	8273*	8613*	8634*	8664*	10769					
RP4CS2	002214	929#	1681*	1683	1704*	7895*	8276*	8606*	8645*	8651	8675*			
RP4CS3	002216	930#												
RP4DA	002212	928#	8279*											
RP4DC	002224	933#	8278*											
RP4DRV	047622	857	8222#											
RP4DS	002220	931#	1685	8602	8614	8625	8631	8640	8659	8670				
RP4ERR	052322	8605	8617#	8643	8673									
RP4ER1	002222	932#												
RP4ER2	002226	934#												
RP4ER3	002230	935#												
RP4FUN	002126	879#	8272*	8521*	8589*	8590	8594*	8595	8644*	8674*				
RP4HAN	002076	857#												
RP4HDA	002112	868#	8266*	8279										
RP4HDC	002114	869#	8265*	8278										
RP4HST	001746	788#	8223*	8224	8229*	8600	8611	8617*	8624*	8627*	8638	8668	8681*	
RP4HWC	001766	801#	8780											
RP4LOO	052350	8601	8624#	8639	8669									
RP4NWH	002044	834#	8662											
RP4NWL	002042	833#	8663											
RP4OF	002234	937#	8277*											
RP4OLD	002012	816#	8281	8282										
RP4PSW	002240	939#	8271*											
RP4REA	052574	8597	8668#											
RP4RPT	052156	8227	8588#											
RP4SRV	052202	8270	8594#											
RP4TRY	002135	891#	8267*	8604	8608*	8630*	8642	8647*	8658*	8672	8677*			
RP4UNI	002062	846#	8276											
RP4VEC	002236	938#	8270*											
RP4WC	002204	915#	8280*											

DC

SR2 = 177576	233#	6229													
SR3 = 172515	234#	7597*	10188*	10293*											
STACK = 001200	75#	76	77	78	1344										
START = 003542	608	1190	1218#	10020	10644										
START1 = 002544	610	1086#	1087												
START2 = 002554	611	1105#	1135												
START3 = 003330	613	1198#	1214												
STKLMT= 177774	83#	6020													
STMM = 043250	7505	7512	7514	7516	7518	7531#	7960								
SUBFLG = 036432	6702*	6731	6738*	6740#											
SUBN = 076051	548#	7064													
SUBN1 = 076151	580#	7055													
SUBP = 076071	563#														
SUBPAS = 001560	745#	1331*	7515	7991*	7992	7996*	8004	9115							
SUBPI = 076171	588#	7150	7160												
SUBO = 016434	3318	3319	3321#												
SUB1 = 017064	3471	3472	3474#												
SUB1A = 017172	3510	3511	3513#												
SUB1B = 017206	3517	3518	3520#												
SUB2 = 017752	3703	3704	3706#												
SUB2A = 020072	3745	3746	3748#												
SUB3 = 020424	3855	3856	3858#												
SUB3A = 020446	3863	3864	3866#												
SUB6 = 021112	3988	3990#													
SUB7 = 021540	4091	4092	4093	4095#											
SUPSTR- 000700	77#	1772	4336	4602	4768	4774	4809	4823	5636	5953	6058	6078	6082		
	626*	10732													
SWAB0 = 011714	2091	2092	2093	2095#											
SWAB1 = 013176	2456	2457	2459#												
SWAB2 = 013416	2536	2538#													
SWAB4 = 014052	2697	2699#													
SWAB6 = 015400	3065	3067#													
SWAB7 = 015766	3179	3181#													
SWITCH = 067176	9470	11060#													
SWR = 177570	85#	86	1086*	1757	7420*	7491*	7502	7513	7532*	7535	7627	7647	7684		
	7692	7696	7845	7909	8079	8157	8230	8296	8954	8972	8979	9019	9022		
	9029	9033	9039												
SW0 = 000001	151#														
SW00 = 000001	141#	151													
SW01 = 000002	140#	150													
SW02 = 000004	139#	149													
SW03 = 000010	138#	148													
SW04 = 000020	137#	147													
SW05 = 000040	136#	146													
SW06 = 000100	135#	145													
SW07 = 000200	134#	144													
SW08 = 000400	133#	143													
SW09 = 001000	132#	142													
SW1 = 000002	150#														
SW10 = 002000	131#														
SW11 = 004000	130#														
SW12 = 010000	129#	7502													
SW13 = 020000	128#														
SW14 = 040000	127#	8954													
SW15 = 100000	126#														
SW2 = 000004	149#														

