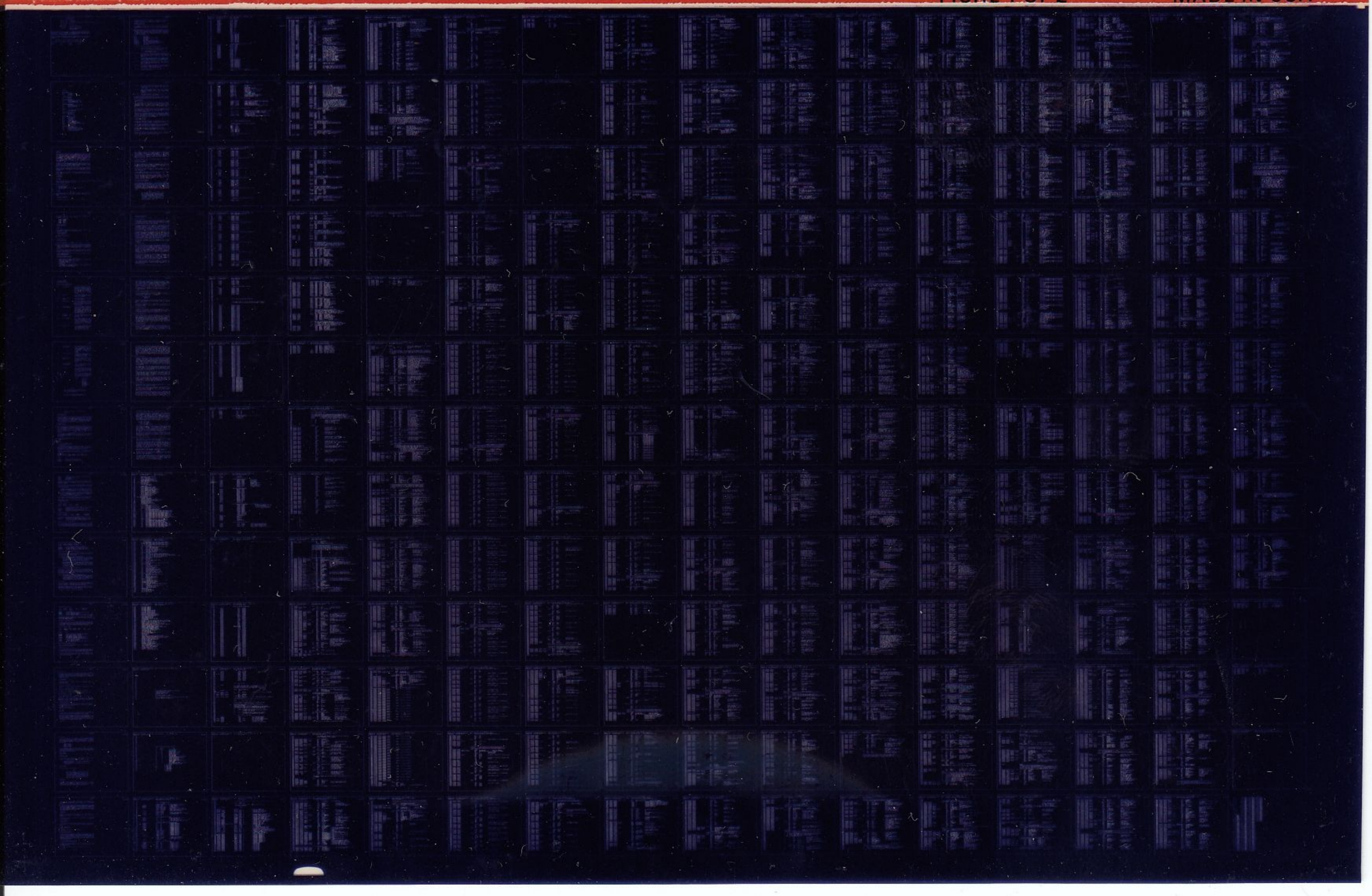
PDP-11/70

11/70-74 MP INST EXR
CEQKCD0

AH-7996D-MC

FICHE 1 OF 2

JAN 1980
digital
MADE IN USA



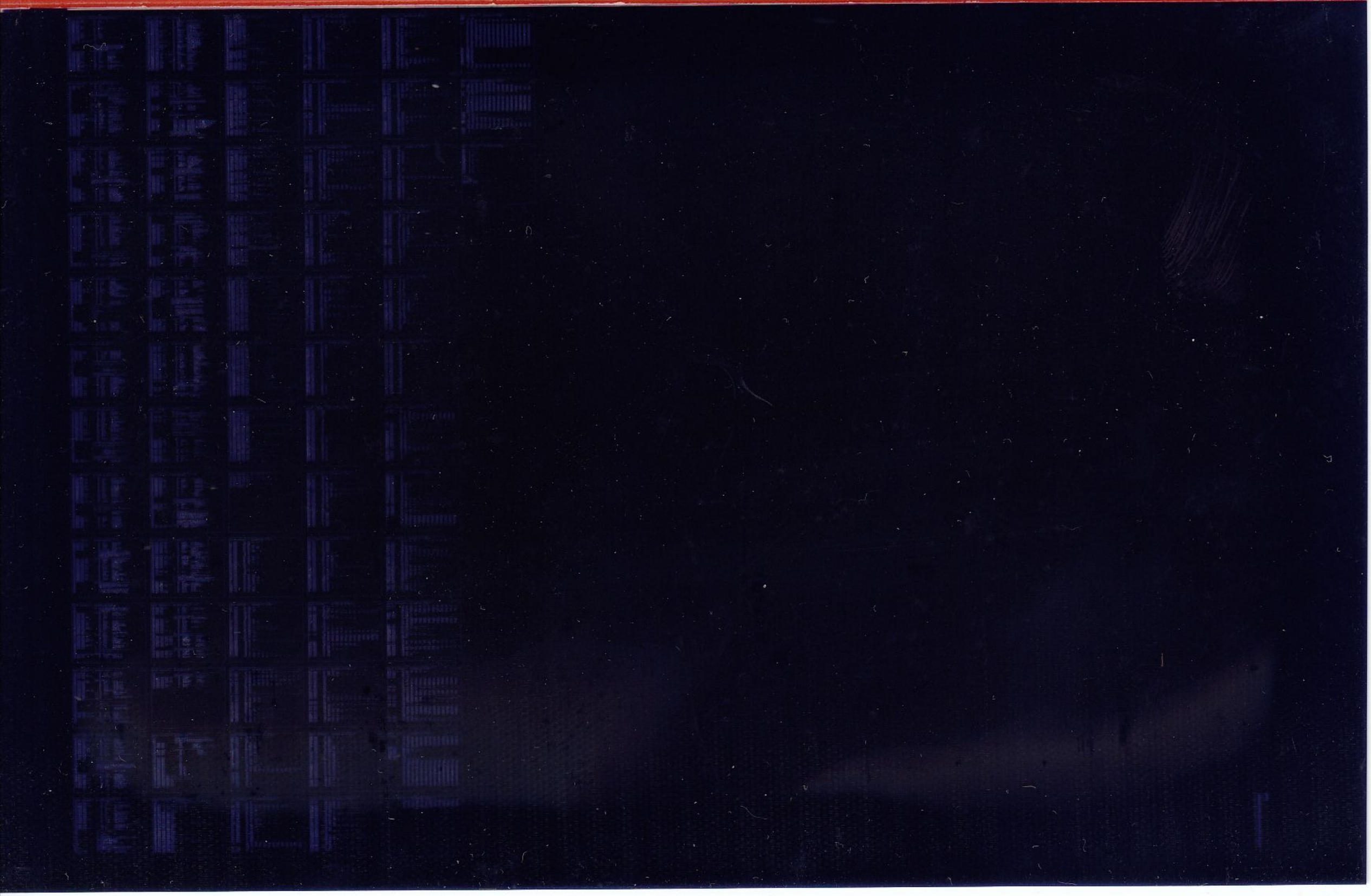
PDP-11/70

11/70-74 MP INST EXR
CEQKCD0

AH-7996D-MC

FICHE 2 OF 2

JAN 1980
digital
MADE IN USA



IDENTIFICATION

PRODUCT CODE: AC-7994D-MC

PRODUCT NAME: CEQKCDO PDP11/70-74MP CPU INSTRUCTION EXERCISER

DATE CREATED: MAY, 1979

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR(S): DONALD W. MONROE-REV B
JOHN ADAMS-REV A

CONTENTS

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 PROCEDURE

5.1 Operational Switch Settings

SW15	HALT ON ERROR	This switch when set will halt the processor when an error is detected. Pressing continue will cause an error message to be typed and the processor will again halt. Pressing continue again will resume testing.
SW14	LOOP ON TEST	This switch when set will cause the program to loop on the current subtest.
SW13	INHIBIT ERROR TYPEOUT	This switch when set inhibits the error typeout.
SW12	INHIBIT UBE	This switch when set inhibits the initialization of the Unibus Exerciser. See section 9.2 for a description of the UBE function.
SW11	INHIBIT SUB-TEST ITERATION	This switch when set inhibits subtest iteration after the first pass. Each subtest is executed 10 times before the next subtest is run. Setting SW11 causes each test to be executed once before starting the next subtest.
SW10	RING BELL ON ERROR	This switch when set will ring the bell when an error is detected.
SW9	LOOP ON ERROR	This switch when set will cause the program to loop on the first failure even if the failure is intermittant. 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 TYPEOUT	This switch when set will inhibit the typeout of the switch definitions and the disks that will be used for relocation. (Typeout only occurs when the program is dumped)
SW6	INHIBIT RELOCATION	This switch when set will inhibit all relocation. Do not change this switch while the program is running.
SW5	INHIBIT ROUND ROBIN	This switch when set will only relocate using the device selected by switches <2:0> rather than all available devices.
SW4	INHIBIT RANDOM DISK ADDRESS	This switch when set will cause relocation to always start at address 0 on the disk(s).
SW3	INHIBIT MBT	This switch when set inhibits the initialization of the Mass Bus Tester. See section 9.3 for a description of the MBT function.
SW2-SW0	DEVICE CODES	These switches (along with SW5) cause the program to relocate the test code using the device specified below:

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

NOTE

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

Unit 0 of the load device is marked not 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 PHYSC 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.
WWWWWW      = 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  PPPPPPPP    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   PPPPPPPP      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   EADRREG   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  CCCCCC  DDDDDD  EEEEEE    FFFFFF    GGGGGG

  ER         CS3
HHHHHH     JJJJJJ
```

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

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

6.1.12 Device Hung

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

6.2 Error Recovery

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

6.2.1 Errors Within Subtests

Execution starts with the instruction following the error call.

6.2.2 Relocation with Memory Mgmt. 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 '$LPERR'. 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.

9.0 PROGRAM DESCRIPTION

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

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

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

Following is a brief description of each section:

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

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

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

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

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

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

Section 5 This section checks the SXT, XOR, SOB, MARK, RTT 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 maping 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 "FRSTAD" and the relocation factor (byte offset from non-relocated code) is calculated and saved in location "FACTOR". The test code is then executed.

At the end of each section, control is transfered 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 transfered 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 transfered 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 transfered (i.e. the first 2K block is identified by word 1, the second 2K by word 2, etc.). The bit configuration of each word in the run table is as follows:

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

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

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

9.5.4.5 Device Service Routines

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

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

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

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

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

```
                        .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).
                        ;*
```

```
                        .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
```

```
                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
001200                  STACK=  1200            ;;FIRST ADDRESS OF THE STACK
001200                  KERSTK= STACK           ;;KERNEL STACK
000700                  SUPSTK= STACK-300       ;;SUPERVISOR STACK
000600                  USESTK= STACK-400       ;;USER STACK
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  SWR=    177570          ;;SWITCH REGISTER
177570                  DISPLAY=SWR

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE LINE FEED
000015                  CR=     15              ;;CODE CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
                        .EQUIV  R0,R10          ;;GENERAL REGISTER
                        .EQUIV  R1,R11          ;;GENERAL REGISTER
                        .EQUIV  R2,R12          ;;GENERAL REGISTER
                        .EQUIV  R3,R13          ;;GENERAL REGISTER
                        .EQUIV  R4,R14          ;;GENERAL REGISTER
                        .EQUIV  R5,R15          ;;GENERAL REGISTER
000006                  SP=%6
                        .EQUIV  SP,KSP          ;;KERNEL STACK POINTER
                        .EQUIV  SP,SSP          ;;SUPERVISOR STACK POINTER
                        .EQUIV  SP,USP          ;;USER STACK POINTER
000007                  PC=%7

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7

                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
```

```
                        SW14=   40000
                        SW13=   20000
                        SW12=   10000
                        SW11=   4000
                        SW10=   2000
                        SW09=   1000
                        SW08=   400
                        SW07=   200
                        SW06=   100
                        SW05=   40
                        SW04=   20
                        SW03=   10
                        SW02=   4
                        SW01=   2
                        SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                        BIT15=  100000
                        BIT14=  40000
                        BIT13=  20000
                        BIT12=  10000
                        BIT11=  4000
                        BIT10=  2000
                        BIT09=  1000
                        BIT08=  400
                        BIT07=  200
                        BIT06=  100
                        BIT05=  40
                        BIT04=  20
                        BIT03=  10
                        BIT02=  4
                        BIT01=  2
                        BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4                       ;;TIME OUT AND OTHER ERRORS
```

```
                        RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        TBITVEC=14              ;;"T" BIT
                        TRTVEC= 14              ;;TRACE TRAP
                        BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                        IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                        PWRVEC= 24              ;;POWER FAIL
                        EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                        TRAPVEC=34              ;;"TRAP" TRAP
                        TKVEC=  60              ;;TTY KEYBOARD VECTOR
                        TPVEC=  64              ;;TTY PRINTER VECTOR
                        CACHVEC=114             ;;CACHE ERROR INTERRUPT VECTOR
                        PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
                        MMVEC=  250             ;;MEMORY MANAGEMENT VECTOR

                        .SBTTL  CACHE   REGISTER DEFINITIONS

                        LOADRS = 177740         ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
                        HIADRS = 177742         ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
                        MEMERR = 177744         ;;CACHE ERROR REGISTER
                        CONTRL = 177746         ;;MEMORY CONTROL REGISTER
                        MAINT  = 177750         ;;MEMORY MAINTENENCE REGISTER
                        HITMIS = 177752         ;;HIT MISS REGISTER "1" IMPLIES HIT IN CACHE

                        .SBTTL  CPU REGISTER DEFINITIONS

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



                        .SBTTL  MEMORY MANAGEMENT DEFINITIONS


                        ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

                        MMR0=   177572
                        MMR1=   177574
                        MMR2=   177576
                        MMR3=   172516
                        .EQUIV  MMR0,SR0
                        .EQUIV  MMR1,SR1
                        .EQUIV  MMR2,SR2
                        .EQUIV  MMR3,SR3

                        ;*USER "I" PAGE DESCRIPTOR REGISTERS

                        UIPDR0= 177600
```

```
                177602          UIPDR1= 177602
                177604          UIPDR2= 177604
                177606          UIPDR3= 177606
                177610          UIPDR4= 177610
                177612          UIPDR5= 177612
                177614          UIPDR6= 177614
                177616          UIPDR7= 177616

                                ;*USER 'D'' PAGE DESCRIPTOR REGISTORS

                177620          UDPDR0= 177620
                177622          UDPDR1= 177622
                177624          UDPDR2= 177624
                177626          UDPDR3= 177626
                177630          UDPDR4= 177630
                177632          UDPDR5= 177632
                177634          UDPDR6= 177634
                177636          UDPDR7= 177636

                                ;*USER ''I'' PAGE ADDRESS REGISTERS

                177640          UIPAR0= 177640
                177642          UIPAR1= 177642
                177644          UIPAR2= 177644
                177646          UIPAR3= 177646
                177650          UIPAR4= 177650
                177652          UIPAR5= 177652
                177654          UIPAR6= 177654
                177656          UIPAR7= 177656

                                ;*USER 'D'' PAGE ADDRESS REGISTERS

                177660          UDPAR0= 177660
                177662          UDPAR1= 177662
                177664          UDPAR2= 177664
                177666          UDPAR3= 177666
                177670          UDPAR4= 177670
                177672          UDPAR5= 177672
                177674          UDPAR6= 177674
                177676          UDPAR7= 177676

                                ;*SUPERVISOR ''I'' PAGE DESCRIPTOR REGISTERS

                172200          SIPDR0= 172200
                172202          SIPDR1= 172202
                172204          SIPDR2= 172204
                172206          SIPDR3= 172206
                172210          SIPDR4= 172210
                172212          SIPDR5= 172212
                172214          SIPDR6= 172214
                172216          SIPDR7= 172216

                                ;*SUPERVISOR 'D'' PAGE DESCRIPTOR REGISTERS

                172220          SDPDR0= 172220
                172222          SDPDR1= 172222
```

```
                        172224          SDPDR2= 172224
                        172226          SDPDR3= 172226
                        172230          SDPDR4= 172230
                        172232          SDPDR5= 172232
                        172234          SDPDR6= 172234
                        172236          SDPDR7= 172236

                                        ;*SUPERVISOR ''I'' PAGE ADDRESS REGISTERS

                        172240          SIPAR0= 172240
                        172242          SIPAR1= 172242
                        172244          SIPAR2= 172244
                        172246          SIPAR3= 172246
                        172250          SIPAR4= 172250
                        172252          SIPAR5= 172252
                        172254          SIPAR6= 172254
                        172256          SIPAR7= 172256

                                        ;*SUPERVISOR ''D'' PAGE ADDRESS REGISTERS

                        172260          SDPAR0= 172260
                        172262          SDPAR1= 172262
                        172264          SDPAR2= 172264
                        172266          SDPAR3= 172266
                        172270          SDPAR4= 172270
                        172272          SDPAR5= 172272
                        172274          SDPAR6= 172274
                        172276          SDPAR7= 172276

                                        ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS

                        172300          KIPDR0= 172300
                        172302          KIPDR1= 172302
                        172304          KIPDR2= 172304
                        172306          KIPDR3= 172306
                        172310          KIPDR4= 172310
                        172312          KIPDR5= 172312
                        172314          KIPDR6= 172314
                        172316          KIPDR7= 172316

                                        ;*KERNEL ''D'' PAGE DESCRIPTOR REGISTERS

                        172320          KDPDR0= 172320
                        172322          KDPDR1= 172322
                        172324          KDPDR2= 172324
                        172326          KDPDR3= 172326
                        172330          KDPDR4= 172330
                        172332          KDPDR5= 172332
                        172334          KDPDR6= 172334
                        172336          KDPDR7= 172336

                                        ;*KERNEL ''I'' PAGE ADDRESS REGISTERS

                        172340          KIPAR0= 172340
                        172342          KIPAR1= 172342
                        172344          KIPAR2= 172344
```

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

                                ;*KERNEL 'D'' PAGE ADDRESS REGISTERS

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

                170200          MAPL00 = 170200
                170202          MAPH00 = 170202
                170204          MAPL01 = 170204
                170206          MAPH01 = 170206
                170210          MAPL02 = 170210
                170212          MAPH02 = 170212
                170214          MAPL03 = 170214
                170216          MAPH03 = 170216
                170220          MAPL04 = 170220
                170222          MAPH04 = 170222
                170224          MAPL05 = 170224
                170226          MAPH05 = 170226
                170230          MAPL06 = 170230
                170232          MAPH06 = 170232
                170234          MAPL07 = 170234
                170236          MAPH07 = 170236
                170240          MAPL10 = 170240
                170242          MAPH10 = 170242
                170244          MAPL11 = 170244
                170246          MAPH11 = 170246
                170250          MAPL12 = 170250
                170252          MAPH12 = 170252
                170254          MAPL13 = 170254
                170256          MAPH13 = 170256
                170260          MAPL14 = 170260
                170262          MAPH14 = 170262
                170264          MAPL15 = 170264
                170266          MAPH15 = 170266
                170270          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
```

```
000001                  AC1=    %1
000002                  AC2=    %2
000003                  AC3=    %3
000004                  AC4=    %4
000005                  AC5=    %5
                ;LINE CLOCK AND PROGRAMMABLE LINE CLOCK REGISTERS
172540                  PLKCSR=172540
172542                  PLKCSB=172542
000104                  PLKVEC=104
```

```
                        LKS=177546
000100                  LKVEC=100

                ;UNIBUS EXERCISER REGISTER
170000                  UBEDB=  170000          ;DATA BUFFER
170002                  UBECC=  170002          ;CYCLE COUNT
170004                  UBEBA=  170004          ;BUS ADDRESS
170006                  UBECR1= 170006          ;CRONTROL REGISTER 1
170010                  UBECLR= 170010          ;ERROR CLEAR
170014                  UBEGO=  170014          ;MULTI-EXERCISER GO
170016                  UBECR2= 170016          ;CONTROL REGISTER 2
000510                  UBEVEC= 510             ;INTERRUPT VECTOR

                ;MASS BUS TESTER REGISTERS
160100                  MBTCS1= 160100
160102                  MBTWC=  160102
160104                  MBTBA=  160104
160106                  MBTMR2= 160106
160110                  MBTCS2= 160110
160112                  MBTST=  160112
160114                  MBTER=  160114
160116                  MBTAS=  160116
160120                  MBTDB=  160120
160124                  MBTMR1= 160124
160126                  MBTDT=  160126
160174                  MBTBAE= 160174
160176                  MBTCS3= 160176
000774                  MBTVEC= 774
000776                  MBTPSW= 776

                ;MISCELLANEOUS BIT ASSIGNMENTS (USED IN OPT.CP)
100000                  KTOPT=  100000          ;BELOW BIT ASSIGNMENTS ARE USED
040000                  EISOPT= 040000          ;IN THE CPCHK ROUTINE
020000                  FPOPT=  020000          ;A BIT FOR EACH OPTION PRESENT
010000                  CISOPT= 010000          ;1174 CIS OPTION PRESENT BIT
002000                  MBTOPT= 002000
001000                  LKOPT=  001000
000400                  TTOPT=  000400
000200                  UBEOPT= 000200
                        .EQUIV  ERROR,HLT
                        .EQUIV  BIT14,SM
                        .EQUIV  BIT12,PSM
                        .EQUIV  BIT11,REG
000010                  CALLHANDLER=10
000000                  KM=0
140000                  UM=140000
000000                  PKM=0
030000                  PUM=30000
177770                  UBREAK=177770

                ;OPCODES USED IN 1174 CISP TESTS

076020                  L2D0=   076020          ;LOAD 2 DESCRIPTORS @R0 OPCODE
076061                  L3D1=   076061          ;LOAD 3 DESCRIPTORS @R1 OPCODE
076601                  MED74C= 076601          ;CISP DIAGNOSTIC ENTRY OPCODE
006600                  CISTST= 6600            ;ADDRESS OF A U-DIAGNOSTIC INSTRUCTION
```

```
528          000007                    MFPT=7                    ;OPCODE FOR MFPT INSTRUCTION USED FOR 1174 ONLY
```

```
                        .SBTTL CIS OPCODE DEFINITIONS

076021                  L2D1    =076021
076022                  L2D2    =076022
076023                  L2D3    =076023
076024                  L2D4    =076024
076025                  L2D5    =076025
076026                  L2D6    =076026
076027                  L2D7    =076027
076030                  MOVC    =076030
076031                  MOVRC   =076031
076032                  MOVTC   =076032
076040                  LOCC    =076040
076041                  SKPC    =076041
076042                  SCANC   =076042
076043                  SPANC   =076043
076044                  CMPC    =076044
076045                  MATC    =076045
076050                  ADDN    =076050
076051                  SUBN    =076051
076052                  CMPN    =076052
076053                  CVTNL   =076053
076054                  CVTPN   =076054
076055                  CVTNP   =076055
076056                  ASHN    =076056
076057                  CVTLN   =076057
076060                  L3D0    =076060
076062                  L3D2    =076062
076063                  L3D3    =076063
076064                  L3D4    =076064
076065                  L3D5    =076065
076066                  L3D6    =076066
076067                  L3D7    =076067
076070                  ADDP    =076070
076071                  SUBP    =076071
076072                  CMPP    =076072
076073                  CVTPL   =076073
076074                  MULP    =076074
076075                  DIVP    =076075
076076                  ASHP    =076076
076077                  CVTLP   =076077
076130                  MOVCI   =076130
076T31                  MOVRCI  =076131
076132                  MOVTCI  =076132
076140                  LOCCI   =076140
076141                  SKPCI   =076141
076142                  SCANCI  =076142
076143                  SPANCI  =076143
076144                  CMPCI   =076144
076145                  MATCI   =076145
076150                  ADDNI   =076150
076151                  SUBNI   =076151
076152                  CMPNI   =076152
076153                  CVTNLI  =076153
076154                  CVTPNI  =076154
076155                  CVTNPI  =076155
```

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


                                        .SBTTL  TRAP CATCHER

                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)
                000200                          .=200

000200  000137  003542                          JMP     @#START         ;;JUMP TO STARTING ADDRESS OF PROGRAM
                000210                  .=210
000210  000137  002544                          JMP     @#START1
000214  000137  002554                          JMP     @#START2
                                        ;;**26-APR-78,G.W.**
000220  000137  003330                          JMP     @#START3                ;ENTRY FOR PID REG. CUTTING AID
                                        ;;******************
                                        ;;*****************************************************************

                                        .SBTTL          ACT11 HOOKS

                                        ;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
                                        ;*
                                        ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOCICAL
                                        ;*END OF THE PROGRAM.
                                        ;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
                                        ;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
                                        ;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
                                        ;*
                                        ;*      BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
                                        ;*            =0 NO POWER FAIL DESIRED
                                        ;*
                                        ;*      BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
                                        ;*            =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
                                        ;*
                                        ;*      BITS 13-0  MUST BE ZERO'S

                000224                          $SVPC=.                 ;;SAVE LOCATION COUNTER
                000046                          .=46                    ;;SET LOCATION COUNTER
000046  046570                          .WORD   $ENDAD                  ;;SET LOC.46 TO ADDRESS $ENDAD
                000052                  .=52                            ;;SET LOCATION COUNTER
000052  040000                          .WORD   40000                   ;;SET LOC.52 TO 40000
                000224                  .=$SVPC                         ;; RESTORE LOCATION COUNTER
```

```
641
```

```
                                ;;*****************************************************************

                                .SBTTL  COMMON TAGS

                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.

                        001200          .=1200

001200                  $CMTAG:                         ;;START OF COMMON TAGS
001200  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001202  000000          $TSTNM: .WORD   0               ;;CONTAINS THE TEST NUMBER
001204  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
        001206                  .EVEN
001206  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001210  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP 1200
001212  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001214  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001216  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001217  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001220  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001222  000000          $GDADR: .WORD   0               ;;CONTAINS 1200 OF 'GOOD' DATA
001224  000000          $BDADR: .WORD   0               ;;CONTAINS 1200 OF 'BAD' DATA
001226  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001230  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001232  000000  000000  000000          .WORD   0,0,0   ;;RESERVED--NOT TO BE USED
001240  177560          $TKS:   177560                  ;;TTY KBD STATUS
001242  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001244  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. 1200
001246  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. 1200
001250  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001251  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001252  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
001253  000             $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
001254  000000          $REGAD: .WORD   0               ;;CONTAINS THE 1200 FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001256  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001260  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001262  000000          $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001264  000000          $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001266  000000          $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001270  000000          $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001272  000000          $REG6:  .WORD   0               ;;CONTAINS (($REGAD)+14)
001274  000000          $REG7:  .WORD   0               ;;CONTAINS (($REGAD)+16)
001276  000000          $REG10: .WORD   0               ;;CONTAINS (($REGAD)+20)
001300  000000          $REG11: .WORD   0               ;;CONTAINS (($REGAD)+22)
001302  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001304  000000          $TMP1:  .WORD   0               ;;USER DEFINED
001306  000000          $TMP2:  .WORD   0               ;;USER DEFINED
001310  000000          $TMP3:  .WORD   0               ;;USER DEFINED
001312  000000          $TMP4:  .WORD   0               ;;USER DEFINED
001314  000000          $TMP5:  .WORD   0               ;;USER DEFINED
001316  000000          $TMP6:  .WORD   0               ;;USER DEFINED
001320  000000          $TMP7:  .WORD   0               ;;USER DEFINED
001322  000000          $TMP10: .WORD   0               ;;USER DEFINED
001324  000000          $TMP11: .WORD   0               ;;USER DEFINED
```

```
                                                B 4
698  001326  000000                $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
699  001330  000000                $ESCAPE:0                       ;;ESCAPE ON ERROR 1200
700  001332  177607  000377        $BELL:  .ASCIZ  <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:    .ASCIZ  <12>            ;;LINE FEED
704  001342  000000                ERRRTN: .WORD
705  001344  000044                $FLBUFF:.BLKB   44      ;BUFFER FOR FLOATING POINT CONVERSION
706  001410  000000                $BUFF:  .WORD           ;
707  001412  000000                $AC0:   .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                $$TMP4: .WORD
714  001430  000000                $$TMP6: .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.CP: .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 INTERUPT 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 FREEE 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  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)
```

```
                                UNITNO: .WORD           ;DEVICE UNIT NUMBER
                                RNTBINX:.WORD           ;INDEX TO RUN TABLE
                                MXMMHI: .WORD           ;BITS<21:16> OF LAST MEM ADDRESS ON SYSTEM
                                MXMMLO: .WORD           ;BITS<15:00> OF LAST MEM ADDRESS ON SYSTEM
                                RP310:  .WORD           ;DATA TO LOAD INTO RP03 CS REGISTER
                                RP311:  .WORD           ;RP03 FLAG FOR FIRST 2K OF PROGRAM
                                RK10:   .WORD           ;DATA TO LOAD INTO RK05 CS REGISTER
                                RK11:   .WORD           ;RK05 FLAG FOR FIRST 2K OF PROGRAM
                                RP411:  .WORD           ;RP04 FLAG FOR FIRST 2K OF PROGRAM
                                RS11:   .WORD           ;RS04 FLAG FOR FIRST 2K OF PROGRAM
                                MTICKS: .WORD           ;ELAPSED RUN TIME IN MINUTES
                                LTICKS: .WORD           ;LOW BYTE=NUMBER OF CLOCK INTERRUPTS (0 TO 59)
                                                        ;HIGH BYTE=ELAPSED RUN TIME IN SECONDS(0 TO 59)
                                LD2PNT: .WORD   0       ;NEXT 3 WORDS USED FOR CISP DETECTION
                                LD2PT1: .WORD   0
                                LD3PNT: .WORD   0
                                $MAINT: .WORD           ;CURRENT VALUE IN MAINTENANCE REGISTER
                                SYSSIZE:.BLKW   10      ;SYSTEM SIZE TABLE(ONE ENTRY FOR EACH DEVICE)
                                RUNTBL: .BLKW   7       ;RUN TIME TABLE(ONE ENTRY FOR EACH 2K BLOCK)
                                RUNTRAK:.BLKW   7       ;RUN TRACK TABLE(ONE ENTRY FOR EACH 2K BLOCK)
                                MAPTBL: .WORD   -1      ;MAP TABLE(ONE BYTE FOR EACH UNIBUS DEVICE)
                                        .WORD   -1      ;UNUSED=377, USED=LOW 5 BITS OF MAP ADDRESS
                                UBESAV: .BLKW   2       ;BASE ADDRESS OF UBE TRANSFER IN PROGRESS
                                UBEADR: .BLKW   2       ;ADDRESS THAT GETS LOADED INTO UBE BA REG
                                ERRBA:  .BLKW   2       ;18 BIT UNIBUS ADDRESS WHEN DEVICE DETECTED AN ERROR
                                .SBTTL  DEVICE HANDLER STATUS WORDS
                                ;*      EACH WORD HAS THE FOLLOWING BIT ASSIGNMENTS:
                                ;*              7       HANDLER READY
                                ;*              8       REPEAT LAST FUNCTION
                                ;*              15      ERROR
                                RP3HSTAT:.WORD  200     ;RP03
                                RKHSTAT:.WORD   200     ;RK05
                                SPARE0: .WORD   200
                                SPARE1: .WORD   200
                                RP4HSTAT:.WORD  200     ;RP04
                                RSHSTAT:.WORD   200     ;RS04
                                        .WORD   200     ;SPARE
                                        .WORD   200     ;SPARE

                                .SBTTL  DEVICE HANDLER WORD COUNTS
                                ;*      THIS TABLE GETS LOADED BY THE I/O
                                ;*      RELOCATION ROUTINE WITH THE TWO'S COMPLIMENT WORD
                                ;*      COUNT FOR THE TRANSFER FOR THE PARTICULAR DEVICE.
                                RP3HWC: .WORD           ;RP03
                                RKHWC:  .WORD           ;RK05
                                        .WORD           ;SPARE
                                        .WORD           ;SPARE
                                RP4HWC: .WORD           ;RP04
                                RSHWC:  .WORD           ;RS04

                                .SBTTL  DEVICE HANDLER OLD BASE ADDRESS
                                ;*      THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE
                                ;*      WITH THE BASE ADDRESS OF THE SOURCE DATA FOR THE
                                ;*      DEVICE THAT IS GOING TO TRANSFER THE DATA.
                                RP3OLD: .WORD           ;RP03
                                        .WORD
```

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

                                .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
                                        .WORD           ;SPARE
                                        .WORD
                                        .WORD           ;SPARE
                                        .WORD
                                RP4NWL: .WORD           ;RP04
                                RP4NWH: .WORD
                                RSNEWL: .WORD           ;RS04
                                RSNEWH: .WORD

                                .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

                                .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

                                .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
```

```
                                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
```

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

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

                                        .SBTTL  UNIBUS EXERCISER REGISTER ADDRESS TABLE
                                        ;*      THIS TABLE IS ASSEMBLED FOR UBE #0. IF THE UBE
                                        ;*      ADDRESSES ARE CUT FOR OTHER THAN UNIT #0, THE PROGRAM
                                        ;*      WILL CHANGE THIS TABLE. THE PROGRAM LOOKS FOR A
                                        ;*      UBE AT ADDRESSES 770002, 770022, 770032, AND 770042.
002270  170002                  UBETBL: .WORD   UBECC   ;CYCLE COUNT
002272  170004                          .WORD   UBEBA   ;BUS ADDRESS REG
002274  170016                          .WORD   UBECR2  ;CONTROL REGISTER #2
002276  170006                          .WORD   UBECR1  ;CONTROL REGISTER #1
002300  170010                          .WORD   UBECLR  ;UBE CLEAR ADDRESS
002302  000510                          .WORD   UBEVEC  ;INTERRUPT VECTOR
002304  000512                          .WORD   UBEVEC+2        ;INTERRUPT VECTOR +2