TST15	015462	3094#	
TST16	016064	3215#	
TST17	016366	3305#	
TST2	011176	1919#	
TST20	016732	3429#	
TST21	017264	3549#	
TST22	017622	3671#	
TST23	020114	3759#	
TST24	020326	3830#	
TST25	020524	3894#	
TST26	020712	3951#	
TST27	021216	4013#	
TST3	011372	1982#	
TST30	021400	4056#	
TST31	021624	4120#	
TST32	021770	4160#	
TST33	022164	4213#	
TST34	022700	4342#	
TST35	023200	4434#	
TST36	023530	4539#	
TST37	023762	4557	4606#
TST4	011716	2099#	
TST40	024176	4662#	
TST41	024422	4712#	
TST42	025130	4832#	
TST43	025446	4920#	
TST44	025670	4971#	
TST45	026104	5038#	
TST46	026252	5088#	
TST47	027044	5261#	
TST5	012320	2222#	
TST50	027302	5351#	
TST51	027404	5402#	
TST52	027624	5447#	
TST53	030100	5514#	
TST54	030544	5632#	
TST55	030702	5684#	
TST56	031156	5768#	
TST57	031242	5792#	
TST6	012660	2346#	
TST60	031420	5836#	
TST61	031646	5889#	
TST62	031736	5912#	
TST63	032150	5957#	
TST64	032246	5978#	
TST65	032336	6001#	
TST66	032700	6095#	
TST67	033024	6135#	
TST7	013274	2497#	
TST70	033204	6171	6195#
TST71	033462	6255#	
TST72	033712	6324#	
TST73	035114	6499	6512#
TST74	036472	6765#	
TST75	036650	6803#	
TST76	042544	7417#	

UDPAR7=	177676	278#												
UDPDR0=	177620	249#	6177											
UDPDR1=	177622	250#												
UDPDR2=	177624	251#												
UDPDR3=	177626	252#												
UDPDR4=	177630	253#												
UDPDR5=	177632	254#												
UDPDR6=	177634	255#												
UDPDR7=	177636	256#												
UIPAR0=	177640	260#	5925	6182	7574*	7575	7577	7579						
UIPAR1=	177642	261#	7575*	7576*										
UIPAR2=	177644	262#	7577*	7578*										
UIPAR3=	177646	263#	7579*	7580*										
UIPAR4=	177650	264#	6202*											
UIPAR5=	177652	265#												
UIPAR6=	177654	266#	5925*											
UIPAR7=	177656	267#	7581*											
UIPDR0=	177600	238#	6174	7569*										
UIPDR1=	177602	239#	7570*											
UIPDR2=	177604	240#	7571*											
UIPDR3=	177606	241#	7572*											
UIPDR4=	177610	242#	6203*											
UIPDR5=	177612	243#												
UIPDR6=	177614	244#	5926*											
UIPDR7=	177616	245#	7573*											
UM	140000	517#	4569	4748	5454	5460	5486	5809	5823	5915	6052	6061		
UNITNO	001602	754#	7776*	7780	7785									
USESTK=	000600	78#												
UW6	014606	2889#	2893*	2900*	2905*	2912*	2920*	2927*	2934*	2941*	2943*	2949*	2955*	2961*
UW7	015560	3114#	3218	3221*	3226*									
UW7	015564	3112	3117#											
VADR	001522	728#	7663*	7850*	7935*	9085	9086*	9092*	9094	9096*	10342	10401*	10611*	10612*
		10701*	10702*	10717*	10718*	10730*	10731*	10902	10913	10928	10960	10973	10991	
X	063746	10645	10648#											
XOR0	026420	5123	5124	5125	5128#									
XOR1	026506	5148	5149	5150	5151	5153	5156#							
XOR24	026542	5170#												
XOR35	027012	5239	5240	5241#										
XOR6	026630	5175	5178	5180	5187#									
XOR6A	026634	5173*	5174*	5179*	5181*	5185	5191#	5196*	5207*	5209*				
XOR6B	026636	5182*	5183*	5184*	5185	5192#								
XOR7	027042	5255#												
XXCP	003570	1226#	1246											
XXDPC	003571	1227#	1241*	1753										
SACO	001412	707#	6351*	6354*	6361*	6363	6372*	6373	6382*	6386*	6390*	6394*	6409*	6424*
		6432*	6436*	6442*	6444	6450*	6452	6457*	6470*	6484*	6491*	6521*	6524*	6531*
		6533	6542*	6543	6552*	6556*	6560*	6564*	6579*	6594*	6602*	6606*	6612*	6614
		6620*	6622	6627*	6640*	6654*	6661*	6678	6689	6703	6708	6719	6721	997c
SAC1	001414	708#	6350*	6354	6357	6363*	6367	6373*	6376	6383*	6397	6405	6411*	6414*
		6415	6423*	6427	6430*	6439	6444*	6447	6452*	6454	6460	6466	6472*	6475*
		6476	6483*	6488	6494*	6496*	6497	6520*	6524	6527	6533*	6537	6543*	6546
		6553*	6567	6575	6581*	6584*	6585	6593*	6597	6600*	6604	6614*	6617	6622*
		6624	6630	6636	6642*	6645*	6646	6653*	6658	6661*	6666*	6667	6678*	6683*
		6703	6707	6720	6721*									
SAC2	001416	709#	6367*	6394	6402	6404*	6405*	6406*	6407*	6415	6427*	6436	6447*	6457
		6463	6465*	6466*	6467*	6468*	6476	6537*	6564	6572	6574*	6575*	6576*	6577*