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

```
 978  002322  160112                        .WORD   MBTST   ;STATUS REGISTER
 979  002324  160114                        .WORD   MBTER   ;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
```

```
                                ;;*****************************************************************

                                .SBTTL  ERROR POINTER TABLE

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

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


002344                          $ERRTB:
                                        ;ITEM 1
002344  065505                          EM1             ;UNEXPECTED TRAP TO 4
002346  065532                          DH1             ;PCOFTP  PHYSC      PSW  CPUERR
002350  065576                          DT1             ;VADR,VADR,$TMP0,$TMP2
002352  065571                          DF1             ;0,1,0,0,0
                                        ;ITEM 2
002354  065610                          EM2             ;UNEXPECTED TRAP TO 10
002356  065636                          DH2             ;PCOFTP  PHYSPC  PSW
002360  065664                          DT2             ;VADR,VADR,$TMP0
002362  065571                          DF1
                                        ;ITEM 3
002364  065674                          EM3             ;UNEXPECTED TRAP TO 250(MGMT)
002366  065731                          DH3             ;PCOFTP  PHYSPC    PSW    MMR0    MMR2
002370  066000                          DT3             ;VADR,VADR,$TMP0,,$TMP2,,$TMP3
002372  065571                          DF1
                                        ;ITEM 4
002374  066014                          EM4             ;UNEXPECTED TRAP TO 114
002376  066043                          DH4             ;PCOFTP  PHYSPC   PSW    ERADREG MEMERRREG
002400  066000                          DT3             ;VADR,VADR,$TMP0,$TMP3,$TMP2
002402  066121                          DF4             ;0,1,0,2,0
                                        ;ITEM 5
002404  066126                          EM5             ;PARITY ERROR DURING DATA CHECK
002406  066165                          DH5             ;SRCADR DSTADR ERRADREG  MEM ERR REG
002410  066236                          DT5             ;$TMP0,PA1500,$TMP3,$TMP2
002412  066232                          DF5
                                        ;ITEM 6
002414  066250                          EM6             ;ERROR DURING CHECK OF RELOCATED DATA
002416  066320                          DH6             ;SRCADR  DSTADR
002420  066340                          DT6             ;$TMP0,PA1500
002422  066121                          DF4
                                        ;ITEM 7
002424  000000                          0
002426  000000                          0
002430  000000                          0
002432  000000                          0
                                        ;ITEM 10
002434  066346                          EM10            ;ERROR DURING DATA CHECK-RELOC WAS BY I/O
002436  066417                          DH10            ;SRCADR   DSTADR   DEVICE THAT DID XFER
002440  066472                          DT10            ;$TMP0,VADR,$TMP2,$TMP3
```

```
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            ;R0 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
1081                                  ;ITEM 20
1082  002534  067144                  EM20            ;CIS INSTRUCTION FAILURE
1083  002536  066320                  DH6
1084  002540  066574                  DT11
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                                  .SBTTL  MICRO-BREAK REGISTER TEST
1092                                  ;*THIS TEST IS EXECUTED BY STARTING THE PROGRAM AT ADDRESS 214.
1093                                  ;*THIS TEST REQUIRES A MAINTENANCE CARD AND OPERATOR INTERVENTION.
1094                                  ;*THE PROCESSOR SHOULD STOP 8 TIMES. FOLLOWING IS THE DATA
1095                                  ;*THAT SHOULD BE IN THE MICRO-ADRESS DATA LIGHTS EACH TIME:
1096                                  ;*      1       000
1097                                  ;*      2       001
1098                                  ;*      3       002
1099                                  ;*      4       004
1100                                  ;*      5       010
1101                                  ;*      6       020
1102                                  ;*      7       040
1103                                  ;*      8       200
1104                                  ;;*****************************************************************
1105  002554  012706  001100          START2: MOV     #1100,SP        ;SETUP THE SP
1106  002560  012737  062144  000034          MOV     #$TRAP,@#TRAPVEC        ;SETUP TRAP VECTOR
1107  002566  012737  054622  000030          MOV     #$ERROR,@#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$:    MOV     R0,@#UBREAK     ;LOAD U BREAK REG
1113  002616  020037  177770                 CMP     R0,@#UBREAK     ;LOAD OK?
1114  002622  001036                         BNE     UBRERR          ;BRANCH IF NO
1115  002624  005000                         CLR     R0
1116  002626  010037  177770                 MOV     R0,@#UBREAK
1117  002632  020037  177770                 CMP     R0,@#UBREAK
1118  002636  001030                         BNE     UBRERR
1119  002640  012700  000125                 MOV     #125,R0
1120  002644  005737  001502                 TST     @#KB11E         ;IS THIS A KB11-E OR KB11EM PROCESSOR?
1121  002650  001402                         BEQ     2$              ;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  001300;        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 STACKK 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     #$TRAP,@#TRAPVEC        ;SET UP TRAP VECTOR
1200  003342  104400  003350                  TYPE    ,65$                ;;TYPE ASCIZ STRING
1201  003346  000415                          BR      64$                 ;;GET OVER THE ASCIZ
1202                                  ;;65$:  .ASCIZ  <15><12>/PID REGISTER SETUP AID/
1203  003402                          64$:
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,@#$HINUM         ;INITIALIZE RANDOM NUM GEN
1220  003554  012737  123456  001550          MOV     #123456,@#$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     @#$ENDAD+4              ;BRANCH IF NOT QV
1231  003602  100003                          BPL     1$
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
```

```
                                        ;SIZE MEMORY AND COMPARE IT WITH THE SYSTEM SIZE REGISTER
                                        ;PRINT A WARNING IF THEY DISAGREE.
003700  052767  000200  055716          BIS     #BIT07,$KT11
003706  004767  055644                  JSR     PC,$SIZE
003712  062767  000037  056222          ADD     #37,    $LSTBK
003720  016702  056216                  MOV     $LSTBK,R2               ;COPY LAST BLOCK COUNT
003724  023702  177760                  CMP     @#SIZELO,R2             ;EQUAL?
003730  001551                          BEQ     OKSIZ
003732  012737  062144  000034          MOV     #$TRAP,@#TRAPVEC        ;SET UP TRAP VECTOR
003740  104400  003746                  TYPE    ,65$                ;;TYPE ASCIZ STRING
003744  000433                          BR      64$                 ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/WARNING- THE SIZE OF MEMORY IS DIFFERENT FROM THAT/
004034                          64$:
004034  104400  004042                  TYPE    ,67$                ;;TYPE ASCIZ STRING
004040  000425                          BR      66$                 ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  <15><12>/INDICATED BY THE SYSTEM SIZE REGISTER./
004114                          66$:
004114  104400  004122                  TYPE    ,69$                ;;TYPE ASCIZ STRING
004120  000421                          BR      68$                 ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  <15><12>/     SIZEHI  SIZELO      ACTUAL/
004164                          68$:
004164  104400  001337                  TYPE    ,$CRLF
004170  013746  177762                  MOV     @#SIZEHI,-(SP)  ;;SAVE @#SIZEHI FOR TYPEOUT
004174  104404                          TYPOS                   ;;GO TYPE--OCTAL ASCII
004176     006                          .BYTE   6               ;;TYPE 6 DIGIT(S)
004177     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
004200  104400  004206                  TYPE    ,71$                ;;TYPE ASCIZ STRING
004204  000404                          BR      70$                 ;;GET OVER THE ASCIZ
                                ;;71$:  .ASCIZ  /       /
004216                          70$:
004216  013746  177760                  MOV     @#SIZELO,-(SP)  ;;SAVE @#SIZELO FOR TYPEOUT
004222  104404                          TYPOS                   ;;GO TYPE--OCTAL ASCII
004224     006                          .BYTE   6               ;;TYPE 6 DIGIT(S)
004225     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
004226  104400  004234                  TYPE    ,73$                ;;TYPE ASCIZ STRING
004232  000404                          BR      72$                 ;;GET OVER THE ASCIZ
                                ;;73$:  .ASCIZ  /       /
004244                          72$:
004244  016746  055672                  MOV     $LSTBK,-(SP)    ;;SAVE $LSTBK FOR TYPEOUT
004250  104404                          TYPOS                   ;;GO TYPE--OCTAL ASCII
004252     006                          .BYTE   6               ;;TYPE 6 DIGIT(S)
004253     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
                                ;
                                ;FORM MXMMHI, MXMMLO, AND THE HIGHEST MEMORY ADDRESS BASED ON THE SIZE OF
                                ;THE MEMORY
                                ;
004254                          OKSIZ:
004254  005002                          CLR     R2
004256  013703  062142                  MOV     @#$LSTBK,R3
004262  073227  000006                  ASHC    #6,R2               ;SHIFT TO FORM CORRECT ADDRESS
004266  052703  000077                  BIS     #77,R3              ;ENSURE LOWER SIX BITS SET
004272  062703  000001                  ADD     #1,R3
004276  005502                          ADC     R2
                                ;;**************
004300  010237  001606                  MOV     R2,@#MXMMHI         ;SAVE UPPER SIX BITS
```

```
1310  004304  010337  001610              MOV     R3,@#MXMMLO     ;SAVE LOWER 16 BITS
1311
1312  004310  012706  001200              MOV     #KERSTK,SP              ;SET STACK PTR
1313  004314  005037  001200              CLR     @#$PASS                 ;CLEAR PASS COUNT
1314  004320  105037  001531              CLRB    @#MMON                  ;SET MEM MGMT ON IND=NOT ON
1315  004324  012737  000700  001546      MOV     #700,@#NEXPAR           ;SET FIRST 'PAR' VALUE
1316  004332  005737  003574              TST     @#PROT
1317  004336  001403                      BEQ     1$
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  000002              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  001736              MOV     #RP3HSTAT,R1            ;GET ADDRESS OF HANDLER STAT
1329  004412  012721  000200      3$:     MOV     #200,(R1)+              ;INITIALIZE STATUS TABLE
1330  004416  077003                      SOB     R0,3$                   ;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  000012              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     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1341  004474  005026                      CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1342  004476  022706  001240              CMP     #$TKS,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     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1346  004516  012737  000340  000022      MOV     #340,@#IOTVEC+2 ;;LEVEL 7
1347  004524  012737  054622  000030      MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1348  004532  012737  000340  000032      MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1349  004540  012737  062144  000034      MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1350  004546  012737  000340  000036      MOV     #340,@#TRAPVEC+2;LEVEL 7
1351  004554  012737  061062  000024      MOV     #$PWRDN,@#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     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1358  004622  012767  004622  174362      MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
1359
1360                          ;CLEAR PROGRAM INDICATORS
1361  004630  052777  000100  174402              BIS     #100,@$TKS              ;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$:     SEC
1381  004744  005737  177546                  TST     @#LKS                   ;BRANCH IF NO KW11-L
1382  004750  103402                          BCS     7$
1383  004752  052702  001000                  BIS     #LKOPT,R2               ;SET OPTION INDICATOR
1384  004756  000261                  7$:     SEC
1385  004760  005777  174260                  TST     @$TPS                   ;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          16$:    BIS     #UBEOPT,R2      ;SET INDICATOR
1396  005022  000425                          BR      17$
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     #ERPRT,@#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  040000          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          BIT     #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                    BEQ     1$                ;BR IF NEITHER. MUST BE KB11CM
 1514  005524  012737  000400  001502    MOV     #BIT8,@#KB11E     ;SET UPPER BYTE (KB11-EM)
```

```
1515  005532  000405                    BR      2$              ;DONE
1516  005534  105237  001504    1$:     INCB    @#KB11CM        ;YES, FLAG THIS AS A MODIFIED PROCESSOR
1517  005540  005737  001502    3$:     TST     @#KB11E         ;IS THIS A KB11E?
1518  005544  001472                    BEQ     RESTORE         ;BR IF NOT. THIS IS AN 1170
1519  005546  104422            2$:     RESREG                  ;RESTORE R5-R0
1520  005550  000403                    BR      ENDKB           ;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                                ENDKB:
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             CMPB    #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     #$ENDAD,@#42     ;;ACT-11?
1569  005762  001430                             BEQ     64$              ;;BRANCH IF YES
1570  005764  104400  005772                     TYPE    ,65$             ;;TYPE ASCIZ STRING
1571  005770  000425                             BR      64$              ;;GET OVER THE ASCIZ
1572                                     ;;65$:  .ASCIZ  <CRLF>''CEQKC-D...PDP 11/70-74MP CPU EXERCISER''<CRLF>
1573  006044                             64$:
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$:   TYPE    ,MSG38          ;CISP OPTION FOUND<15><12>
1595  006146                  100$:
1596  006146  104400  006154          TYPE    ,67$            ;;TYPE ASCIZ STRING
1597  006152  000415                  BR      66$             ;;GET OVER THE ASCIZ
1598                          ;;67$:   .ASCIZ  <15><12>/PROCESSOR ID REGISTER =/
1599  006206                  66$:
1600  006206  013746  177764          MOV     @#177764,-(SP)  ;;SAVE @#177764 FOR TYPEOUT
1601  006212  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1602  006214  104400  006222          TYPE    ,69$            ;;TYPE ASCIZ STRING
1603  006220  000406                  BR      68$             ;;GET OVER THE ASCIZ
1604                          ;;69$:   .ASCIZ  / (OCTAL)  /
1605  006236                  68$:
1606  006236  013746  177764          MOV     @#177764,-(SP)  ;;SAVE @#177764 FOR TYPEOUT
1607  006242  104410                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
1608  006244  104400  006252          TYPE    ,71$            ;;TYPE ASCIZ STRING
1609  006250  000406                  BR      70$             ;;GET OVER THE ASCIZ
1610                          ;;71$:   .ASCIZ  / (DECIMAL) /
1611  006266                  70$:
1612  006266  104400  001337          TYPE    ,$CRLF
1613                          ;;*****************************************************************
1614                          .SBTTL  SYSTEM SIZER
1615                          ;       THIS ROUTINE DETERMINES WHAT DRIVES ARE AVAILABLE ON
1616                          ;       THE FOLLOWING DEVICES: RK05, RP03, RP04, AND RS04. THE
1617                          ;       INFORMATION IS STORED IN THE TABLE "SYSSIZE" IN THE FOLLOWING FORMAT:
1618                          ;               A. EACH DEVICE IS ASSIGNED A WORD
1619                          ;               B. THE LOW BYTE OF THIS WORD INDICATES WHICH DRIVES ARE AVAILABLE
1620                          ;               C. THE HIGH BYTE INDICATES WHICH DRIVES HAVE BEEN USED
1621                          ;                  BY THE RELOCATION ROUTINE.
1622                          ;;*****************************************************************
1623  006272  012737  006404  000004  SIZE:   MOV     #21$,@#ERRVEC           ;SETUP TIMEOUT VECTOR
1624  006300  005037  001302          CLR     @#$TMP0                 ;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     @#$TMP0,@RKDA           ;SET UNIT NUMBER
1628  006320  012777  000015  173640          MOV     #15,@RKCS               ;SEND DRIVE RESET
1629  006326  032777  000200  173630          BIT     #BIT7,@RKER             ;NON EXISTANT DISK?
1630  006334  001011                  BNE     7$                      ;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              7$:     ROR     R0
1637  006362  012777  000001  173576      MOV     #1,@RKCS                ;CLEAR THE ERRORS
1638  006370  062737  020000  001302      ADD     #20000,@#$TMP0          ;SELECT NEXT UNIT
1639  006376  077133                      SOB     R1,9$                   ;CONTINUE
1640  006400  110037  001644              MOVB    R0,@#SYSSIZE+2          ;STORE IN TABLE
1641
1642                                      ;;**********************************************************************
1643                                      ;THIS CODE DETERMINES IF THERE IS AN RP03 OR AN RP04 OR BOTH.
1644                                      ;IF BOTH ARE ON THE SYSTEM, THE OPERATOR MUST CHANGE THE RP04
1645                                      ;ADDRESSES IN THE TABLE IN ''COMMON TAGS'' AND 'NOP'' THE BRANCH
1646                                      ;AT ''100$''.
1647
1648  006404  012737  007012  000004  21$:     MOV     #11$,@#ERRVEC           ;SET THE ERROR VECTOR
1649  006412  005737  176710                  TST     @#176710                ;IS THERE AN RP ON THE SYSTEM?
1650                                                                          ;STAY HERE IF YES
1651  006416  012737  006432  000004          MOV     #1$,@#ERRVEC
1652  006424  005777  173552                  TST     @RP4CS1         ;IS THERE AN RP04 ON SYSTEM?
1653  006430  000441                  100$:   BR      10$                     ;BRANCH IF YES
1654                                      ;;**********************************************************************
1655
1656                                      ;;**********************************************************************
1657  006432  012737  006534  000004  1$:     MOV     #10$,@#ERRVEC           ;SETUP TIMEOUT VEC FOR RP03 TEST
1658  006440  012737  000001  001302          MOV     #1,@#$TMP0              ;SETUP TEMP0
1659  006446  005000                          CLR     R0                      ;USED TO SET UNIT AVAILABLE BITS
1660  006450  012701  000010                  MOV     #10,R1                  ;SOB COUNT
1661  006454  013777  001302  173462  3$:     MOV     @#$TMP0,@RP3CS          ;SET FUNCTION IDLE WITH UNIT NO
1662  006462  005777  173456                  TST     @RP3CS                  ;WAS THERE AN ERROR?
1663  006466  100006                          BPL     6$                      ;BRANCH IF NO
1664  006470  006000                  4$:     ROR     R0                      ;UNIT NOT AVAILABLE
1665  006472  062737  000400  001302          ADD     #400,@#$TMP0            ;SELECT NEXT UNIT
1666  006500  077113                          SOB     R1,3$                   ;CONTINUE
1667  006502  000412                          BR      5$
1668  006504  017702  173430          6$:     MOV     @RP3DS,R2               ;GET STATUS REGISTER
1669  006510  042702  037777                  BIC     #37777,R2               ;GET BITS 14, 15 ONLY
1670  006514  022702  140000                  CMP     #140000,R2              ;IS DRIVE READY?
1671  006520  001363                          BNE     4$                      ;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$:     MOVB    R0,@#SYSSIZE            ;STORE IN TABLE
1675
1676                                      ;;**********************************************************************
1677  006534  012737  007012  000004  10$:     MOV     #11$,@#ERRVEC           ;SETUP ERROR VEC FOR RP04 TEST
1678  006542  005037  001302                  CLR     @#$TMP0
1679  006546  005000                          CLR     R0                      ;UNIT AVAILABLE WORD
1680  006550  012701  000010                  MOV     #10,R1                  ;SOB COUNT
1681  006554  113777  001302  173432  14$:     MOVB    @#$TMP0,@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     @#$TMP0,-(SP)   ;;SAVE @#$TMP0 FOR TYPEOUT
1693  006644  104402                          TYPOC                   ;;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          8$:     BIC     #163277,R2              ;GET BITS 12, 11, 8, & 6 ONLY
1700  006750  022702  010500                  CMP     #10500,R2               ;IS DRIVE READY?
1701  006754  001002                          BNE     12$                     ;BRANCH IF NO
1702  006756  052700  000400                  BIS     #BIT8,R0                ;SET UNIT AVAILABLE
1703  006762  006000                  12$:    ROR     R0
1704  006764  052777  000040  173222          BIS     #BIT5,@RP4CS2           ;CLEAR ERROR BITS
1705  006772  005237  001302                  INC     @#$TMP0                 ;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     @#$TMP0
1714  007024  005000                          CLR     R0
1715  007026  012701  000010                  MOV     #10,R1                  ;SOB COUNT
1716  007032  113777  001302  173214  18$:    MOVB    @#$TMP0,@RSCS2          ;SET UNIT NUMBER
1717  007040  012777  000001  173174          MOV     #1,@RSCS1               ;TRY NOP OPERATION
1718  007046  032777  010000  173200          BIT     #BIT12,@RSCS2           ;NON EXISTANT DRIVE?
1719  007054  001011                          BNE     16$                     ;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$:    ROR     R0
1726  007102  052777  000040  173144          BIS     #BIT5,@RSCS2            ;CLEAR ANY ERROR BITS
1727  007110  005237  001302                  INC     @#$TMP0                 ;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$:    MOVB    @#41,R0                 ;GET LOCATION 41
1737  007146  042700  177770                  BIC     #177770,R0              ;GET LEAST SIG 3 BITS
1738  007152  000241                          CLC                             ;ENSURE C CLEAR
1739  007154  006100                          ROL     R0                      ;ADJUST
1740  007156  122700  000002                  CMPB    #2,R0                   ;
1741  007162  002404                          BLT     40$                     ;BRANCH IF NO
1742  007164  042737  000001  001642          BIC     #BIT0,@#SYSSIZE
1743  007172  000403                          BR      20$
1744  007174  042760  000001  001646  40$:    BIC     #BIT0,SYSSIZE+4(R0)
```

```
1745  007202  005227  177777          20$:    INC     #-1
1746  007206  001057                          BNE     LOOP1           ;;BRANCH IF NOT FIRST TIME
1747  007210  104400  065475                  TYPE    ,MSG25
1748  007214  013746  001500                  MOV     @#OPT.CP,-(SP)
1749  007220  104402                          TYPOC
1750  007222  104400  001337                  TYPE    ,$CRLF
1751  007226  005737  001532                  TST     @#QV                    ;ACT11?
1752  007232  001045                          BNE     LOOP1                   ;BRANCH IF YES
1753  007234  105737  003571          50$:    TSTB    @#XXDPC                 ;XXDP CHAIN MODE?
1754  007240  001042                          BNE     LOOP1           ;;BRANCH IF YES
1755  007242  105737  001200                  TSTB    @#$PASS                 ;FIRST PASS?
1756  007246  001037                          BNE     LOOP1           ;;BRANCH IF NO
1757  007250  032737  000200  177570          BIT     #SW7,@#SWR              ;INHIBIT SIZE TYPEOUT?
1758  007256  001031                          BNE     SW8MSG          ;;BRANCH IF YES
1759  007260  004767  047462                  JSR     PC,TYPSIZ               ;GO TYPE SYSTEM SIZE
1760  007264  104400  007272                  TYPE    ,69$            ;;TYPE ASCIZ STRING
1761  007270  000417                          BR      68$             ;;GET OVER THE ASCIZ
1762                                  ;;69$:  .ASCIZ  /TYPE A CHARACTER TO CONTINUE/<CRLF>
1763  007330                          68$:
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                 .=10000
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     #$RTRN,(R0)+            ;SET T BIT VEC
1783  010044  042701  000340                  BIC     #PR7,R1
1784  010050  005020                          CLR     (R0)+                   ;SET TBIT VEC+2
1785  010052  005720                          TST     (R0)+                   ;BUMP R0 TO SCOPE VEC+2
1786  010054  005020                          CLR     (R0)+                   ;SET SCOPE VEC+2
1787  010056  062700  000006                  ADD     #6,R0                   ;SET R0 TO ERROR TRAP VEC
1788  010062  012720  000340                  MOV     #PR7,(R0)+              ;SET ERROR VEC
1789  010066  005720                          TST     (R0)+
1790  010070  012720  000340                  MOV     #PR7,(R0)+              ;SET TRAP VEC+2
1791  010074  012737  063464  000114          MOV     #.PARSRV,@#CACHVEC      ;SET PARITY ERROR VECTOR
1792  010102  052701  000340                  BIS     #PR7,R1
1793  010106  010137  000116                  MOV     R1,@#CACHVEC+2
1794  010112  012737  064122  000250          MOV     #KTABRT,@#MMVEC         ;SET KT11 ABORT VECTOR
1795  010120  010137  000252                  MOV     R1,@#MMVEC+2
1796  010124  042737  000340  177776          BIC     #PR7,@#PSW
1797                                  ;;**********************************************************************
1798                                  ;*TEST 1         MEMORY VERIFICATION TEST
1799                                  ;;**********************************************************************
1800  010132  012767  000001  171166          MOV     #1,$TIMES       ;;DO 1 ITERATION
```

```
 1801  010140  000004
 1802  010142  112737  000001  001202  TST1:   SCOPE
 1803  010150  013737  001202  177570          MOVB    #1,@#$TSTNM             ;LOAD TEST NUMBER
 1804                                          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
 1805
 1806                                          .SBTTL  START OF SECTION 0
 1807  010156  010700                  ;0000000000000 FIRST ADDRESS TO BE RELOCATED 000000000
 1808  010160  005740                  RELO:   MOV     PC,R0           ;GET PC
 1809  010162  010037  001540                  TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF RELO
 1810  010166  010700                          MOV     R0,@#FRSTAD     ;SAVE
 1811  010170  162700  010170                  MOV     PC,R0           ;GET CURRENT PC
 1812  010174  010037  001534                  SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
 1813  010200  010737  001212                  MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOP
 1814  010204  062737  000030  001212          MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
 1815  010212  013737  001212  001210          ADD     #30,@#$LPERR    ;ADJUST
 1816  010220  105737  001530                  MOV     @#$LPERR,@#$LPADR
 1817  010224  001402                          TSTB    @#NEXEC         ;BR IF TEST CODE TO BE EXECUTED
 1818  010226  000167  000720                  BEQ     .+6
 1819                                          JMP     RELE0
 1820  010232  000167  000714          ;MEMORY AND DISK (IF SELECTED) VERIFICATION TEST.
 1821  010236  177777  177777  177777        JMP     1$
 1822  010244  177777  000000  000000        .WORD   -1,-1,-1,-1,0,0,0,0
 1823  010252  000000  000000
 1824  010256  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1825  010264  177777  000000  000000
 1826  010272  000000  000000
 1827  010276  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1828  010304  177777  000000  000000
 1829  010312  000000  000000
 1830  010316  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1831  010324  177777  000000  000000
 1832  010332  000000  000000
 1833  010336  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1834  010344  177777  000000  000000
 1835  010352  000000  000000
 1836  010356  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1837  010364  177777  000000  000000
 1838  010372  000000  000000
 1839  010376  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1840  010404  177777  000000  000000
 1841  010412  000000  000000
 1842  010416  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1843  010424  177777  000000  000000
 1844  010432  000000  000000
 1845  010436  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1846  010444  177777  000000  000000
 1847  010452  000000  000000
 1848  010456  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1849  010464  177777  000000  000000
 1850  010472  000000  000000
 1851  010476  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1852  010504  177777  000000  000000
 1853  010512  000000  000000
 1854  010516  177777  177777  177777        .WORD   -1,-1,-1,-1,0,0,0,0
 1855  010524  177777  000000  000000
 1856  010532  000000  000000
```

```
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                          1$:
1908  011152  000004                  RELE0:  SCOPE
1909  011154  010702                          MOV     PC,R2
1910  011156  062702  000012                  ADD     #12,R2
1911  011162  012707  043764                  MOV     #RELOC,PC       ;GO RELOCATE PROGRAM CODE
1912  011166  000000                  RELO0:  .WORD   0
```

```
                                        ;0000000000000 LAST ADDRESS OF CODE TO BE RELOCATED 00000000000

                                        ;;*****************************************************************
                                        ;*TEST 2         CHECK BRANCH INSTRUCTIONS
                                        ;;*****************************************************************
011170  012767  000001  170130                MOV     #1,$TIMES       ;;DO 1 ITERATION
011176  000004                        TST2:   SCOPE
011200  112737  000002  001202                MOVB    #2,@#$TSTNM             ;LOAD TEST NUMBER
011206  013737  001202  177570                MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER

                                                .SBTTL  START OF SECTION 1
                                        ;1111111111111 FIRST ADDRESS TO BE RELOCATED 111111111
011214  010700                        REL1:   MOV     PC,R0           ;GET PC
011216  005740                                TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF REL1
011220  010037  001540                        MOV     R0,@#FRSTAD     ;SAVE
011224  010700                                MOV     PC,R0           ;GET CURRENT PC
011226  162700  011226                        SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
011232  010037  001534                        MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOR
011236  010737  001212                        MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
011242  062737  000030  001212                ADD     #30,@#$LPERR    ;ADJUST
011250  013737  001212  001210                MOV     @#$LPERR,@#$LPADR
011256  105737  001530                        TSTB    @#NEXEC         ;BR IF TEST CODE TO BE EXECUTED
011262  001402                                BEQ     .+6
011264  000167  004146                        JMP     RELE1
                                        ;
011270  000257                                CCC                     ;CC'S=0000
011272  103407                                BCS     CC0             ;SAME AS BLO
011274  102406                                BVS     CC0
011276  001405                                BEQ     CC0
011300  100404                                BMI     CC0
011302  002403                                BLT     CC0
011304  003402                                BLE     CC0
011306  101401                                BLOS    CC0
011310  101001                                BHI     .+4
011312  104000                        CC0:    HLT                     ;ONE OF THE ABOVE BRANCHES FAILED

                                        ;CONTINUE
011314  000270                                SEN                     ;CC'S=1000
011316  100003                                BPL     CC1
011320  002002                                BGE     CC1
011322  003001                                BGT     CC1
011324  002401                                BLT     .+4
011326  104000                        CC1:    HLT                     ;ONE OF THE ABOVE BRANCHES FAILED

                                        ;CONTINUE
011330  000262                                SEV                     ;CC'S=1010
011332  102003                                BVC     CC2
011334  002402                                BLT     CC2
011336  003401                                BLE     CC2
011340  002001                                BGE     .+4
011342  104000                        CC2:    HLT                     ;ERROR! ONE OF THE ABOVE BRANCHES FAILED

                                        ;CONTINUE
011344  000261                                SEC                     ;CC'S=1011
011346  103002                                BCC     CC3
011350  101001                                BHI     CC3
```

```
1969  011352  003001                          BGT     .+4
1970  011354  104000                  CC3:    HLT                     ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
1971
1972                                  ;CONTINUE
1973  011356  000264                          SEZ                     ;CC'S=1111
1974  011360  001003                          BNE     CC4
1975  011362  003002                          BGT     CC4
1976  011364  101001                          BHI     CC4
1977  011366  003401                          BLE     .+4
1978  011370  104000                  CC4:    HLT                     ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
1979                                  ;;*********************************************************************
1980                                  ;*TEST 3         TEST UNIARY CONDITION CODES
1981                                  ;;*********************************************************************
1982  011372  000004                  TST3:   SCOPE
1983  011374  112737  000003  001202          MOVB    #3,@#$TSTNM             ;LOAD TEST NUMBER
1984  011402  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
1985                                  ;CLR    R0
1986  011410  000277                          SCC
1987  011412  000244                          CLZ
1988  011414  005000                          CLR     R0                      ;R0=0,CC'S=0100
1989  011416  103404                          BCS     CLR0
1990  011420  102403                          BVS     CLR0
1991  011422  001002                          BNE     CLR0
1992  011424  100401                          BMI     CLR0
1993  011426  003401                          BLE     .+4
1994  011430  104000                  CLR0:   HLT                     ;ERROR! INCORRECT CC'S AFTER CLR
1995
1996  011432  000277                          SCC
1997  011434  000244                          CLZ
1998  011436  005700                          TST     R0                      ;R0=0,CC'S=0100
1999  011440  103404                          BCS     TST0
2000  011442  102403                          BVS     TST0
2001  011444  001002                          BNE     TST0
2002  011446  100401                          BMI     TST0
2003  011450  101401                          BLOS    .+4
2004  011452  104000                  TST0:   HLT                     ;ERROR! INCORRECT CC'S AFTER TST
2005
2006  011454  000257                          CCC
2007  011456  000266                          +SEZ!SEV
2008  011460  005100                          COM     R0              ;R0=-1,CC'S=1001
2009  011462  103004                          BCC     COM0
2010  011464  102403                          BVS     COM0
2011  011466  001402                          BEQ     COM0
2012  011470  100001                          BPL     COM0
2013  011472  002401                          BLT     .+4
2014  011474  104000                  COM0:   HLT                     ;ERROR! INCORRECT CC'S AFTER COM
2015
2016  011476  000261                          SEC
2017  011500  005500                          ADC     R0              ;R0=000000,CC'S=0101
2018  011502  103003                          BCC     ADC0
2019  011504  102402                          BVS     ADC0
2020  011506  001001                          BNE     ADC0
2021  011510  002001                          BGE     .+4
2022  011512  104000                  ADC0:   HLT                     ;ERROR! INCORRECT CC'S AFTER ADC
2023
2024  011514  000261                          SEC
```

```
2025  011516  006000                    ROR     R0              ;R0=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     R0              ;R0=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     R0              ;R0=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     R0              ;R0=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     R0              ;R0=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     R0              ;R0=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     R0              ;R0=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                                  ;*TEST 4         CHECK REGISTER SELECTION
2098                                  ;;*******************************************************************
2099  011716  000004                  TST4:   SCOPE
2100  011720  112737  000004  001202          MOVB    #4,@#$TSTNM             ;LOAD TEST NUMBER
2101  011726  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
2102  011734  012737  000005  001326          MOV     #5,@#$TIMES     ;SET ITERATION COUNT TO 5
2103  011742  005000                          CLR     R0
2104  011744  000277                          SCC
2105  011746  006100                          ROL     R0              ;R0=1
2106  011750  010002                          MOV     R0,R2
2107  011752  006302                          ASL     R2              ;R2=2
2108  011754  010203                          MOV     R2,R3
2109  011756  006303                          ASL     R3              ;R3=4
2110  011760  010304                          MOV     R3,R4
2111  011762  006304                          ASL     R4              ;R4=10
2112  011764  010405                          MOV     R4,R5
2113  011766  006305                          ASL     R5              ;R5=20
2114  011770  010546                          MOV     R5,-(SP)        ;SET BITS SET IN REGISTERS
2115  011772  050416                          BIS     R4,(SP)         ;INTO STACK ADDRESS
2116  011774  050316                          BIS     R3,(SP)
2117  011776  050216                          BIS     R2,(SP)
2118  012000  050016                          BIS     R0,(SP)
2119  012002  022726  000037                  CMP     #37,(SP)+
2120  012006  001401                          BEQ     .+4             ;WERE SET
2121  012010  104000                          HLT                     ;MISSING BIT(S) REPRESENT
2122                                                                  ;INCORRECT REGISTER SELECTION
2123
2124                                  ;CHECK THAT ALL BITS CAN BE SET & CLEARED IN ALL REGISTERS
2125  012012  000257                          CCC
2126  012014  112700  000377                  MOVB    #377,R0         ;SET ALL BITS (MOVB EXTENDS SIGN)
2127  012020  006100                  1$:     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$:     HLT
2181
2182                            ;CHECK TRANSFER OF REGISTER DATA BETWEEN THE GS AND GD REGISTERS
2183  012170  032737  000020  177776  GSTST:  BIT     #20,@#PSW       ;CHECK IF 'T' BIT IS SET
2184  012176  001050                    BNE     7$              ;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                  4$:     .WORD   0               ;CONTAINS GS STACK PTR
2201  012246  016706  177730                  MOV     1$,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                  5$:     MOV     R0,-(SP)        ;GET GS REG 0
2209  012300  010146                          MOV     R1,-(SP)        ;ETC...
2210  012302  010246                          MOV     R2,-(SP)
2211  012304  010346                          MOV     R3,-(SP)
2212  012306  010446                          MOV     R4,-(SP)
2213  012310  010546                          MOV     R5,-(SP)
2214  012312  104000                          HLT                     ;ERROR! DATA IN GS STK PTR NOT = GS REG 0
2215                                                                  ;GS REG 0-GS REG 5 ARE ON THE STACK
2216  012314  016706  177662                  MOV     1$,SP           ;RESTORE STACK PTR
2217  012320                          6$:
2218  012320                          7$:
2219                                  ;;*****************************************************************
2220                                  ;*TEST 5         TEST UNIARY WORD INSTRUCTIONS USING ADDRESS MODE 1
2221                                  ;;*****************************************************************
2222  012320  000004                  TST5:   SCOPE
2223  012322  112737  000005  001202          MOVB    #5,@#$TSTNM             ;LOAD TEST NUMBER
2224  012330  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
2225  012336  012737  000005  001326          MOV     #5,@#$TIMES
2226  012344  000401                          BR      .+4
2227  012346  000000                          .WORD   0               ;RESERVE ADDRESS FOR TESTS
2228  012350  010702                          MOV     PC,R2
2229  012352  162702  000004                  SUB     #4,R2           ;R2 POINTS TO RESERVED WORD
2230  012356  005012                          CLR     (R2)            ;PRESET (R2)
2231
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 SHOEN 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                            ;*TEST 6         CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1
2345                            ;;**********************************************************************
2346  012660  000004            TST6:   SCOPE
2347  012662  112737  000006  001202    MOVB    #6,@#$TSTNM             ;LOAD TEST NUMBER
2348  012670  013737  001202  177570    MOV     @#$TSTNM,@#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            1$:     SEC
2358  012720  105513                    ADCB    (R3)            ;ADD CARRY TO EVEN BYTE
2359  012722  100402                    BMI     2$              ;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)=000400,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                                ;*TEST 7         CHECK UNIARY WORD OPS USING ADDRESS MODES 2 & 4
2496                                ;;*****************************************************************
2497  013274  000004                TST7:   SCOPE
2498  013276  112737  000007  001202       MOVB    #7,@#$TSTNM             ;LOAD TEST NUMBER
2499  013304  013737  001202  177570       MOV     @#$TSTNM,@#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)=000200,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                                ;*TEST 10       CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4
2599                                ;;*****************************************************************
2600  013564  000004                TST10:  SCOPE
2601  013566  112737  000010  001202        MOVB    #10,@#$TSTNM            ;LOAD TEST NUMBER
2602  013574  013737  001202  177570        MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
2603  013602  000401                        BR      .+4             ;RESERVE A WORD
2604  013604  000000                        .WORD   0               ;RESERVED WORD
2605  013606  010705                        MOV     PC,R5
2606  013610  162705  000004                SUB     #4,R5           ;R5 POINTS TO EVEN BYTE OF RESERVED WORD
2607  013614  010500                        MOV     R5,R0
2608  013616  010002                        MOV     R0,R2
2609  013620  005202                        INC     R2              ;R2 POINTS TO ODD BYTE OF RESERVED WORD
2610  013622  005010                        CLR     (R0)            ;PRESET
2611
2612  013624  000277                        SCC
2613  013626  000241                        CLC
2614  013630  105125                        COMB    (R5)+           ;(R0)=000377,CC=1001
2615  013632  103002                        BCC     COMB2
2616  013634  102401                        BVS     COMB2
2617  013636  100401                        BMI     .+4
2618  013640  104000                COMB2:  HLT                     ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2619
2620  013642  105542                        ADCB    -(R2)           ;(R0)=000000,CC=0101
2621  013644  001401                        BEQ     .+4
2622  013646  104000                        HLT                     ;ERROR! INCORRECT RESULT AS SHOWN ABOVE
2623  013650  105525                        ADCB    (R5)+           ;(R0)=000400,CC=0000
2624  013652  103401                        BCS     ADCB2
2625  013654  001001                        BNE     .+4
2626  013656  104000                ADCB2:  HLT                     ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2627
2628  013660  000263                        +SEC!SEV
2629  013662  106045                        RORB    -(R5)           ;(R0)=100000,CC=1001
2630  013664  103003                        BCC     RORB4
2631  013666  102402                        BVS     RORB4
2632  013670  001401                        BEQ     RORB4
2633  013672  100401                        BMI     .+4
2634  013674  104000                RORB4:  HLT                     ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2635
2636  013676  000277                        SCC
2637  013700  106122                        ROLB    (R2)+           ;(R0)=100001,CC=0000
2638  013702  103403                        BCS     ROLB2
2639  013704  102402                        BVS     ROLB2
2640  013706  001401                        BEQ     ROLB2
```

```
2641  013710  100001                    BPL     .+4
2642  013712  104000          ROLB2:    HLT                     ;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                     ;ERROR! INCORRECT CC'S AS SHOWNABOVE
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                    BCS     SWAB4
 2698  014050  100401                    BMI     .+4
 2699  014052  104000            SWAB4:  HLT
 2700
 2701  014054  000277                    SCC
 2702  014056  105225                    INCB    (R5)+           ;(R0)=000601=[0004][201],CC=0000
 2703  014060  103003                    BCC     INCB2
 2704  014062  102402                    BVS     INCB2
 2705  014064  001401                    BEQ     INCB2
 2706  014066  100001                    BPL     .+4
```

```
  2707  014070  104000                INCB2:  HLT
```

```
2708
```

```
2709  014072  022227  000601                CMP     (R2)+,#000601   ;CHECK END RESULT
2710  014076  001401                        BEQ     .+4
2711  014100  104000                        HLT
2712  014102  020205                        CMP     R2,R5           ;CHECK REGISTERS
2713  014104  001401                        BEQ     .+4
2714  014106  104000                        HLT
2715                                ;;*****************************************************************
2716                                ;*TEST 11        CHECK UNIARY WORD OPS USING ADDRESS MODES 3 & 5
2717                                ;;*****************************************************************
2718  014110  000004                TST11:  SCOPE
2719  014112  112737  000011  001202        MOVB    #11,@#$TSTNM            ;LOAD TEST NUMBER
2720  014120  013737  001202  177570        MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
2721  014126  000402                        BR      .+6             ;RESERVE 2 WORDS
2722  014130  000000                        .WORD   0               ;1 FOR THE ADDRESS
2723  014132  000000                        .WORD   0               ;AND 1 FOR DATA
2724  014134  010703                        MOV     PC,R3
2725  014136  162703  000004              SUB     #4,R3
2726  014142  005013                        CLR     (R3)            ;PRESET DATA
2727  014144  010300                        MOV     R3,R0           ;R0 POINTS TO DATA WORD
2728  014146  005743                        TST     -(R3)
2729  014150  010013                        MOV     R0,(R3)
2730  014152  010304                        MOV     R3,R4
2731
2732  014154  000257                        CCC
2733  014156  005733                        TST     @(R3)+          ;(R0)=000000,CC=0100
2734  014160  001401                        BEQ     .+4
2735  014162  104000                        HLT
2736
2737  014164  000261                        SEC
2738  014166  006053                        ROR     @-(R3)          ;(R0)=100000,CC=1010
2739  014170  103402                        BCS     ROR5
2740  014172  102001                        BVC     ROR5
2741  014174  100401                        BMI     .+4
2742  014176  104000                ROR5:   HLT
2743
2744  014200  000257                        CCC
2745  014202  006234                        ASR     @(R4)+          ;(R0)=140000,CC=1010
2746  014204  102001                        BVC     ASR3
2747  014206  100401                        BMI     .+4
2748  014210  104000                ASR3:   HLT
2749
2750  014212  000250                        CLN
2751  014214  006333                        ASL     @(R3)+          ;(R0)=100000,CC=1001
2752  014216  103002                        BCC     ASL3
2753  014220  102401                        BVS     ASL3
2754  014222  100401                        BMI     .+4
2755  014224  104000                ASL3:   HLT
2756
2757  014226  000277                        SCC
2758  014230  005354                        DEC     @-(R4)          ;(R0)=077777, CC=0010
2759  014232  103003                        BCC     DEC5
2760  014234  102002                        BVC     DEC5
2761  014236  001401                        BEQ     DEC5
2762  014240  100001                        BPL     .+4
2763  014242  104000                DEC5:   HLT
2764
```

```
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
2800                            ;;**********************************************************************
2801                            ;*TEST 12        CHECK UNIARY BYTE OPS USING ADDRESS MODES 3 & 5
2802                            ;;**********************************************************************
2803  014340  000004            TST12:  SCOPE
2804  014342  112737  000012  001202    MOVB    #12,@#$TSTNM            ;LOAD TEST NUMBER
2805  014350  013737  001202  177570    MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
2806  014356  000403                    BR      .+10                    ;RESERVE 3 WORDS
2807  014360  000000                    .WORD   0                       ;1 FOR EVEN BYTE ADDRESS
2808  014362  000000                    .WORD   0                       ;1 FOR ODD BYTE ADDRESS
2809  014364  000000                    .WORD   0                       ;AND 1 FOR DATA
2810  014366  010702                    MOV     PC,R2
2811  014370  005742                    TST     -(R2)                   ;BACK R2 UP TO
2812  014372  005742                    TST     -(R2)                   ;DATA WORD
2813  014374  010200                    MOV     R2,R0                   ;R0 POINTS TO THE DATA WORD
2814  014376  005010                    CLR     (R0)                    ;PRESET DATA
2815  014400  005742                    TST     -(R2)                   ;BACK R2 UP TO
2816  014402  005742                    TST     -(R2)                   ;EVEN BYTE ADDRESS WORD
2817  014404  010022                    MOV     R0,(R2)+                ;LOAD ADDRESS
2818  014406  005200                    INC     R0                      ;CDD BYTE ADDRESS