\$AC3	001420	6585 710# 6633	6597* 6376* 6635	6606 6390	6617* 6402	6627 6404	6633 6454*	6635* 6463	6636* 6465	6637* 6546*	6638* 6560	6646 6572	6574	6624*
\$AC4	001422	711#												
\$AC5	001424	712#												
\$BDADR	001224	564#												
\$BDDAT	001230	666#	8840*	8898*	10438*	10467								
\$BELL	001332	700#	9024	9047										
\$BUFF	001410	706#	9830*	9855										
\$CHARC	056502	9377-	9384	9395*	9400#									
\$CMTAG	001200	651#	1339	1340	1347	1353	1354	1355						
\$CM1 =	000012	678#	679#	680#	681#	682#	683#	684#	685#	686#	687#	688#		
\$CM2 =	000024	678#	679#	680#	681#	682#	683#	684#	685#	686#	687#	688#		
\$CM3 =	000012	676#	678											
\$CM4 =	000012	688#	689#	690#	691#	692#	693#	694#	695#	696#	697#	698#		
\$SCORE	062002	10174	10202#											
\$CRLF	001337	702#	1276	1612	1750	8055	9032	9047	9077	9080	9122	9137	9142	9146
		9298	9317	9318	9367	9403	9454	9493	10077	10138	10535	10543	10553	
\$CROUT	062032	10202	10209#											
\$DBLK	057534	9600	9634	9642#										
\$DB20	057544	9098	9173	9293	9655#									
\$DOAGN	046600	8037	8058	8064#										
\$DTBL	057524	9603	9638#											
\$ENDAD	046570	637	1230	1568	8060#	9036								
\$ENDCT	046434	1353	8039#											
\$ENULL	046604	8066#												
\$EOP	046376	8000	8029#											
\$ENPCT	046426	1353*	8035#	8040										
\$ERFLG	001204	654#	7842	8946	8968	8970	8976*	8992	8994	8998	9015	9016*	9047	
\$ERMAX	001217	661#	1356*	8970	8991*	8998								
\$ERPSW	001562	746#												
\$ERROR	054622	1107	1347	4610	4654	9014#								
\$ERRPC	001220	662#	9026*	9027*	9028	9047	9081	9086						
\$ERRTB	002344	1004#	7839	8789	8791	9132	10275	10277						
\$ERRTY	055006	9031	9076#											
\$ERTTL	001214	659#	8052	8056*	9025*	9047								
\$ESCAP	001330	699#	1355*	8990*	9042	9044	9047							
\$FACTO	001536	736#	7637*	7753	7830	7833								
\$FILLC	001252	674#	9370	9403										
\$FILLS	001251	673#	9403											
\$FLBUF	001344	705#	9754	9802										
\$FLD20	060264	9824#	10269											
\$FL20	057776	9749#	10268											
\$GDADR	001222	663#												
\$GDDAT	001226	665#	8839*	8897*	10437*	10466	11015							
\$GET42	046560	8057#												
\$HD	000000	32												
\$INUM	001552	742#	1219*	6807	6811*	8085	8163	8236	8302	9875	9883	9888*	9911	9915
		10440												
\$ICNT	001206	656#	8983*	8984	8986*	8997								
\$ILLUP	061210	9991	10022#											
\$ITEMB	001216	660#	9028*	9047	9087	9089	9124							
\$KTNEA	061774	10175	10201#											
\$KTOUT	061764	10192	10198#	10230										
\$KT11	061624	1257*	10173#	10177*	10201*									
\$LF	001340	703#	9047	9403	10068	10077	10138							