2819  014410  010022                    MOV     R0,(R2)+                ;LOAD ODD BYTE ADDRESS
2820  014412  010200                    MOV     R2,R0                   ;RESET R0
```

```
 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                    BCS     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           ;BRANCH IF C OR Z IS SET
 2863  014526  102401                    BVS     ROLB3
 2864  014530  100001                    BPL     .+4
 2865  014532  104000            ROLB3:  HLT
 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                    BCS     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                                ;*TEST 13        CHECK UNIARY WORD OPS USING ADDRESS MODE 6 (PC)
2884                                ;;*****************************************************************
2885  014566  000004                TST13:  SCOPE
2886  014570  112737  000013  001202       MOVB    #13,@#$TSTNM            ;LOAD TEST NUMBER
2887  014576  013737  001202  177570       MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
2888  014604  005027                        CLR     (PC)+           ;PRESET DATA = 0
2889  014606  000000                UWM6:   .WORD   0               ;RESERVED FOR DATA
2890  014610  010700                        MOV     PC,R0
2891  014612  024040                        CMP     -(R0),-(R0)     ;R0 POINTS TO DATA WORD
2892  014614  000277                        SCC
2893  014616  006167  177764               ROL     UWM6            ;(R0)=000001,CC=0000
2894  014622  103403                        BCS     ROL6
2895  014624  102402                        BVS     ROL6
2896  014626  001401                        BEQ     ROL6
2897  014630  100001                        BPL     .+4
2898  014632  104000                ROL6:   HLT
2899
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  103002                        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
2967                        ;;**************************************************************************
2968                        ;*TEST 14        CHECK UNIARY BYTE OPS (EVEN/ODD) USING ADDRESS MODE 6 (PC)
2969                        ;;**************************************************************************
2970  015054  000004        TST14:  SCOPE
2971  015056  112737  000014  001202        MOVB    #14,@#$TSTNM            ;LOAD TEST NUMBER
2972  015064  013737  001202  177570        MOV     @#$TSTNM,@#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     UBM6            ;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
```

```
2989
2990  015142  105667  000266          SBCB    UBM6            ;(R0)=000000, CC=0100
2991  015146  103402                  BCS     SBCB6
2992  015150  102401                  BVS     SBCB6
2993  015152  001401                  BEQ     .+4
2994  015154  104000          SBCB6:  HLT
2995
2996  015156  000261          1$:     SEC
2997  015160  105267  000250          INCB    UBM6            ;LOOP UNTIL (R0)=077600, CC=1011
2998  015164  100403                  BMI     2$
2999  015166  105567  000243          ADCB    UBM6+1          ;INCB INST INCREMENTS EVEN BYTE
3000  015172  000771                  BR      1$              ;ADCB INCREMENTS ODD BYTE
3001  015174  103001          2$:     BCC     INCB6
3002  015176  102401                  BVS     .+4
3003  015200  104000          INCB6:  HLT
3004
3005  015202  106367  000226          ASLB    UBM6            ;(R0)=077400, CC=0111
3006  015206  103003                  BCC     ASLB6
3007  015210  102002                  BVC     ASLB6
3008  015212  001001                  BNE     ASLB6
3009  015214  100001                  BPL     .+4
3010  015216  104000          ASLB6:  HLT
3011
3012  015220  000242                  CLV
3013  015222  105567  000207          ADCB    UBM6+1          ;(R0)=100000, CC=1010
3014  015226  103402                  BCS     ADCB6
3015  015230  102001                  BVC     ADCB6
3016  015232  100401                  BMI     .+4
3017  015234  104000          ADCB6:  HLT
3018
3019  015236  000261                  SEC
3020  015240  106067  000171          RORB    UBM6+1          ;(R0)=140000, CC=1010
3021  015244  103402                  BCS     RORB6
3022  015246  102001                  BVC     RORB6
3023  015250  100401                  BMI     .+4
3024  015252  104000          RORB6:  HLT
3025
3026  015254  105167  000154          COMB    UBM6            ;(R0)=140377 CC=1001
3027  015260  103002                  BCC     COMB6
3028  015262  102401                  BVS     COMB6
3029  015264  100401                  BMI     .+4
3030  015266  104000          COMB6:  HLT
3031
3032  015270  000262                  SEV
3033  015272  105467  000137          NEGB    UBM6+1          ;(R0)=040377, CC=0001
3034  015276  103002                  BCC     NEGB6
3035  015300  102401                  BVS     NEGB6
3036  015302  100001                  BPL     .+4
3037  015304  104000          NEGB6:  HLT
3038
3039  015306  106167  000123          ROLB    UBM6+1          ;(R0)=100777, CC=1010
3040  015312  103402                  BCS     ROLB6
3041  015314  102001                  BVC     ROLB6
3042  015316  100401                  BMI     .+4
3043  015320  104000          ROLB6:  HLT
3044
```

```
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                          ;1111111111111 LAST ADDRESS OF CODE TO BE RELOCATED 11111111111
3089
3090                          ;;*****************************************************************
3091                          ;*TEST 15        CHECK UNIARY WORD OPS USING ADDRESS MODE 7
3092                          ;;*****************************************************************
3093  015454  012767  000001  163644          MOV     #1,$TIMES       ;;DO 1 ITERATION
3094  015462  000004          TST15:  SCOPE
3095  015464  112737  000015  001202          MOVB    #15,@#$TSTNM            ;LOAD TEST NUMBER
3096  015472  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
3097
3098                                  .SBTTL  START OF SECTION 2
3099                          ;2222222222222 FIRST ADDRESS TO BE RELOCATED 222222222
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,@#$LPERR     ;SET LOOP ADDRESS
3107  015526  062737  000030  001212          ADD     #30,@#$LPERR    ;ADJUST
3108  015534  013737  001212  001210          MOV     @#$LPERR,@#$LPADR
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 UWM7
3114  015560  000000                  UWM7:   .WORD   0               ;CONTAINS DATA
3115  015562  000000                          .WORD   0               ;CONTAINS ADDRESS OF UWM7
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                          BCC     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                          ;;**********************************************************************
```

```
                                ;*TEST 16       CHECK UNIARY BYTE OPS USING ADDRESS MODE 7
                                ;;*****************************************************************
016064  000004                  TST16:  SCOPE
016066  112737  000016  001202          MOVB    #16,@#$TSTNM            ;LOAD TEST NUMBER
016074  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
016102  012700  015560                  MOV     #UWM7,R0
016106  063700  001534                  ADD     @#FACTOR,R0
016112  010002                          MOV     R0,R2
016114  010067  177442                  MOV     R0,UWM7+2
016120  005720                          TST     (R0)+
016122  005210                          INC     (R0)            ;WORD FOLLOWING UWM7 CONTAINS ADDRESS
016124  005740                          TST     -(R0)           ;OF ODD BYTE, R0 POINTS TO DATA WORD
016126  005010                          CLR     (R0)            ;PRESET DATA
016130  010067  177422                  MOV     R0,UWM7-2
                                ;NOTE:  @2(2) REFERENCES THE ODD BYTE, AND @-2(2) REFERENCES THE EVEN BYTE.

016134  000263                          +SEC!SEV                ;SET C AND V
016136  105672  000002                  SBCB    @2(2)           ;(R0)=177400, CC=1001
016142  103003                          BCC     SBCB7
016144  102402                          BVS     SBCB7
016146  001401                          BEQ     SBCB7
016150  100401                          BMI     .+4
016152  104000                  SBCB7:  HLT

016154  000277                          SCC                     ;SET CONDITION CODES
016156  105572  177776                  ADCB    @-2(2)          ;(R0)=177401, CC=0000
016162  103403                          BCS     ADCB7
016164  102402                          BVS     ADCB7
016166  001401                          BEQ     ADCB7
016170  100001                          BPL     .+4
016172  104000                  ADCB7:  HLT

016174  105172  177776                  COMB    @-2(2)          ;(R0)=177776, CC=1001
016200  103002                          BCC     COMB7
016202  102401                          BVS     COMB7
016204  100401                          BMI     .+4
016206  104000                  COMB7:  HLT

016210  000241                          CLC                     ;CLEAR CARRY
016212  106072  000002                  RORB    @2(2)           ;(R0)=077776, CC=0011
016216  103002                          BCC     RORB7
016220  102001                          BVC     RORB7
016222  100001                          BPL     .+4
016224  104000                  RORB7:  HLT

016226  105272  000002                  INCB    @2(2)           ;(R0)=100376, CC=1011
016232  103002                          BCC     INCB7
016234  102001                          BVC     INCB7
016236  100401                          BMI     .+4
016240  104000                  INCB7:  HLT

016242  105372  177776                  DECB    @-2(2)          ;(R0)=100375, CC=1001
016246  103002                          BCC     DECB7
016250  102401                          BVS     DECB7
016252  100401                          BMI     .+4
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                     ;CLEAR CARRY
3277  016274  106272  177776          ASRB    @-2(2)          ;(R0)=000376, CC=1001
3278  016300  103002                  BCC     ASRB7
3279  016302  102401                  BVS     ASRB7
3280  016304  100401                  BMI     .+4
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     .+4
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                          ;*TEST 17        CHECK BINARY OPS USING ADDRESS MODE 0
3304                          ;;**********************************************************************
3305  016366  000004          TST17:  SCOPE
3306  016370  112737  000017  001202  MOVB    #17,@#$TSTNM            ;LOAD TEST NUMBER
3307  016376  013737  001202  177570  MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
3308  016404  000277                  SCC                     ;SET CONDITION CODES
3309  016406  010700                  MOV     PC,R0           ;R0=PC, CC=X001
3310  016410  103002                  BCC     MOV0
3311  016412  102401                  BVS     MOV0
3312  016414  001001                  BNE     .+4
3313  016416  104000          MOV0:   HLT
3314
3315  016420  010002                  MOV     R0,R2           ;R2=R0
3316  016422  000262                  SEV                     ;SET V
3317  016424  160002                  SUB     R0,R2           ;R2=000000, CC=0100
3318  016426  103402                  BCS     SUB0
3319  016430  102401                  BVS     SUB0
3320  016432  001401                  BEQ     .+4
3321  016434  104000          SUB0:   HLT
3322
3323  016436  000244                  CLZ
3324  016440  010203                  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     CMP0
3333  016460  102402                BVS     CMP0
3334  016462  001001                BNE     CMP0
3335  016464  100001                BPL     .+4
3336  016466  104000        CMP0:   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                BEQ     .+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     BIS0
3370  016560  102401                BVS     BIS0
3371  016562  001001                BNE     .+4
3372  016564  104000        BIS0:   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     BIC0
3379  016600  102402                BVS     BIC0
3380  016602  001001                BNE     BIC0
```

```
3381  016604  100001                  BPL     .+4
3382  016606  104000            BICO: HLT
3383
3384  016610  010004                  MOV     R0,R4
3385  016612  005104                  COM     R4
3386  016614  040004                  BIC     R0,R4           ;R0=COMPLEMENT OF R4, R4 REMAINS UNCHANGED
3387  016616  005104                  COM     R4
3388  016620  020004                  CMP     R0,R4           ;R0=R4
3389  016622  001401                  BEQ     .+4
3390  016624  104000                  HLT
3391
3392  016626  010004                  MOV     R0,R4
3393  016630  005104                  COM     R4
3394  016632  010403                  MOV     R4,R3           ;R3=R4
3395  016634  050003                  BIS     R0,R3           ;R3=COMPLEMENT OF R0, CC=1001
3396  016636  103001                  BCC     BISOA
3397  016640  100401                  BMI     .+4
3398  016642  104000            BISOA: HLT
3399  016644  005203                  INC     R3              ;R3=0 AFTER INC
3400  016646  001401                  BEQ     .+4
3401  016650  104000                  HLT
3402  016652  010304                  MOV     R3,R4           ;R3=R4=0
3403  016654  005103                  COM     R3              ;R3=177777
3404  016656  000261                  SEC                     ;SET C
3405  016660  006004                  ROR     R4              ;R4=100000
3406  016662  060304                  ADD     R3,R4           ;R3=177777,R4=077777, CC=0011
3407  016664  103003                  BCC     ADDO
3408  016666  102002                  BVC     ADDO
3409  016670  001401                  BEQ     ADDO
3410  016672  100001                  BPL     .+4
3411  016674  104000            ADDO: HLT
3412  016676  010700                  MOV     PC,R0           ;R0=PC
3413  016700  022020                  CMP     (R0)+,(R0)+     ;R0=R0+4
3414  016702  020007                  CMP     R0,PC           ;PC=PC+4=R0
3415  016704  001401                  BEQ     .+4
3416  016706  104000                  HLT
3417
3418  016710  010700                  MOV     PC,R0           ;R0=PC
3419  016712  062700  000010          ADD     #10,R0          ;R0=PC+10(8)
3420  016716  010002                  MOV     R0,R2           ;R2=R0
3421  016720  020700                  CMP     PC,R0           ;R0=PC
3422  016722  001002                  BNE     CMPOA
3423  016724  020200                  CMP     R2,R0           ;R2=R0
3424  016726  001401                  BEQ     .+4
3425  016730  104000            CMPOA: HLT
3426                            ;;**********************************************************************
3427                            ;*TEST 20        CHECK BINARY OPS USING ADDRESS MODE 1
3428                            ;;**********************************************************************
3429  016732  000004            TST20: SCOPE
3430  016734  112737  000020  001202  MOVB    #20,@#$TSTNM            ;LOAD TEST NUMBER
3431  016742  013737  001202  177570  MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
3432  016750  000402                  BR      .+6             ;RESERVE TWO WORDS
3433  016752  000000                  .WORD   0               ;RESERVED FOR SOURCE DATA
3434  016754  000000                  .WORD   0               ;RESERVED FOR DESTINATION DATA
3435  016756  010704                  MOV     PC,R4
3436  016760  005744                  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 SOUCE 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                  BMI     .+4
3494  017130  104000          BIC1:   HLT
3495
3496  017132  000262                  SEV                     ;SET V
3497  017134  021314                  CMP     (R3),(R4)       ;(R3)=000377,(R4)=177400, CC=0001
3498  017136  103003                  BCC     CMP1A
3499  017140  102402                  BVS     CMP1A
3500  017142  001401                  BEQ     CMP1A
3501  017144  100001                  BPL     .+4
3502  017146  104000          CMP1A:  HLT
3503
3504  017150  005013                  CLR     (R3)            ;(R3)=000000
3505  017152  000261                  SEC
3506  017154  006013                  ROR     (R3)            ;(R3)=100000
3507  017156  011314                  MOV     (R3),(R4)       ;(R3)=(R4)=100000
3508  017160  005114                  COM     (R4)            ;(R4)=077777
3509  017162  161314                  SUB     (R3),(R4)       ;(R3)=100000,(R4)=177777, CC=1011
3510  017164  103002                  BCC     SUB1A
3511  017166  102001                  BVC     SUB1A
3512  017170  100401                  BMI     .+4
3513  017172  104000          SUB1A:  HLT
3514
3515  017174  000277                  SCC
3516  017176  161314                  SUB     (R3),(R4)       ;(R3)=100000,(R4)=077777, CC=0000
3517  017200  101402                  BLOS    SUB1B           ;BRANCH IF C OR Z IS SET
3518  017202  102401                  BVS     SUB1B
3519  017204  100001                  BPL     .+4
3520  017206  104000          SUB1B:  HLT
3521
3522  017210  011314                  MOV     (R3),(R4)       ;(R3)=100000,(R4)=100000, CC=1000
3523  017212  001401                  BEQ     MOV1
3524  017214  100401                  BMI     .+4
3525  017216  104000          MOV1:   HLT
3526
3527  017220  061314                  ADD     (R3),(R4)       ;(R3)=100000,(R4)=000000, CC=0111
3528  017222  103003                  BCC     ADD1A
3529  017224  102002                  BVC     ADD1A
3530  017226  001001                  BNE     ADD1A
3531  017230  100001                  BPL     .+4
3532  017232  104000          ADD1A:  HLT
3533
3534  017234  005113                  COM     (R3)            ;(R3)=077777
3535  017236  011314                  MOV     (R3),(R4)       ;(R4)=077777
3536  017240  061314                  ADD     (R3),(R4)       ;(R3)=077777,(R4)=177776, CC=1010
3537  017242  103402                  BCS     ADD1B
3538  017244  102001                  BVC     ADD1B
3539  017246  100401                  BMI     .+4
3540  017250  104000          ADD1B:  HLT
3541
3542  017252  062714  000002          ADD     #2,(R4)
3543  017256  005714                  TST     (R4)            ;CHECK FINAL RESULT
3544  017260  001401                  BEQ     .+4
3545  017262  104000                  HLT
3546                          ;;*****************************************************************
3547                          ;*TEST 21        CHECK BINARY BYTE OPS USING ADDRESS MODE 1
3548                          ;;*****************************************************************
```

```
3549  017264  000004                  TST21:  SCOPE
3550  017266  112737  000021  001202          MOVB    #21,@#$TSTNM            ;LOAD TEST NUMBER
3551  017274  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
3552  017302  000402                          BR      .+6
3553  017304  000000                          .WORD   0
3554  017306  000000                          .WORD   0
3555  017310  010705                          MOV     PC,R5
3556  017312  005745                          TST     -(R5)
3557  017314  005045                          CLR     -(R5)           ;(R5)=000000
3558  017316  010502                          MOV     R5,R2
3559  017320  005042                          CLR     -(R2)           ;(R2)=000000
3560  017322  005202                          INC     R2              ;R2 POINTS TO ODD BYTE
3561  017324  105112                          COMB    (R2)            ;(R2)=177400
3562
3563  017326  000277                          SCC
3564  017330  111215                          MOVB    (R2),(R5)       ;(R2)=177400,(R5)=000377,CC=1001
3565  017332  103005                          BCC     MOVB1
3566  017334  102404                          BVS     MOVB1
3567  017336  001403                          BEQ     MOVB1
3568  017340  100002                          BPL     MOVB1
3569  017342  105215                          INCB    (R5)            ;CHECK RESULT
3570  017344  001401                          BEQ     .+4
3571  017346  104000                  MOVB1:  HLT
3572
3573  017350  106312                          ASLB    (R2)            ;SHIFT (R2) UNTIL
3574  017352  102376                          BVC     .-2             ;(R2)=000000
3575  017354  106012                          RORB    (R2)            ;(R2)=100000
3576  017356  105315                          DECB    (R5)            ;(R5)=00377
3577  017360  106015                          RORB    (R5)            ;(R5)=000177
3578  017362  000257                          CCC
3579  017364  121512                          CMPB    (R5),(R2)       ;(R5)=000177,(R2)=100000, CC=1010
3580  017366  102001                          BVC     CMPB1
3581  017370  100401                          BMI     .+4
3582  017372  104000                  CMPB1:  HLT
3583
3584  017374  005003                          CLR     R3
3585  017376  000261                          SEC
3586  017400  006003                          ROR     R3              ;R3=100000
3587  017402  050315                          BIS     R3,(R5)         ;(R5)=100177
3588  017404  000273                          +SEC!SEV!SEN            ;SET C,V, & N
3589  017406  131215                          BITB    (R2),(R5)       ;(R2)=100000,(R5)=100177, CC=0101
3590  017410  103002                          BCC     BITB1
3591  017412  102401                          BVS     BITB1
3592  017414  001401                          BEQ     .+4
3593  017416  104000                  BITB1:  HLT
3594
3595  017420  151215                          BISB    (R2),(R5)       ;(R2)=100000,(R5)=100377, CC=1001
3596  017422  103001                          BCC     BISB1
3597  017424  100401                          BMI     .+4
3598  017426  104000                  BISB1:  HLT
3599
3600  017430  141215                          BICB    (R2),(R5)       ;(R2)=100000,(R5)=100177, CC=0001
3601  017432  103002                          BCC     BICB1
3602  017434  001401                          BEQ     BICB1
3603  017436  100001                          BPL     .+4
3604  017440  104000                  BICB1:  HLT
```

```
3605
3606  017442  105112                BICB    (R5),(R2)       ;(R5)=100177,(R2)=000000, CC=0100
3607  017444  001002                BNE     BICB1A
3608  017446  105712                TSTB    (R2)
3609  017450  001401                BEQ     .+4
3610  017452  104000      BICB1A:   HLT
3611
3612  017454  000402                BR      .+6             ;RESERVE TWO WORDS FOR DATA
3613  017456  000000                .WORD   0               ;SOURCE DATA
3614  017460  000000                .WORD   0               ;DEST DATA
3615  017462  010705                MOV     PC,R5
3616  017464  005745                TST     -(R5)
3617  017466  105045                CLRB    -(R5)           ;R5 POINTS TO DEST ODD BYTE
3618  017470  010504                MOV     R5,R4
3619  017472  105044                CLRB    -(R4)           ;R4 POINTS TO DEST EVEN BYTE
3620  017474  010403                MOV     R4,R3
3621  017476  105043                CLRB    -(R3)           ;R3 POINTS TO SOURCE ODD BYTE
3622  017500  010302                MOV     R3,R2
3623  017502  105042                CLRB    -(R2)           ;R2 POINTS TO SOURCE EVEN BYTE
3624
3625                        ;COMMENTS ARE LEAST SIGNIFICANT 4 BITS OF BYTES POINTED TO BY R2,R3
3626                        ;R4, AND R5 RESPECTIVELY AND THE REMAINING BITS ARE 0'S.
3627  017504  000261                SEC                     ;SET CARRY
3628                                                        ;(R2),(R3),(R4),(R5)
3629  017506  106112                ROLB    (R2)            ;0001,0000,0000,0000
3630  017510  111214                MOVB    (R2),(R4)       ;0001,0000,0001,0000
3631  017512  106112                ROLB    (R2)            ;0010,0000,0001,0000
3632  017514  111213                MOVB    (R2),(R3)       ;0010,0010,0001,0000
3633  017516  106112                ROLB    (R2)            ;0100,0010,0001,0000
3634  017520  111315                MOVB    (R3),(R5)       ;0100,0010,0001,0010
3635  017522  106112                ROLB    (R2)            ;1000,0010,0001,0010
3636  017524  106113                ROLB    (R3)            ;1000,0100,0001,0010
3637  017526  151215                BISB    (R2),(R5)       ;1000,0100,0001,1010
3638  017530  131512                BITB    (R5),(R2)       ;1000,0100,0001,1010
3639  017532  001426                BEQ     BIN1
3640  017534  151314                BISB    (R3),(R4)       ;1000,0100,0101,1010
3641  017536  131413                BITB    (R4),(R3)       ;1000,0100,0101,1010
3642  017540  001423                BEQ     BIN1
3643  017542  105213                INCB    (R3)            ;1000,0101,0101,1010
3644  017544  121314                CMPB    (R3),(R4)       ;1000,0101,0101,1010
3645  017546  001020                BNE     BIN1
3646  017550  106113                ROLB    (R3)            ;1000,1010,0101,1010
3647  017552  121315                CMPB    (R3),(R5)       ;1000,1010,0101,1010
3648  017554  001015                BNE     BIN1
3649  017556  106212                ASRB    (R2)            ;0100,1010,0101,1010
3650  017560  131214                BITB    (R2),(R4)       ;0100,1010,0101,1010
3651  017562  001412                BEQ     BIN1
3652  017564  106015                RORB    (R5)            ;0100,1010,0101,0101
3653  017566  121415                CMPB    (R4),(R5)       ;0100,1010,0101,0101
3654  017570  001007                BNE     BIN1
3655  017572  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                                  ;*TEST 22        CHECK BINARY WORD OPS USING ADDRESS MODE 2 & 4
3670                                  ;;**********************************************************************
3671  017622  000004                  TST22:  SCOPE
3672  017624  112737  000022  001202          MOVB    #22,@#$TSTNM               ;LOAD TEST NUMBER
3673  017632  013737  001202  177570          MOV     @#$TSTNM,@#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                                  ;*TEST 23        CHECK BINARY BYTE OPS USING ADDRESS MODE 2 & 4
3758                                  ;;*****************************************************************
3759  020114  000004                  TST23:  SCOPE
3760  020116  112737  000023  001202          MOVB    #23,@#$TSTNM            ;LOAD TEST NUMBER
3761  020124  013737  001202  177570          MOV     @#$TSTNM,@#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              BINB:     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              CMPB2:    HLT
3815
3816  020302  000261                        SEC                     ;SET C BIT, CC=0X11
3817  020304  134344                        BITB    -(R3),-(R4)     ;CC=X101
3818  020306  103002                        BCC     BITB2
3819  020310  102401                        BVS     BITB2
3820  020312  001401                        BEQ     .+4
3821  020314  104000              BITB2:    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                              ;*TEST 24        CHECK BINARY WORD OPS USING ADDRESS MODES 3 & 5
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                              ;*TEST 25       CHECK BINARY BYTE OPS USING ADDRESS MODES 3 & 5
3893                              ;;*****************************************************************
3894  020524  000004              TST25:    SCOPE
3895  020526  112737  000025  001202        MOVB    #25,@#$TSTNM            ;LOAD TEST NUMBER
```

```
3896  020534  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
3897  020542  000406                          BR      1$                      ;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                  MOVB    #200,@(R4)+             ;SOURCE DATA=100001
3922  020624  115433                          MOVB    @-(R4),@(R3)+
3923  020626  115433                          MOVB    @-(R4),@(R3)+           ;DEST DATA=000600
3924  020630  103401                          BCS     .+4
3925  020632  104000                          HLT                             ;ERROR! MOV DOES AFFECT C BIT IN PSW
3926  020634  022715  000600                  CMP     #600,(R5)               ;CHECK DEST DATA
3927  020640  001401                          BEQ     .+4
3928  020642  104000                          HLT                             ;ERROR! INCORRECT RESULT
3929  020644  024343                          CMP     -(R3),-(R3)             ;POINT R4 BACK TO EVEN BYTE
3930  020646  153433                          BISB    @(R4)+,@(R3)+
3931  020650  153433                          BISB    @(R4)+,@(R3)+           ;DEST DATA=100601
3932  020652  022715  100601                  CMP     #100601,(R5)            ;CHECK RESULT
3933  020656  001401                          BEQ     .+4
3934  020660  104000                          HLT                             ;ERROR! INCORRECT DEST DATA AFTER BISB
3935  020662  145453                          BICB    @-(R4),@-(R3)
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                                ;*TEST 26        CHECK BINARY OPS USING ADDRESS MODE 6
3950                                ;;*****************************************************************
3951  020712  000004                TST26:  SCOPE
```

```
3952  020714  112737  000026  001202          MOVB    #26,@#$TSTNM            ;LOAD TEST NUMBER
3953  020722  013737  001202  177570          MOV     @#$TSTNM,@#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 ROUTSINE 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),R0
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        MOVB    #27,@#$TSTNM            ;LOAD TEST NUMBER
4015  021226  013737  001202  177570        MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
4016                                ;NOTE: SDATAB(2), AND DDATAB(4) REFERENCE EVEN BYTE OF SOURCE & DEST DATA
4017                                ;AND SDATAB(3), AND DDATAB(5) FEFERENCE 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  MOVB    #177125,SDATAB(3)       ;SOURCE DATA = 052652
4027  021266  016264  021374  021376  MOV     SDATAB(2),DDATAB(4)
4028  021274  052764  125125  021376  BIS     #125125,DDATAB(4)       ;DEST DATA = 177777
4029  021302  136263  021374  021374  BITB    SDATAB(2),SDATAB(3)
4030  021310  001401                  BEQ     .+4
4031  021312  104000            BITB6:  HLT
4032
4033  021314  146264  021374  021376  BICB    SDATAB(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    SDATAB(3),DDATAB(4)
4037  021334  001401                  BEQ     .+4
4038  021336  104000                  HLT
4039
4040  021340  146365  021374  021376  BICB    SDATAB(3),DDATAB(5)
4041  021346  126265  021374  021376  CMPB    SDATAB(2),DDATAB(5)
4042  021354  001401                  BEQ     .+4
4043  021356  104000                  HLT
4044
4045  021360  136564  021376  021376  BITB    DDATAB(5),DDATAB(4)
4046  021366  001401                  BEQ     .+4
4047  021370  104000                  HLT
4048  021372  000415                  BR      UB7             ;RESERVE TWO WORDS
4049  021374  000000          SDATAB: .WORD   0               ;RESERVED FOR SOURCE DATA
4050  021376  000000          DDATAB: .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        MOVB    #30,@#$TSTNM            ;LOAD TEST NUMBER
4058  021410  013737  001202  177570        MOV     @#$TSTNM,@#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                                    ;*TEST 31       SOME MISCELLANEOUS OPERATIONS INVOLVING THE PC
4118                                    ;*      NOTE: NONE OF THESE OPERATIONS SHOULD AFFECT THE PC
4119                                    ;;*****************************************************************
```

```
 4120  021624  000004
 4121  021626  112737  000031  001202  TST31:  SCOPE
 4122  021634  013737  001202  177570          MOVB    #31,@#$TSTNM            ;LOAD TEST NUMBER
 4123  021642  005000                          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
 4124  021644  005067  000072                  CLR     R0
 4125  021650  010707                          CLR     1$
 4126  021652  120707                          MOV     PC,PC
 4127  021654  030707                          CMPB    PC,PC
 4128  021656  060007                          BIT     PC,PC
 4129  021660  105707                          ADD     R0,PC
 4130  021662  005507                          TSTB    PC
 4131  021664  021007                          ADC     PC
 4132  021666  131007                          CMP     (R0),PC
 4133  021670  062707  000000                  BITB    (R0),PC
 4134  021674  023707  001534                  ADD     #0,PC
 4135  021700  133707  001534                  CMP     @#FACTOR,PC
 4136  021704  000240                          BITB    @#FACTOR,PC
 4137                                          NOP
 4138                                  ;THE NEXT TWO INSTRUCTION CAUSE THE PROGRAM TO JUMP TO THE UNRELOCATED
 4139  021706  163707  001534                  ;CODE AND TO RETURN ON THE FOLLOWING INST (IF THE CODE IS RELOCATED)
 4140  021712  063707  001534                  SUB     @#FACTOR,PC     ;JUMPS TO UNRELOCATED CODE
 4141  021716  000240                          ADD     @#FACTOR,PC     ;RETURNS
 4142  021720  024607                          NOP
 4143  021722  132607                          CMP     -(SP),PC
 4144  021724  026707  000012                  BITB    (SP)+,PC
 4145  021730  166707  000006                  CMP     1$,PC
 4146  021734  046707  000002                  SUB     1$,PC
 4147  021740  000401                          BIC     1$,PC
 4148  021742  000000                          BR      .+4             ;BRANCH OVER 1$
 4149  021744  000004                  1$:     0
 4150  021746  010702                  RELE2:  SCOPE
 4151  021750  062702  000012                  MOV     PC,R2
 4152  021754  012707  043764                  ADD     #12,R2
 4153  021760  000000                          MOV     #RELOC,PC       ;GO RELOCATE PROGRAM CODE
 4154                                  REL22:  .WORD   0
 4155                                  ;2222222222222 LAST ADDRESS OF CODE TO BE RELOCATED 22222222222
 4156
 4157                                  ;;*****************************************************************
 4158                                  ;*TEST 32        CHECK BINARY BYTE OPS USING ADDRESS MODE 0
 4159  021762  012767  000001  157336  ;;*****************************************************************
 4160  021770  000004                          MOV     #1,$TIMES       ;;DO 1 ITERATION
 4161  021772  112737  000032  001202  TST32:  SCOPE
 4162  022000  013737  001202  177570          MOVB    #32,@#$TSTNM            ;LOAD TEST NUMBER
 4163                                          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
 4164                                  
 4165                                          .SBTTL  START OF SECTION 3
 4166  022006  010700                  ;3333333333333 FIRST ADDRESS TO BE RELOCATED 333333333
 4167  022010  005740                  REL3:   MOV     PC,R0           ;GET PC
 4168  022012  010037  001540                  TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF REL3
 4169  022016  010700                          MOV     R0,@#FRSTAD     ;SAVE
 4170  022020  162700  022020                  MOV     PC,R0           ;GET CURRENT PC
 4171  022024  010037  001534                  SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
 4172  022030  010737  001212                  MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOR
 4173  022034  062737  000030  001212          MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
 4174  022042  013737  001212  001210          ADD     #30,@#$LPERR    ;ADJUST
 4175  022050  105737  001530                  MOV     @#$LPERR,@#$LPADR
                                                TSTB    @#NEXEC         ;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                          ;*TEST 33        CHECK BINARY BYTE OPS USING ADDRESS MODE 7
4212                          ;;*****************************************************************
4213  022164  000004          TST33:  SCOPE
4214  022166  112737  000033  001202    MOVB    #33,@#$TSTNM            ;LOAD TEST NUMBER
4215  022174  013737  001202  177570    MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
4216  022202  000406                    BR      BINB7           ;RESERVE SPACE FOR ADDRESSES & DATA
4217  022204  000000          SBINB7: .WORD   0               ;CONTAINS ADDRESS OF SOURCE EVEN BYTE
4218  022206  000000                  .WORD   0               ;CONTAINS ADDRESS OF SOURCE ODD BYTE
4219  022210  000000                  .WORD   0               ;CONTAINS ADDRESS OF DEST EVEN BYTE
4220  022212  000000                  .WORD   0               ;CONTAINS ADDRESS OF DEST ODD BYTE
4221  022214  000000          DBINB7: .WORD   0               ;CONTAINS SOURCE DATA
4222  022216  000000                  .WORD   0               ;CONTAINS DEST DATA
4223
4224  022220  010700          BINB7:  MOV     PC,R0
4225  022222  024040                  CMP     -(R0),-(R0)     ;R0 = ADDRESS OF DEST DATA
4226  022224  010060  177772          MOV     R0,-6(R0)       ;LOAD ADDRES OF DEST EVEN BYTE DATA
4227  022230  010060  177774          MOV     R0,-4(R0)
4228  022234  005260  177774          INC     -4(R0)          ;LOAD ADDRESS OF DEST ODD BYTE DATA
4229  022240  005740                  TST     -(R0)           ;R0=ADDRESS OF SOURCE DATA
4230  022242  010060  177770          MOV     R0,-10(R0)      ;LOAD ADDRESS OF SOURCE EVEN BYTE DATA
4231  022246  010060  177772          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 EVEN &
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  MOVB    #-1,@SBINB7(3)  ;SRC DATA = 177400
4249  022322  132772  000377  022204  BITB    #377,@SBINB7(2) ;CHECK THAT EVEN BYTE WAS NOT AFFECTED
4250  022330  001401                  BEQ     .+4             ;BY MOVB INSTRUCTION
4251  022332  104000                  HLT
4252
4253  022334  157374  022204  022214  BISB    @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  CMPB    @SBINB7(3),@DBINB7(5)
4261  022370  001401                  BEQ     .+4
4262  022372  104000                  HLT
4263
4264  022374  147375  022204  022214  BICB    @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                                ;THIS ROUTINE SETS ALL BITS IN THE SOURCE ODD BYTE BY BISING A BIT FROM
4270                                ;THE DEST EVEN BYTE INTO THE SOURCE ODD BYTE
4271  022412  157473  022214  022204  BIS7:   BISB    @DBINB7(4),@SBINB7(3)
4272  022420  106174  022214          ROLB    @DBINB7(4)
4273  022424  103372                  BCC     BIS7
4274  022426  022772  177400  022204  CMP     #177400,@SBINB7(2)      ;CHECK RESULT
4275  022434  001401                  BEQ     .+4
4276  022436  104000                  HLT
4277
4278  022440  000372  022204          SWAB    @SBINB7(2)      ;SRC DATA = 000377
4279  022444  112775  000200  022214  MOVB    #200,@DBINB7(5) ;DEST DATA = 100000
4280
4281  022452  147572  022214  022204  BIC7:   BICB    @DBINB7(5),@SBINB7(2)
4282  022460  106075  022214          RORB    @DBINB7(5)
4283  022464  103372                  BCC     BIC7
4284  022466  005772  022204          TST     @SBINB7(2)
4285  022472  001401                  BEQ     .+4
4286  022474  104000                  HLT
4287
```

```
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                          ASL     -(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                          ;*TEST 34        CHECK JUMP INSTRUCTIONS
4341                          ;;*****************************************************************
4342  022700  000004          TST34:  SCOPE
4343  022702  112737  000034  001202          MOVB    #34,@#$TSTNM            ;LOAD TEST NUMBER
```

```
4344  022710  013737  001202  177570          MOV     @#$TSTNM,@#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  177770    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                          ;*TEST 35        CHECK JSR INSTRUCTIONS
4433                          ;;*****************************************************************
4434  023200  000004          TST35:  SCOPE
4435  023202  112737  000035  001202          MOVB    #35,@#$TSTNM            ;LOAD TEST NUMBER
4436  023210  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
4437  023216  013705  001534  JSR1:  MOV     @#FACTOR,R5     ;GET RELOCATION FACTOR
4438  023222  012702  023254          MOV     #3$,R2          ;FORM DEST ADRS
4439  023226  060502                  ADD     R5,R2           ;ADD RELOCATION FACTOR
4440  023230  000277                  SCC                     ;PRESET CC'S
4441  023232  000242                  CLV
4442  023234  004512                  JSR     R5,(R2)         ;GO TO 3$ 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$:   BR      4$
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$:
4534                                  ;;**********************************************************************
4535                                  ;*TEST 36       CHECK IOT TRAP (AND ROLB/ASLB)
4536                                  ;*      THIS TEST CHECKS THAT THE PSW IS CORRECT AFTER THE IOT AND THAT THE
4537                                  ;*      'NEW'PSW (FROM IOTVEC+2) IS CORRECT.
4538                                  ;;**********************************************************************
4539  023530  000004                  TST36:  SCOPE
4540  023532  112737  000036  001202          MOVB    #36,@#$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
```

```
                        BNE     99$
                        BIT     #UM,R3                  ;BRANCH TO 3$ IF IN USER MODE
                        BMI     3$
                        BNE     2$                      ;BRANCH TO 2$ IF IN SUPER MODE
                        CMP     R2,R4                   ;CHECK PSW AFTER IOT
                        BNE     99$
                        BR      4$

                2$:     BIC     #PUM,R4                 ;CLEAR PREV MODE BITS
                        BIS     #PSM,R4                 ;SET PREV SUPER MODE
                        CMP     R2,R4                   ;CHECK PSW AFTER IOT
                        BNE     99$
                        BR      4$

                3$:     BIS     #PUM,R4                 ;SET PREV USER MODE
                        CMP     R2,R4                   ;CHECK PSW AFTER IOT
                        BNE     99$

                4$:     CLR     R2
                        SEC
                        ROLB    R0                      ;ROTATE R0
                        BVC     .-2                     ;UNTIL V SETS (R0=200)

                        ASLB    R0                      ;SHIFT SHOULD SET CARRY
                        BCC     99$
                        BVC     99$
                        BNE     99$
                        TST     R0
                        BEQ     .+4
               99$:     HLT                             ;ERROR! ROL/ASL FAILED TO SET
                                                        ;CC'S PROPERLY (IF R2=0) OR IN-
                                                        ;CORRECT PSW AFTER IOT (IF R2 NOT 0)
                        BIC     #PR7,R4
                        MOV     R4,@#PSW                ;RESTORE PSW
                        MOV     #SUPSTK,SP              ;RESTORE STACK PTR
                ;;*******************************************************************
                ;*TEST 37        CHECK EMT TRAP SEQUENCE
                ;;*******************************************************************
               TST37:   SCOPE
                        MOVB    #37,@#$TSTNM            ;LOAD TEST NUMBER
                        MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
                        .EQUIV  IOT,HLT                 ;REDEFINE HLT CALL
                        MOV     #$ERROR,@#IOTVEC        ;SETUP VECTOR
                        MOV     #PR7,@#IOTVEC+2
                        CLR     R0
                        MOV     PC,-(SP)
                        ADD     #EMT1-.,(SP)
                        MOV     (SP)+,@#EMTVEC
                        SEV                             ;SET V
                        MOV     @#PSW,@#EMTVEC+2        ;RETAIN CURRENT PSW ON TRAP
                        +SEZ!SEC
                        EMT                             ;TRAP TO EMT1
                        BEQ     EMT1C                   ;GO TO EMT1C
                        HLT                             ;ERROR! INCORRECT CC'S WERE SET ON RETURN
               EMT1:    BVC     EMT1B                   ;'V' SHOULD'VE SET ON EMT TRAP
                        COMB    R0                      ;R0=000377,CC'S=1001
```

```
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,@#$TSTNM            ;LOAD TEST NUMBER
4664  024206  013737  001202  177570          MOV     @#$TSTNM,@#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     #$TRAP,@#TRAPVEC        ;RESTORE TRAP VECTOR
4681  024304  104000                          HLT
4682  024306  000420                          BR      TRAP1C
4683  024310  100404                  TRAP1:  BMI     .+12                    ;N BIT GOT SET ON TRAP
4684  024312  012737  062144  000034          MOV     #$TRAP,@#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     #$TRAP,@#TRAPVEC        ;RESTORE TRAP VECTOR
4690  024340  104000                          HLT
4691  024342  124646                          CMPB    -(SP),-(SP)
4692  024344  032626                          BIT     (SP)+,(SP)+
4693  024346  000002                          RTI                             ;RETURN TO INST FOLLOWING TRAP (1$)
4694
4695  024350  012702  000036          TRAP1C: MOV     #TRAPVEC+2,R2           ;RESTORE VECTORS
4696  024354  012712  000340                  MOV     #PR7,(R2)
4697  024360  012742  062144                  MOV     #$TRAP,-(R2)
4698  024364  042737  000340  000016          BIC     #PR7,@#TBITVEC+2
4699  024372  105037  177776                  CLRB    @#PSW                   ;GO BACK TO PRIORITY 0
4700
4701  024376  000004                  RELE3:  SCOPE
4702  024400  010702                          MOV     PC,R2
4703  024402  062702  000012                  ADD     #12,R2
4704  024406  012707  043764                  MOV     #RELOC,PC       ;GO RELOCATE PROGRAM CODE
4705  024412  000000                  REL33:  .WORD   0
4706                                  ;3333333333333 LAST ADDRESS OF CODE TO BE RELOCATED 33333333333
4707
4708                                  ;;*****************************************************************
4709                                  ;*TEST 41        CHECK STACK OVERFLOW
4710                                  ;;*****************************************************************
4711  024414  012767  000001  154704          MOV     #1,$TIMES       ;;DO 1 ITERATION
4712  024422  000004                  TST41:  SCOPE
4713  024424  112737  000041  001202          MOVB    #41,@#$TSTNM            ;LOAD TEST NUMBER
4714  024432  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
4715
4716                                          .SBTTL  START OF SECTION 4
4717                                  ;4444444444444 FIRST ADDRESS TO BE RELOCATED 44444444
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,@#$LPERR             ;SET LOOP ADDRESS
4725  024466  062737  000030  001212          ADD     #30,@#$LPERR            ;ADJUST
4726  024474  013737  001212  001210          MOV     @#$LPERR,@#$LPADR
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,7$                ;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                 43$:    .WORD   0
 4741  024566  062716  000100                 ADD     #41$-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                 CMP     (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 1$:     BISB    7$+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                 2$:     MOV     (SP)+,R0            ;SAVE PC OF INSTRUCTION THAT TRAPPED
 4773  024716  012602                         MOV     (SP)+,R2            ;SAVE PSW
 4774  024720  012706  000700                 MOV     #SUPSTK,SP          ;SET STACK PTR
 4775  024724  104000                         HLT                         ;ERROR! AN INSTRUCTION THAT WAS NOT
 4776                                                                     ;SUPPOESED TO TRAP TRAPPED
 4777                                                                     ;R0 CONTAINS PC, R2 CONTAINS PSW
 4778  024726  000443                         BR      6$                  ;EXIT TEST
 4779                                 ;THE BELOW INSTRUCTIONS WILL CAUSE A STACK OVERFLOW
 4780                                 ;STACK PTR IS AT 376
 4781  024730  062737  000066  000004 3$:     ADD     #4$-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$:    BIS     #400,R0                 ;SET IND THAT BPT WAS EXECUTED
4806  025020  000002                          RTI
4807
4808                                  ;CHECK THAT ABOVE INSRUCTIONS DID TRAP
4809  025022  012706  000700          5$:     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                  CLR     @#BPTVEC+2
4818  025054  012746                          MOV     (PC)+,-(SP)     ;PUSH OLD PSW ONTO STACK
4819  025056  000000                  7$:     .WORD   0               ;CONTAINS SAVED PSW
4820  025060  010746                          MOV     PC,-(SP)        ;PUSH CURRENT PC ONTO STACK
4821  025062  062716  000006                  ADD     #6,(SP)         ;ADD OFFSET
4822  025066  000002                          RTI
4823  025070  012706  000700                  MOV     #SUPSTK,SP      ;SET STACK PTR
4824  025074  012737  064270  000004          MOV     #ERPRT,@#ERRVEC ;RESET TIME OUT VECTOR
4825  025102  013737  177776  000006          MOV     @#PSW,@#ERRVEC+2
4826  025110  052737  000340  000006          BIS     #PR7,@#ERRVEC+2
4827  025116  042737  000020  000006          BIC     #BIT4,@#ERRVEC+2
4828  025124  005037  177766                  CLR     @#CPUERR
4829                          ;;*****************************************************************
4830                          ;*TEST 42        CHECK THAT ALL RESERVED INSTRUCTIONS TRAP
4831                          ;;*****************************************************************
4832  025130  000004          TST42:  SCOPE
4833  025132  112737  000042  001202          MOVB    #42,@#$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                          0                       ;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                   ;GROUP 4D
4897  025374  076127                          76127                   ;   ''
4898  025376  076133                          76133                   ;GROUP 4E
4899  025400  076137                          76137                   ;   ''
4900  025402  076146                          76146                   ;GROUP 4F
4901  025404  076147                          76147                   ;   ''
4902  025406  076160                          76160                   ;GROUP 4G
4903  025410  076167                          76167                   ;   ''
4904  025412  076200                          76200                   ;GROUP 4H
4905  025414  076600                          76600                   ;   ''
4906  025416  076602                          76602                   ;GROUP 4I
4907  025420  076777                          76777                   ;   ''
4908  025422  106400                          106400                  ;GROUP 5
4909  025424  106477                          106477                  ;   ''
4910  025426  106700                          106700                  ;GROUP 6
4911  025430  107777                          107777                  ;   ''
4912  025432  170000                  51$:    170000                  ;GROUP 7        FLOATING POINT
4913  025434  177777                          177777                  ;   ''          INSTRUCTIONS
4914  025436  000000                          0                       ;0 TERMINATES THE TABLE
4915
4916  025440  012737  064216  000010  7$:     MOV     #RESERR,@#RESVEC        ;RESTORE RESERVED TRAP
4917                                  ;;*****************************************************************
4918                                  ;*TEST 43        CHECK THAT ALL BITS IN THE PSW CAN BE SET AND CLEARED
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,@#CLRTBIT    ;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          MOV     #RTI,@#TBITVEC+2;SET RTI INTO RETURN
4945  025576  005014                  20$:    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  063162                  JSR     PC,@#CLRTBIT    ;GO CLEAR 'T' BIT IF SET
4968                                  ;;*********************************************************************
4969                                  ;*TEST 44        CHECK THAT ALL BITS IN THE CURRENT STACK PTR CAN BE SET CLEARED
4970                                  ;;*********************************************************************