\$SAVR6	061214	9999*	10005	10006*	10007*	10024#								
\$SCOPE	054370	1345	4555	4561	4656	5952	8953#							
\$SETUP=	000037	1320#	1345	1347	1349	1351	1353	1354	1355	1357	1568	8031	9033	
\$SIZE	061556	1258	10161#											
\$SIZE#	062036	10200	10210#											
\$STUP =	177777	1320#												
\$SVLAD	054554	8963	8988#											
\$SVPC =	000224	635#	640											
\$SWR =	167377	1#	32	47	48	49	50	51	52	53	54	698	699	700
		1354	1355	1357	1358	1804	1922	1985	2102	2225	2349	2500	2603	2721
		2806	2888	2973	3097	3218	3308	3432	3552	3674	3762	3833	3897	3954
		4016	4059	4123	4163	4216	4345	4437	4542	4609	4665	4715	4835	4923
		4974	5041	5091	5264	5354	5405	5450	5517	5635	5687	5771	5795	5839
		5892	5915	5960	5981	6004	6098	6138	6198	6258	6327	6515	6768	6806
		7420	7491	8026	8032	8059	8065	8066	8946	8947	8948	8949	8950	8954
		8966	8968	8969	8970	8977	8978	8979	8989	8992	8997	9005	9006	9007
		9008	9009	9010	9019	9022	9029	9033	9039	9047				
		8950												
\$SWRMK-	000000													
\$TIMES	001326	698#	1354*	1800*	1918*	2102*	2225*	3093*	4159*	4711*	5087*	5513*	6000*	6323*
		6327*	6515*	6764*	8032*	8977*	8984	8987*	8997					
\$TKB	001242	669#	10032	10043	10531									
\$TKS	001240	668#	1342	1361*	10032	10041								
\$TMP0	001302	688#	1133*	1624*	1627	1638*	1658*	1661	1665*	1678*	1681	1692	1705*	1713*
		1716	1727*	5862*	6348	6362	6421	6429	6449	6481	6489	6518	6532	6591
		6599	6619	6651	6659	6793*	7222*	7225*	7228*	7231*	7662*	7690*	7694	7699*
		7716	7762	7767	7849*	7934*	9904	9919	9951	10402*	10610*	10700*	10716*	10729*
		10739	10902	10913	10928	10960	10973	10991	11005					
\$TMP1	001304	689#	1134*	6794*	7223*	7226*	7229*	7232*	7707*	7709	7712	7714*	7770	7772*
		7871*	7874*	11005										
\$TMP10	001322	696#	8804*	8831*	8835	8863*	8889*	8893						
\$TMP11	001324	697#	8832*	8844	8890*	8900								
\$TMP2	001306	690#	6349	6370	6379	6422	6431	6441	6482	7688*	7870*	10398*	10628*	10703*
		10733*	10902	10928	10960	10991								
\$TMP3	001310	691#	7868*	10399*	10615*	10704*	10928	10960	10991					
\$TMP4	001312	692#	6356*	6408	6438*	6469	6487*	6519	6540	6549	6592	6601	6611	6652
		10400*	10616*	11032										
\$TMP5	001314	693#												
\$TMP6	001316	694#	6396*	6459*	6490*	11032								
\$TMP7	001320	695#												
\$TN	000100	1#	32	1797	1804#	1915	1922#	1979	1985#	2096	2102#	2219	2225#	2343
		2349#	2494	2500#	2597	2603#	2715	2721#	2800	2806#	2882	2888#	2967	2973#
		3090	3097#	3212	3218#	3302	3308#	3426	3432#	3546	3552#	3668	3674#	3756
		3762#	3827	3833#	3891	3897#	3948	3954#	4010	4016#	4052	4059#	4116	4123#
		4156	4163#	4210	4216#	4339	4345#	4431	4437#	4534	4542#	4557	4603	4609#
		4659	4665#	4708	4715#	4829	4835#	4917	4923#	4968	4974#	5035	5041#	5084
		5091#	5256	5264#	5348	5354#	5378	5405#	5444	5450#	5510	5517#	5626	5635#
		5677	5687#	5762	5771#	5789	5795#	5829	5839#	5885	5892#	5909	5915#	5954
		5960#	5975	5981#	5997	6004#	6090	6098#	6131	6138#	6171	6189	6198#	6251
		6258#	6313	6327#	6499	6502	6515#	6754	6768#	6800	6806#	7414	7420#	7484
		7491#												
\$TPB	001246	671#	9392*	9403	10576*									
\$TPFLG	001253	675#	9350	9403										
\$TPS	001244	670#	1385	7426	7429*	7441	7444*	7472*	7482*	7989	8005*	8012	8015*	9390
		9403	10536*	10544*	10554*	10572*	10574	10589*						
\$TRAP	062144	1106	1199	1263	1349	4677	4680	4684	4689	4697	10242#			
\$TRP	000030	10249#	10259#	10260#	10261#	10262#	10263#	10264#	10265#	10266#	10267#	10268#	10269#	10270#