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$:     HLT
5030  026066  105037  177776                  CLRB    @#PSW
5031  026072  010446                          MOV     R4,-(SP)        ;RESTORE ORIGINAL PSW TO STACK
5032  026074  010746                          MOV     PC,-(SP)
5033  026076  062716  000006                  ADD     #6,(SP)
5034  026102  000002                          RTI
5035                                  ;;**********************************************************************
5036                                  ;*TEST 45        CHECK THAT 'C' BIT SETS/CLEARS PROPERLY
5037                                  ;;**********************************************************************
5038  026104  000004                  TST45:  SCOPE
5039  026106  112737  000045  001202          MOVB    #45,@#$TSTNM            ;LOAD TEST NUMBER
5040  026114  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5041  026122  012727  177776          CBIT:   MOV     #177776,(PC)+   ;LOAD CONSTANT
5042  026126  000000                  1$:     .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                  CMPNUM: 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 SHOLD CLEAR
5074  026220  104000                2$:     HLT                     ;ERROR! COMPARE #+1 WITH # FAILED TO SET
5075                                                                  ;CONDITION CODES IN PSW CORRECTLY
5076  026222  105037  177776              CLRB    @#PSW           ;ENSURE PRIORITY 0
5077  026226  000004                RELE4:  SCOPE
5078  026230  010702                        MOV     PC,R2
5079  026232  062702  000012                ADD     #12,R2
5080  026236  012707  043764                MOV     #RELOC,PC       ;GO RELOCATE PROGRAM CODE
5081  026242  000000                REL44:  .WORD   0
5082                                ;444444444444 LAST ADDRESS OF CODE TO BE RELOCATED 44444444444
5083
5084                                ;;*****************************************************************
5085                                ;*TEST 46        CHECK EXTENDED INSTRUCTION SET
5086                                ;;*****************************************************************
5087  026244  012767  000001  153054        MOV     #1,$TIMES       ;;DO 1 ITERATION
5088  026252  000004                TST46:  SCOPE
5089  026254  112737  000046  001202        MOVB    #46,@#$TSTNM            ;LOAD TEST NUMBER
5090  026262  013737  001202  177570        MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5091
5092                                        .SBTTL  START OF SECTION 5
5093                                ;5555555555555 FIRST ADDRESS TO BE RELOCATED 555555555
5094  026270  010700                REL5:   MOV     PC,R0           ;GET PC
5095  026272  005740                        TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF REL5
5096  026274  010037  001540                MOV     R0,@#FRSTAD     ;SAVE
5097  026300  010700                        MOV     PC,R0           ;GET CURRENT PC
5098  026302  162700  026302                SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
5099  026306  010037  001534                MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOR
5100  026312  010737  001212                MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
5101  026316  062737  000030  001212        ADD     #30,@#$LPERR    ;ADJUST
5102  026324  013737  001212  001210        MOV     @#$LPERR,@#$LPADR
5103  026332  105737  001530              TSTB    @#NEXEC         ;BR IF TEST CODE TO BE EXECUTED
5104  026336  001402                        BEQ     .+6
5105  026340  000167  001510                JMP     RELE5
5106  026344  005000                EXTINST:CLR     R0
5107  026346  000277                        SCC                     ;PRESET CC'S
5108  026350  006700                        SXT     R0              ;EXTEND SIGN (1) INTO R0
5109  026352  103005                        BCC     SXT0            ;CHECK RESULT CC'S
5110  026354  102404                        BVS     SXT0
5111  026356  001403                        BEQ     SXT0
5112  026360  100002                        BPL     SXT0
5113  026362  005200                        INC     R0              ;CHECK RESULT
5114  026364  001401                        BEQ     .+4
5115  026366  104000                SXT0:   HLT
5116
5117  026370  010700                        MOV     PC,R0
5118  026372  010002                        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)+
5160  026514  000401                  BR      SXT4            ;PRESERVE WORD FOR DATA
5161  026516  000000                  .WORD   0               ;RESERVED FOR DATA
5162  026520  012722  125252  SXT4:   MOV     #125252,(R2)+   ;PRESET DATA
5163  026524  006742                  SXT     -(R2)           ;EXTEND SIGN
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           ;CHECK RESULT OF SXT
5210  026710  001401                          BEQ     .+4
5211  026712  104000                  SXT6A:  HLT                     ;ERROR!  SXT ABOVE FAILED TO EXTEND
5212                                                                  ;SIGN
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 SXRB
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)+       ;SXRB = 152525 (COMPLEMENT OF R4)
5237  026772  005002                        CLR     R2
5238  026774  074253                        XOR     R2,@-(R3)       ;SXRB 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,SXRB         ;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                     ;CLEAR N BIT
5248  027020  006773  000002                SXT     @2(R3)          ;SXRB = 0
5249  027024  001401                        BEQ     .+4
5250  027026  104000                SXT7:   HLT                     ;ERROR!  SXT FAILED
5251
5252  027030  074473  000002                XOR     R4,@2(R3)       ;SXRB = 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                                        ;*TEST 47        SOB TEST
5258                                        ;*      NOTE:   DO NOT INSERT ANY CODE IN FOLLOWING SOB TESTS
5259                                        ;               SINCE IT TESTS THE MAXIMUM BRANCH WIDTH OF THE INSTRUCTION.
5260                                        ;;*****************************************************************
5261  027044  000004                TST47:  SCOPE
5262  027046  112737  000047  001202        MOVB    #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                     ;SET CONDITION CODES BEFORE SOB
5302  027172  077211                  SOB     R2,SOB3         ;BRANCH TO SOB3 UNTIL R2=0
5303  027174  001403                  BEQ     SOB4            ;CHECK CONDITION CODES AFTER
5304  027176  100402                  BMI     SOB4            ;SOB FALLS THROUGH
5305  027200  005702                  TST     R2              ;CHECK IF R2=0
5306  027202  001401                  BEQ     .+4
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                     ;ERROR!
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                     ;ERROR!
5317
5318  027226  012704  100000  SOB5A:  MOV     #100000,R4      ;R4=100000
5319  027232  000403                  BR      1$
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                                                          ;INCREMENT PROPERLY
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  2$:     MOV     #100,R3         ;R3=100
5329  027252  077301          SOB6:   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                                                          ;IF SOB BRANCHES INCORRECTLY
5336                                                          ;WHEN CHECKING MAX. BRANCH,
5337                                                          ;R5 WILL NOT BE CLEARED AT
5338                                                          ;THIS POINT INDICATING AN ERROR.
5339
5340  027264  001401                  BEQ     .+4             ;BRANCH IF SOB BRANCHES CORRECTLY
5341  027266  104000                  HLT                     ;ERROR!
5342
```

```
                        INC     R5              ;SET INDICATOR (R5)
                        SOB     R4,SOB9         ;TEST MAX. BRANCH OF SOB
                        TST     R4              ;CHECK IF R4=0
                        BEQ     .+4
                        HLT                     ;ERROR!
                    ;;*********************************************************************
                    ;*TEST 50        CHECK THE MARK INSTRUCTION
                    ;;*********************************************************************
                    TST50:  SCOPE
027302  000004
027304  112737  000050  001202      MOVB    #50,@#$TSTNM            ;LOAD TEST NUMBER
027312  013737  001202  177570      MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
027320  010602                  MRKTST: MOV     SP,R2
027322  010705                      MOV     PC,R5           ;THE STACK LOOKS LIKE THIS AFTER
027324  010500                      MOV     R5,R0           ;THE JSR INSTRUCTION
027326  010546                      MOV     R5,-(SP)        ; -2(SP)= R0    THIS IS A
027330  010746                      MOV     PC,-(SP)        ; -4(SP)= PC    STRING
027332  010746                      MOV     PC,-(SP)        ; -6(SP)= PC+2  OF
027334  010746                      MOV     PC,-(SP)        ;-10(SP)= PC+4  FIVE
027336  010746                      MOV     PC,-(SP)        ;-12(SP)= PC+6  DUMMY
027340  010746                      MOV     PC,-(SP)        ;-14(SP)= PC+10 ARGUMENTS
027342  012746  006405              MOV     #MARK+5,-(SP)   ;-16(SP)= MARK 5
027346  010605                      MOV     SP,R5           ;-20(SP)= PC PUSHED BY JSR
027350  004767  000002              JSR     PC,MARK1
027354  000403                      BR      .+10
027356  000205                  MARK1:  RTS     R5
027360  104000                      HLT                     ;ERROR! SHOULD BE DOING MARK 5 INST.
027362  000407                      BR      MARKEX
027364  020602                      CMP     SP,R2
027366  001402                      BEQ     .+6
027370  104000                      HLT                     ;ERROR! SP NOT RETURNED TO PROPER
027372  000403                      BR      MARKEX          ;VALUE BY MARK INSTRUCTION
027374  020005                      CMP     R0,R5
027376  001401                      BEQ     .+4
027400  104000                      HLT                     ;ERROR! DID NOT RESTORE R5 FROM STACK
027402  010206                  MARKEX: MOV     R2,SP           ;RESTORE SP
                    ;;*********************************************************************
                    ;*TEST 51        RTT/RTI TEST
                    ;*      RTT/RTI TEST INSURES THAT CP DOES THE INSTRUCTION FOLLOWING
                    ;*      AN RTT IF THE 'T'BIT IS SET IN THE PSW,BUT DOES HONOR
                    ;*      THE TRAP IMMEDIATELY IF IT EXECUTES AN RTI
                    ;*      INSTRUCTION SEQUENCE-RTT
                    ;*      2$:     RTT                     ;NO 'T' TRAP AFTER RTT
                    ;*              INC     R0              ;R0=000001
                    ;*                                      ;'T' TRAP TO 5$ AFTER INC
                    ;*      5$:     COM     R0              ;R0=177776
                    ;*              MOV     SAVPSW,2(SP)    ;CLEAR 'T' BIT IN RETURN PSW
                    ;*              RTI                     ;RETURN TO INSTRUCTION FOLLOWING INC
                    ;*              CMP     #RTT,2$         ;CHECK
                    ;*                      ETC
                    ;*      INSTRUCTION SQUENCE-RTI
                    ;*      2$:     RTI                     ;'T' TRAP AFTER RTI
                    ;*      5$:     COM     R0              ;R0=177777
                    ;*              MOV     SAVPSW,2(SP)    ;CLEAR 'T' BIT IN RETURN PSW
                    ;*              RTI                     ;RETURN TO INC INSTRUCTION
                    ;*              INC     R0              ;R0=000000
```

```
5399                                    ;*              CMP     #RTT,2$         ;CHECK
5400                                    ;*                      ETC
5401                                    ;;**************************************************************************
5402  027404  000004                    TST51:  SCOPE
5403  027406  112737  000051  001202            MOVB    #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$:     MOV     PC,-(SP)                ;GET CURRENT PC
5410  027446  062716  000116                    ADD     #5$-.,(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$                      ;CHECK IF INC WAS EXECUTED
5425  027532  022700  177776                    CMP     #177776,R0              ;CHECK IF COM-R0 EXECUTED
5426  027536  001406                            BEQ     4$
5427  027540  104000                            HLT                             ;ERROR!R0 NOT COMPLIMENTED
5428  027542  000415                            BR      6$                      ;EXIT TEST
5429  027544  005700                    3$:     TST     R0                      ;TEST IF TRAPED BEFORE INC INST.
5430                                                                            ;WAS EXECUTED
5431  027546  001413                            BEQ     6$
5432  027550  104000                            HLT                             ;ERROR!
5433  027552  000411                            BR      6$                      ;EXIT TEST
5434  027554  012767  000002  177726    4$:     MOV     #RTI,2$
5435  027562  000730                            BR      1$
5436  027564  005100                    5$:     COM     R0                      ;RTT CHECK
5437  027566  016766  000052  000002            MOV     SAVPSW,2(SP)
5438  027574  000002                            RTI
5439  027576  012767  000006  177704    6$:     MOV     #RTT,2$
5440  027604  012737  001520  000014            MOV     #$RTRN,@#TBITVEC                ;RESTORE 'T' TRAP VECTOR
5441  027612  005037  000016                    CLR     @#TBITVEC+2
5442  027616  042737  000360  000016            BIC     #PR7+BIT4,@#TBITVEC+2
5443  027624                            RTT1EX:
5444                                    ;;**************************************************************************
5445                                    ;*TEST 52        SECOND RTT TEST
5446                                    ;;**************************************************************************
5447  027624  000004                    TST52:  SCOPE
5448  027626  112737  000052  001202            MOVB    #52,@#$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 PRIOITY 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                             ;ERROR!DID NOT CLEAR BIT #11 OF PSW
5475  027744  022700  000100          4$:   CMP     #PR2,R0         ;TESTS THE PSW AFTER THE RTT
5476  027750  001436                        BEQ     RTT2EX
5477  027752  104000                        HLT                             ;ERROR! INCORRECT PSW AFTER THE RTT
5478  027754  000434                        BR      RTT2EX
5479
5480                                  ;TEST TO INSURE THAT RTI DOES NOT CLEAR BITS 11-15 IN USER MODE
5481  027756  052737  030340  177776  2$:   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$-.,(SP)
5485  027774  000002                        RTI                             ;ATTEMPS TO INSERT A PSW OF 0
5486  027776  022737  174340  177776  5$:   CMP     #UM+PUM+REG+PR7,@#PSW   ;SHOULD CHECK AGAINST REG #0
5487  030004  001420                        BEQ     RTT2EX
5488  030006  104000                        HLT                             ;ERROR! RTI CLEARED BITS IN PSW
5489  030010  000416                        BR      RTT2EX
5490
5491                                  ;TEST THAT BITS 11-15 AND PRIORTY BITS ARE NOT ALTERED IN SUPER 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                             ;ATTEMPTS TO CLEAR 11-15 AND ALTER PR
5497
5498  030034  022737  074200  177776  6$:   CMP     #SM+PUM+REG+PR4,@#PSW
5499  030042  001401                        BEQ     RTT2EX
5500  030044  104000                        HLT                             ;ERROR! RTT ALTERED PR IN
5501                                                                        ;SUPER MODE OR BITS 11-15.
5502  030046  016737  177572  177776  RTT2EX: MOV     SAVPSW,@#PSW
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                                  ;5555555555555 LAST ADDRESS OF CODE TO BE RELOCATED 55555555555
5509
5510                                  ;;*****************************************************************
```

```
                                        ;*TEST 53       CHECK ASH, ASHC, MUL, AND DIV INSTRUCTIONS
                                        ;;**************************************************************
030072  012767  000001  151226          TST53:  MOV     #1,$TIMES       ;;DO 1 ITERATION
030100  000004                          TST53:  SCOPE
030102  112737  000053  001202                  MOVB    #53,@#$TSTNM            ;LOAD TEST NUMBER
030110  013737  001202  177570                  MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER

                                                .SBTTL  START OF SECTION 6
                                        ;6666666666666 FIRST ADDRESS TO BE RELOCATED 666666666
030116  010700                          REL6:   MOV     PC,R0           ;GET PC
030120  005740                                  TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF REL6
030122  010037  001540                          MOV     R0,@#FRSTAD     ;SAVE
030126  010700                                  MOV     PC,R0           ;GET CURRENT PC
030130  162700  030130                          SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
030134  010037  001534                          MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOR
030140  010737  001212                          MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
030144  062737  000030  001212                  ADD     #30,@#$LPERR    ;ADJUST
030152  013737  001212  001210                  MOV     @#$LPERR,@#$LPADR
030160  105737  001530                          TSTB    @#NEXEC         ;BR IF TEST CODE TO BE EXECUTED
030164  001402                                  BEQ     .+6
030166  000167  002120                          JMP     RELE6
030172  012700  000001                  ASHL0:  MOV     #1,R0           ;R0 WILL BE THE SHIFT COUNT
030176  012703  000021                          MOV     #17.,R3         ;MAX SHIFT COUNT
030202  005067  000014                  1$:     CLR     2$              ;PRESET SAVED CC'S LOCATION=0
030206  010002                                  MOV     R0,R2           ;GET SHIFT COUNT FOR PASS
030210  010705                                  MOV     PC,R5           ;R5 & R4 WILL BE DATA SHIFTED BY
030212  010504                                  MOV     R5,R4           ;ASH & ASL INSTRUCTIONS
030214  072502                                  ASH     R2,R5           ;SHIFT R5
030216  113727  177776                          MOVB    @#PSW,(PC)+     ;SAVE CC'S
030222  000000                          2$:     .WORD   0               ;CONTAINS ASH CC'S IN EVEN BYTE
                                                                        ;ASL CC'S IN ODD BYTE
030224  006304                          3$:     ASL     R4              ;SHIFT R4
030226  113746  177776                          MOVB    @#PSW,-(SP)     ;SAVE PSW ON STACK
030232  132716  000002                          BITB    #2,(SP)         ;CHECK IF ASL SET V BIT
030236  001403                                  BEQ     30$
030240  152767  000002  177755                  BISB    #2,2$+1         ;IF ASL SET V THEN SET V IN 2$+1
030246  112637  177776                  30$:    MOVB    (SP)+,@#PSW     ;RESTORE ORIGINAL PSW
030252  077214                                  SOB     R2,3$           ;SHIFT R4 R2 TIMES
030254  153767  177776  177741                  BISB    @#PSW,2$+1      ;SAVE CC'S AFTER ASL
030262  020504                                  CMP     R5,R4           ;CHECK ASH & ASL RESULTS
030264  001004                                  BNE     4$
030266  126767  177730  177727                  CMPB    2$,2$+1         ;CHECK ASH & ASL CC'S
030274  001401                                  BEQ     .+4
030276  104000                          4$:     HLT                     ;ERROR! INCORRECT RESULT OR CC'S
030300  005200                                  INC     R0              ;INCREMENT PASS SHIFT COUNT
030302  020003                                  CMP     R0,R3
030304  001336                                  BNE     1$

030306  012700  177777                  ASHR0:  MOV     #-1,R0          ;R0 = RIGHT SHIFT COUNT FOR PASS
030312  012703  177757                          MOV     #-17.,R3        ;MAX SHIFT COUNT
030316  010002                          1$:     MOV     R0,R2           ;GET SHIFT COUNT FOR PASS
030320  010705                                  MOV     PC,R5           ;R5 & R4 = DATA TO BE SHIFTED
030322  010504                                  MOV     R5,R4           ;BY ASH & ASR INSTRUCTIONS
030324  072502                                  ASH     R2,R5           ;SHIFT R5 R2 TIMES
030326  113727  177776                          MOVB    @#PSW,(PC)+     ;SAVE CC'S IN EVEN BYTE
030332  000000                          2$:     .WORD   0               ;CONTAINS ASH CC'S IN EVEN BYTE
```

```
5567
5568  030334  005402                          NEG     R2                      ;ASR CC'S IN ODD BYTE
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          ASHCL0: 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          ASHCR0: 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$:     MOV     (SP),R0                 ;GET PASS SHIFT COUNT
5606  030472  010702                          MOV     PC,R2                   ;R2,R3 & R4,R5 ARE THE DATA REGISTERS
5607  030474  010204                          MOV     R2,R4                   ;TO BE SHIFTED BY TEST
5608  030476  005003                          CLR     R3
5609  030500  005005                          CLR     R5
5610  030502  000262                          SEV                             ;SET V BIT IN PSW
5611  030504  073200                          ASHC    R0,R2                   ;SHIFT R2,R3 RIGHT R0 TIMES
5612  030506  102410                          BVS     3$                      ;SHIFT RIGHT CLEARS V
5613  030510  005400                          NEG     R0                      ;NEGATE SHIFT COUNT FOR SOB
5614  030512  006204                  2$:     ASR     R4                      ;SHIFT R4,R5 RIGHT R0 TIMES
5615  030514  006005                          ROR     R5
5616  030516  077003                          SOB     R0,2$
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                  CMP     (SP),2(SP)      ;CHECK IF MAX SHIFT COUNT
5624  030540  001353                          BNE     1$
5625  030542  022626                          CMP     (SP)+,(SP)+     ;RESTORE STACK PTR
5626                                  ;;*****************************************************************
5627                                  ;*TEST 54        CHECK MUL
5628                                  ;*      THE BELOW TEST OF THE MUL INSTRUCTION MULTIPLIES THE CURRENT PC
5629                                  ;*      BY 1,2,4,8 ETC AND SHIFTS THE SAME PC VALUE USING AN ASHC LEFT BY
5630                                  ;*      0,1,2,3,ETC AND COMPARES THE RESULTS. CONDITION CODE RESULTS ARE NOT CHECKED.
5631                                  ;;*****************************************************************
5632  030544  000004                  TST54:  SCOPE
5633  030546  112737  000054  001202          MOVB    #54,@#$TSTNM            ;LOAD TEST NUMBER
5634  030554  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5635  030562  012700  000001          MUL0:   MOV     #1,R0           ;R0 CONTAINS MULTIPLIER FOR MUL
5636  030566  012706  000700                  MOV     #SUPSTK,SP      ;SETUP THE STACK
5637  030572  005016                          CLR     (SP)            ;(SP) CONTAINS SHIFT VALUE FOR ASHC
5638  030574  010702                  1$:     MOV     PC,R2           ;R3,R2 & R5,R4 ARE DATA REGISTERS
5639  030576  010227                          MOV     R2,(PC)+        ;SAVE MULTIPICAND
5640  030600  000000                          .WORD   0               ;CONTAINS ORIGINAL MULTIPICAND
5641  030602  005003                          CLR     R3
5642  030604  005004                          CLR     R4
5643  030606  010205                          MOV     R2,R5           ;FOR MUL AND ASHC
5644  030610  100001                          BPL     .+4             ;IF MULTPICAND IS NEG THEN SET R4 = -1
5645  030612  005104                          COM     R4              ;FOR ASHC
5646  030614  000277                          SCC                     ;PRESET CC'S
5647  030616  070200                          MUL     R0,R2           ;MULTIPLY R2 BY R0 LEAVE PRODUCT
5648                                                                  ;IN R2,R3 MSH IN R2,LSH IN R3
5649  030620  102406                          BVS     2$
5650  030622  001405                          BEQ     2$              ;PRODUCT WILL NEVER BE = 0
5651  030624  073416                          ASHC    (SP),R4         ;'MULTIPLY' R4,R5 BY (SP) LEAVE PRODUCT
5652                                                                  ;IN R4,R5 MSH IN R4,LSH IN R5
5653  030626  020204                          CMP     R2,R4           ;CHECK MSH RESULT
5654  030630  001002                          BNE     2$
5655  030632  020305                          CMP     R3,R5           ;CHECK LSH RESULT
5656  030634  001401                          BEQ     .+4
5657  030636  104000                  2$:     HLT
5658  030640  005216                          INC     (SP)            ;INCREMENT ASHC SHIFT COUNT
5659  030642  006300                          ASL     R0              ;SHIFT MUL MULTIPLIER
5660  030644  102353                          BVC     1$
5661                                  ;CHECK MUL INST WITH MULTIPIER (R0) = 100000
5662  030646  010702                          MOV     PC,R2           ;R2 = MULTIPICAND
5663  030650  005202                          INC     R2
5664  030652  010227                          MOV     R2,(PC)+        ;SAVE MULTIPICAND
5665  030654  000000                          .WORD   0               ;CONTAINS ORIGINAL MULTIPICAND
5666  030656  005103                          COM     R3
5667  030660  010204                          MOV     R2,R4           ;R4 WILL BE MSH 'PRODUCT'