	5138	5141	5155	5169	5186	5189	5201	5210	5214	5221	5227	5232	5243
	5249	5254	5270	5289	5306	5310	5312	5315	5340	5346	5366	5371	5375
	5410	5469	5484	5495	5524	5530	5553	5576	5596	5620	5644	5656	5675
	5707	5731	5755	5787	5807	5821	5918	5932	5961	5968	5986	6011	6017
	6043	6070	6209	6217	6245	6335	6341	6775	6781	7307#	7334#	7404#	7739#
	7940	7990	8044#	8066	8067#	8209	8354	8366	8404	8445	8472	8485	8525
	8566	8997	8998	9047	9327#	9403	9642#	9685#	10002	10023	10032	10076#	10077
.PARSR 063464	10138	10575	10642#	10927#	10972#	11014#	11044#						
	1791	7611	7984	10394	10609#	10636							

.SWRLO	1#	55#	58	61
.SACT1	1#	615		
.SCATC	1#	597		
.SCMTA	1#	2#	642	
.SDB2D	1#			
.SDB20	1#	6#	9643	
.SDIV	1#			
.SEOP	1#	8019		
.SERRO	1#	8998		
.SERRI	1#			
.SMULT	1#			
.SPOWE	1#	9986		
.SRAND	1#	7#	9858	
.SRDDE	1#	10077		
.SRDOC	1#			
.SREAD	1#	10028		
.SSAVE	1#	9686		
.SSB2D	1#			
.SSB20	1#			
.SSCOP	1#	2#	8940	
.SSIZE	1#	10138		
.SSUPR	1#			
.STRAP	1#	10233		
.STYPB	1#			
.STYPD	1#	9575		
.STYPE	1#	642#	9328	
.STYPO	1#	9497		
.1170	1#	71		

. ABS. 071235 000

ERRORS DETECTED: 0

DSKM:CEQKCD,DSKZ:CEQKCD.LST/CRF/SOL=CEQKCD.SML,CEQKCD.P11
RUN-TIME: 80 118 14 SECONDS
RUN-TIME RATIO: 1185/214=5.5
CORE USED: 33K (65 PAGES)