5668  030662  006204                          ASR     R4              ;FORM 'PRODUCT'
5669  030664  005104                          COM     R4              ;COMPLEMENT MSH 'PRODUCT'
5670  030666  070200                          MUL     R0,R2           ;MULTIPLY R2 BY 100000 LEAVING
5671                                                                  ;R2 = MSH, R3 = LSH PRODUCT
5672  030670  020204                          CMP     R2,R4           ;COMPARE MSH PRODUCTS
5673  030672  001002                          BNE     3$
5674  030674  020003                          CMP     R0,R3           ;CHECK LSH PRODUCT
5675  030676  001401                          BEQ     .+4
5676  030700  104000                  3$:     HLT
5677                                  ;;*****************************************************************
5678                                  ;*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 MULTPLIED 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,@#$TSTNM            ;LOAD TEST NUMBER
5686  030712  013737  001202  177570            MOV     @#$TSTNM,@#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     2$
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     .+4
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$:     .WORD   0               ;CONTAINS ASH (SP),R3 CC'S IN EVEN BYTE
5725                                                                    ;AND ASH R0,R4 CC'S IN ODD BYTE
5726  031034  072400                            ASH     R0,R4           ;SHIFT R4 LEFT R0 TIMES
5727  031036  113767  177776  177767            MOVB    @#PSW,2$+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     .+4
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)            ;INCRENENT 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     @#PSW,(PC)+     ;SAVE CC'S
5748  031116  000000                  2$:     .WORD   0               ;CONTAINS ASH (SP),R4 CC'S IN EVEN BYTE
5749                                                                  ;AND ASH R0,R5 CC'S IN ODD BYTE
5750  031120  072500                          ASH     R0,R5           ;SHIFT R5 RIGHT R0 TIMES
5751  031122  113767  177776  177767          MOVB    @#PSW,2$+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$:     HLT                     ;ERROR! EITHER RESULTS OR RESULT CC'S
5757                                                                  ;DID NOT COMPARE
5758  031146  005300                          DEC     R0              ;DECREMENT SHIFT COUNT
5759  031150  005316                          DEC     (SP)            ;DECREMENT SHIFT COUNT FOR ASH (SP),R4
5760  031152  020002                          CMP     R0,R2           ;CHECK FOR MAX RIGHT SHIFT
5761  031154  001351                          BNE     1$
5762                                  ;;**********************************************************************
5763                                  ;*TEST 56        DIVIDE AGAIN
5764                                  ;*      THE BELOW TEST CHECKS THE DIVIDE INSTRUCTION BY DIVIDING
5765                                  ;*      THE CURRENT PC BY ITSELF+1. THE QUOTIENT (IN R2) ALWAYS = 0,
5766                                  ;*      AND THE REMAINDER (IN R3) ALWAYS = THE CURRENT PC.
5767                                  ;;**********************************************************************
5768  031156  000004                  TST56:  SCOPE
5769  031160  112737  000056  001202          MOVB    #56,@#$TSTNM            ;LOAD TEST NUMBER
5770  031166  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5771  031174  010703                  DIV1:   MOV     PC,R3           ;CURRENT PC IS LSH DIVIDEND
5772  031176  006702                          SXT     R2              ;EXTEND SIGN TO R2 (MSH DIVIDEND)
5773  031200  010304                          MOV     R3,R4           ;SAVE ORIGINAL DIVIDEND
5774  031202  010316                          MOV     R3,(SP)         ;PUT ON STACK
5775  031204  005216                          INC     (SP)            ;ADD 1 (WILL BE DIVISOR)
5776  031206  100002                          BPL     1$              ;BRANCH IF POSITIVE
5777  031210  162716  000002                  SUB     #2,(SP)         ;MAKE DIVISOR 1 LESS THAN DIVIDEND
5778  031214  071216                  1$:     DIV     (SP),R2         ;DIVIDE R2 BY (SP)
5779  031216  103410                          BCS     2$              ;CHECK CONDITION CODES
5780  031220  102407                          BVS     2$
5781  031222  001006                          BNE     2$
5782  031224  100405                          BMI     2$
5783  031226  005702                          TST     R2              ;CHECK QUOTIENT (R2 = 0)
5784  031230  001361                          BNE     DIV1
5785  031232  010416                          MOV     R4,(SP)         ;GET ORIGINAL DIVISOR
5786  031234  020316                          CMP     R3,(SP)         ;CHECK REMAINDER
5787  031236  001401                          BEQ     .+4
5788  031240  104000                  2$:     HLT                     ;REPORT ERROR
5789                                  ;;**********************************************************************
5790                                  ;*TEST 57        CHECK SPL INSTRUCTION
```

```
5791                                    ;;*****************************************************************
5792  031242  000004                    TST57:  SCOPE
5793  031244  112737  000057  001202            MOVB    #57,@#$TSTNM            ;LOAD TEST NUMBER
5794  031252  013737  001202  177570            MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5795  031260  012702                    SPLTST: MOV     (PC)+,R2        ;R2 CONTAINS OP CODE FOR SPL 7
5796  031262  000237                            SPL     7
5797  031264  005004                            CLR     R4
5798  031266  042744  000340                    BIC     #PR7,-(R4)      ;CLEAR PRIORITY LEVEL BITS IN PSW
5799  031272  011403                            MOV     (R4),R3         ;GET CURRENT PSW
5800  031274  042703  177757                    BIC     #177757,R3      ;R3 CONTAINS CORRECT PSW AFTER SPL
5801
5802  031300  012767  000230  000010            MOV     #SPL+0,2$       ;INITIALIZE SPL INSTRUCTIONS
5803  031306  012767  000237  000050            MOV     #SPL+7,5$
5804  031314  000257                    1$:     CCC                     ;CLEAR CONDITION CODES
5805  031316  000230                    2$:     SPL     0               ;SET PRIORITY LEVEL (NOTE: SPL=NOP IF USER/SUPER MODE)
5806  031320  121403                            CMPB    (R4),R3         ;CHECK RESULT OF SPL ABOVE
5807  031322  001401                            BEQ     .+4
5808  031324  104000                            HLT                     ;ERROR! SPL ABOVE FAILED
5809  031326  032714  140000                    BIT     #UM,(R4)        ;IF NOT IN KERNEL MODE THEN SPL
5810  031332  001002                            BNE     3$              ;ACTS AS A NOP
5811  031334  062703  000040                    ADD     #40,R3          ;SET NEXT CORRECT PSW RESULT
5812  031340  005267  177752            3$:     INC     2$              ;SET NEXT SPL INSTRUCTION
5813  031344  026702  177746                    CMP     2$,R2           ;CHECK IF DONE
5814  031350  002761                            BLT     1$              ;LOOP UNTIL DONE CHANGING SPL EACH PASS
5815  031352  012702                            MOV     (PC)+,R2        ;R2 CONTAINS SPL INSTRUCTION BELOW
5816  031354  000230                            SPL     0
5817  031356  052703  000017                    BIS     #17,R3          ;SET CONDITION CODE RESULT INTO R3
5818  031362  000277                    4$:     SCC                     ;SET CONDITION CODES
5819  031364  000237                    5$:     SPL     7               ;SET PRIORITY LEVEL
5820  031366  121403                            CMPB    (R4),R3         ;CHECK RESULT OF SPL ABOVE
5821  031370  001401                            BEQ     .+4
5822  031372  104000                            HLT                     ;ERROR! SPL ABOVE FAILED
5823  031374  032714  140000                    BIT     #UM,(R4)        ;CHECK IF IN KERNEL MODE
5824  031400  001002                            BNE     6$
5825  031402  162703  000040                    SUB     #40,R3          ;SET NEXT CORRECT PSW RESULT
5826  031406  005367  177752            6$:     DEC     5$              ;SET NEXT SPL
5827  031412  026702  177746                    CMP     5$,R2           ;CHECK IF DONE ALL SPL'S
5828  031416  002361                            BGE     4$
5829                                    ;;*****************************************************************
5830                                    ;*TEST 60       CHECK PIRQ LOGIC
5831                                    ;*      THIS TEST CHECKS THAT WHEN A REQUEST IS MADE AT A LEVEL = TO THE
5832                                    ;*      CURRENT PROCESSER PRIORITY LEVEL THAT NO INTERRUPT TAKES PLACE, AND
5833                                    ;*      THAT WHEN A REQUEST IS MADE AT A LEVEL 1 GREATER THAN THE CURRENT PRO-
5834                                    ;*      CESSER LEVEL THAT AN INTERRUPT OCCURS
5835                                    ;;*****************************************************************
5836  031420  000004                    TST60:  SCOPE
5837  031422  112737  000060  001202            MOVB    #60,@#$TSTNM            ;LOAD TEST NUMBER
5838  031430  013737  001202  177570            MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5839  031436  012700  031602            PIRQ0:  MOV     #4$,R0          ;R0 POINTS TO A TABLE OF CORRECT PIRQ
5840                                                                    ;CONTENTS AFTER AN INTERRUPT
5841  031442  012702  000400                    MOV     #400,R2         ;R2 CONTAINS INTERRUPT REQUEST LEVEL
5842  031446  005003                            CLR     R3              ;R3 CONTAINS PROCESSER PRIORITY LEVEL
5843  031450  012704  177772                    MOV     #PIRQ,R4        ;R4 CONTAINS ADDRESS OF PIRQ REGISTER
5844  031454  005014                            CLR     (R4)            ;INITIALZE REQUEST LEVEL TO 0
5845  031456  013737  177776  000242            MOV     @#PSW,@#PIRQVEC+2       ;RETAIN MODE & REG SET ON TRAP
5846  031464  112737  000340  000242            MOVB    #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     5$                      ;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,@#$TMP0          ;SAVE PIRQ
5863  031554  005037  177772                  CLR     @#PIRQ
5864  031560  104000                          HLT                             ;ERROR! INCORRECT PIRQ CONTENTS
5865  031562  062703  000040          6$:     ADD     #40,R3                  ;SET NEXT CP PRIORITY LEVEL
5866  031566  040214                          BIC     R2,(R4)                 ;LOWER LEVEL BY 1
5867  031570  012716  031500                  MOV     #1$,(SP)                ;ADJUST RETURN ADDRESS
5868  031574  063716  001534                  ADD     @#FACTOR,(SP)           ;TO RETURN TO 1$
5869  031600  000006                  30$:    RTT
5870
5871                                  ;TABLE OF CORRECT PIRQ REGISTER CONTENTS ON INTERRUPT
5872  031602  001042                  4$:     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$:     CLR     (R4)                    ;CLEAR PIRQ REGISTER
5881  031622  012737  000242  000240          MOV     #PIRQVEC+2,@#PIRQVEC    ;RESET PIRQVEC TO HALT AT PIRQVEC+2
5882  031630  005037  000242                  CLR     @#PIRQVEC+2
5883  031634  105037  177776                  CLRB    @#PSW
5884  031640  042737  000340  000016          BIC     #PR7,@#TBITVEC+2
5885                                  ;;*****************************************************************
5886                                  ;*TEST 61        CHECK MICRO-BREAK REGISTER
5887                                  ;*      THIS TEST SHIFTS A '0' BIT THRU ALL BIT POSITIONS.
5888                                  ;;*****************************************************************
5889  031646  000004                  TST61:  SCOPE
5890  031650  112737  000061  001202          MOVB    #61,@#$TSTNM            ;LOAD TEST NUMBER
5891  031656  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
5892  031664  012702  177770          MBRK:   MOV     #UBREAK,R2      ;SET ADDRESS OF MICRO BREAK REGISTER
5893  031670  011246                          MOV     (R2),-(SP)      ;SAVE ORIG CONTENTS
5894  031672  012700  177776                  MOV     #177776,R0      ;SET DATA PATTERN
5895  031676  010003                  1$:     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     5$              ;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$:     MOV     (SP)+,(R2)      ;RESTORE ORIG UBREAK CONTENTS
5909                                  ;;*****************************************************************
5910                                  ;*TEST 62        CHECK MFPI/MTPI INSTRUCTIONS
5911                                  ;;*****************************************************************
5912  031736  000004                  TST62:  SCOPE
5913  031740  112737  000062  001202          MOVB    #62,@#$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$-.,(SP)
5933  032056  012637  000020                  MOV     (SP)+,@#IOTVEC  ;SET IOT TRAP VECTOR
5934  032062  000004                          IOT                     ;TRAP TO 3$ BELOW
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     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
 5960  032166  010746                  HALT1:  MOV     PC,-(SP)        ;GET CURRENT PC
 5961  032170  062716  000022                  ADD     #2$-.,(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                                  ;*TEST 64        CHECK RESET IN SUPER/USER MODE
 5977                                  ;;**********************************************************************
 5978  032246  000004                  TST64:  SCOPE
 5979  032250  112737  000064  001202          MOVB    #64,@#$TSTNM            ;LOAD TEST NUMBER
 5980  032256  013737  001202  177570          MOV     @#$TSTNM,@#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                          ENDCP:
 5990  032312  000004                  RELE6:  SCOPE
 5991  032314  010702                          MOV     PC,R2
 5992  032316  062702  000012                  ADD     #12,R2
 5993  032322  012707  043764                  MOV     #RELOC,PC       ;GO RELOCATE PROGRAM CODE
 5994  032326  000000                  REL66:  .WORD   0
 5995                                  ;6666666666666 LAST ADDRESS OF CODE TO BE RELOCATED 66666666666
 5996
 5997                                  ;;**********************************************************************
 5998                                  ;*TEST 65        TEST STACK LIMIT REGISTER
 5999                                  ;;**********************************************************************
 6000  032330  012767  000001  146770          MOV     #1,$TIMES       ;;DO 1 ITERATION
 6001  032336  000004                  TST65:  SCOPE
 6002  032340  112737  000065  001202          MOVB    #65,@#$TSTNM            ;LOAD TEST NUMBER
 6003  032346  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
 6004
 6005                                          .SBTTL  START OF SECTION 7
 6006                                  ;7777777777777 FIRST ADDRESS TO BE RELOCATED 777777777
 6007  032354  010700                  REL7:   MOV     PC,R0           ;GET PC
 6008  032356  005740                          TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF REL7
 6009  032360  010037  001540                  MOV     R0,@#FRSTAD     ;SAVE
 6010  032364  010700                          MOV     PC,R0           ;GET CURRENT PC
 6011  032366  162700  032366                  SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
 6012  032372  010037  001534                  MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOR
 6013  032376  010737  001212                  MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
 6014  032402  062737  000030  001212          ADD     #30,@#$LPERR    ;ADJUST
```

```
 6015  032410  013737  001212  001210          MOV     @#$LPERR,@#$LPADR
 6016  032416  105737  001530                  TSTB    @#NEXEC         ;BR IF TEST CODE 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 'T' 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                             ;ERROR! STACK LIMIT DID NOT
 6031                                                                          ;LOAD CORRECTLY. CORRECT RESULT
 6032                                                                          ;IS IN R0
 6033  032464  006300                  2$:     ASL     R0                      ;SHIFT '1' BIT LEFT
 6034  032466  103372                          BCC     1$                      ;LOOP UNTIL 1 BIT SHIFTS OUT
 6035  032470  005042                          CLR     -(R2)                   ;CLEAR STACK LIMIT REG
 6036
 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 ERRPR
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                           101$:
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
```

```
                                MOVB    #66,@#$TSTNM            ;LOAD TEST NUMBER
                                MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER

                        KTPDR:  MOV     #PDRTBL,R2      ;SET TABLE ADDRESS OF PDR'S
                                MOV     #100360,R5      ;SET BIT MASK
                        1$:     MOV     (R2)+,R0        ;GET PDR ADDRESS
                                BEQ     100$            ;EXIT ON '0' TERMINATOR
                        2$:     MOV     #8.,(SP)        ;SET LOOP COUNT (FOR 8 REGS)
                                TSTB    @#MMON          ;BRANCH IF MEM MGMT DISABLED
                                BEQ     3$
                                ADD     #10,R0          ;SET R0 TO PDR4
                                MOV     #3,(SP)         ;AND LIMIT TO TEST 3 PDRS
                        3$:     MOV     #32.,R3         ;SET DATA COUNT
                                CLR     R4              ;INITIALIZE DATA TO BE WRITTEN
                        4$:     BIC     R5,R4           ;CLEAR NON-SETTABLE BITS
                                MOV     R4,(R0)         ;WRITE INTO PDR
                                CMP     (R0),R4         ;AND CHECK DATA READ BACK
                                BNE     99$             ;GO TO ERROR CALL
                                COM     R4              ;COMPLEMENT DATA
                                BIC     R5,R4           ;CLEAR NON-SETTABLE BITS
                                MOV     R4,(R0)         ;WRITE COMPLEMENT DATA INTO PDR
                                CMP     (R0),R4         ;AND CHECK
                                BNE     99$             ;GO TO ERROR CALL
                                ADD     R1,R4           ;STEP DATA
                                SOB     R3,4$
                        5$:     CLR     (R0)+           ;STEP TO NEXT REGISTER
                                DEC     (SP)            ;DECREMENT REGISTER COUNT
                                BNE     3$
                                BR      1$              ;GET NEXT SET OF 8 REGISTERS

                        99$:    HLT                     ;ERROR! INCORRECT DATA READ
                                                        ;BACK FROM PDR. ADDRESS OF
                                                        ;PDR IS IN R0, DATA IS IN R4
                                BR      5$              ;STEP TO NEXT REGISTER
                        100$:
                        ;;**************************************************************************
                        ;*TEST 67       PAR TEST
                        ;*      PAR TEST - THIS TEST WRITES 64. COMPLEMENTING RANDOM #'S INTO EACH PAR.
                        ;;**************************************************************************
                        TST67:  SCOPE
                                MOVB    #67,@#$TSTNM            ;LOAD TEST NUMBER
                                MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
                        KTPAR:  MOV     #PARTBL,R2      ;GET TABLE ADDRESS OF PAR'S
                                CLR     R5
                        1$:     MOV     (R2)+,R0        ;GET PAR ADDRESS
                                BEQ     100$            ;EXIT ON '0' TERMINATOR
                        2$:     MOV     #8.,(SP)        ;SET LOOP COUNT (FOR 8 REGS.)
                                TSTB    @#MMON          ;BRANCH IF MEM MGMT DISABLED
                                BEQ     3$
                                ADD     #10,R0          ;SET R0 TO PAR4
                                MOV     #3,(SP)         ;AND LIMIT TEST TO 3 PARS
                        3$:     MOV     #32.,R3         ;SET DATA COUNT
                                CLR     R4              ;INITIALIZE DATA
                        4$:     BIC     R5,R4           ;CLEAR NON-SETTABLE BITS
                                MOV     R4,(R0)         ;WRITE INTO PAR
                                CMP     (R0),R4         ;AND CHECK
```

```
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                      100$:
6171  033146  000416                        BR      TST70           ;;GO TO NEXT TEST
6172                              ;TABLES FOR PDR & PAR TESTS ABOVE
6173  033150  172300              PDRTBL:   .WORD   KIPDR0
6174  033152  177600                        .WORD   UIPDR0
6175  033154  172200                        .WORD   SIPDR0          ;CHANGED TO '0' IF 11/40
6176  033156  172320                        .WORD   KDPDR0
6177  033160  177620                        .WORD   UDPDR0
6178  033162  172220                        .WORD   SDPDR0
6179  033164  000000                        .WORD   0               ;TERMINATOR
6180
6181  033166  172340              PARTBL:   .WORD   KIPAR0
6182  033170  177640                        .WORD   UIPAR0
6183  033172  172240                        .WORD   SIPAR0          ;CHANGED TO '0' IF 11/40
6184  033174  172360                        .WORD   KDPAR0
6185  033176  177660                        .WORD   UDPAR0
6186  033200  172260                        .WORD   SDPAR0
6187  033202  000000                        .WORD   0               ;TERMINATOR
6188
6189                              ;;*****************************************************************
6190                              ;*TEST 70       CHECK KT ABORT LOGIC
6191                              ;*      THIS TEST CHECKS KT ABORT LOGIC. TEST CREATES AN ABORT CONDITION
6192                              ;*      AND INSURES THAT ABORT IS TAKEN PROPERLY. NOTE: TEST IS EXECUTED ONLY
6193                              ;*      IF TEST IS ENTERED WITH MEM MGMT ENABLED.
6194                              ;;*****************************************************************
6195  033204  000004              TST70:    SCOPE
6196  033206  112737  000070  001202        MOVB    #70,@#$TSTNM            ;LOAD TEST NUMBER
6197  033214  013737  001202  177570        MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
6198  033222  105737  001531              KTABT:    TSTB    @#MMON          ;BRANCH IF MEM MGMT DISABLED
6199  033226  001515                        BEQ     KTEX
6200  033230  005037  172350              CLR     @#KIPAR4        ;SET UP MEM MGMT REGISTERS
6201  033234  005037  172310              CLR     @#KIPDR4        ;TO ABORT IF A MEMORY
6202  033240  005037  177650              CLR     @#UIPAR4        ;REFERENCE IS MADE TO
6203  033244  005037  177610              CLR     @#UIPDR4        ;ADDRESSES (VIRTUAL) BETWEEN
6204  033250  005037  172250              CLR     @#SIPAR4
6205  033254  005037  172210              CLR     @#SIPDR4
6206  033260  013746  000250              1$:     MOV     @#MMVEC,-(SP)   ;SAVE MEM MGMT VECTOR
6207  033264  013746  000252                      MOV     @#MMVEC+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                     ;REPORT ERROR
6219  033330  000445                          BR      100$
6220                                  ;ABORT HERE
6221  033332  013700  177776          4$:     MOV     @#PSW,R0        ;SR0 SHOULD CONTAIN
6222  033336  000300                          SWAB    R0              ;CAUSE FOR ABORT AND
6223  033340  006200                          ASR     R0              ;ALSO WHICH SEGMENT
6224  033342  042700  177637                  BIC     #177637,R0      ;WAS IN USE WHEN ABORT
6225  033346  062700  100011                  ADD     #100011,R0      ;OCCURRED.
6226  033352  020037  177572                  CMP     R0,@#SR0
6227  033356  001025                          BNE     99$
6228  033360  012700  033320                  MOV     #2$,R0          ;GET ADDRESS OF INST
6229  033364  020037  177576                  CMP     R0,@#SR2        ;THAT ABORTED
6230  033370  001020                          BNE     99$
6231  033372  012700  000362                  MOV     #362,R0
6232  033376  120037  177574                  CMPB    R0,@#SR1        ;SR1 CONTAINS REGISTER
6233  033402  001013                          BNE     99$             ;MODIFICATIONS MADE
6234  033404  012700  000023                  MOV     #23,R0
6235  033410  120037  177575                  CMPB    R0,@#SR1+1
6236  033414  001006                          BNE     99$
6237  033416  012700  033320                  MOV     #2$,R0
6238  033422  005720                  5$:     TST     (R0)+           ;R0=ADDRESS OF INST FOLLOWING ABORT
6239  033424  020016                          CMP     R0,(SP)         ;(3$)
6240  033426  001001                          BNE     99$
6241  033430  000002                          RTI                     ;RETURN
6242                                  ;ENTER HERE ON ERROR
6243  033432  104000                  99$:    HLT                     ;REPORT ERROR
6244  033434  010716                          MOV     PC,(SP)
6245  033436  062716  177664                  ADD     #3$-.,(SP)
6246  033442  000002                          RTI                     ;RETURN
6247  033444  012637  000252          100$:   MOV     (SP)+,@#MMVEC+2 ;RESTORE ABORT VECTOR
6248  033450  012637  000250                  MOV     (SP)+,@#MMVEC   ;& PRIORITY.
6249  033454  012737  000001  177572          MOV     #1,@#SR0        ;CLEAR ERROR CONDITIONS
6250  033462                          KTEX:
6251                                  ;;*****************************************************************
6252                                  ;*TEST 71        MAPPING REGISTER TESTS
6253                                  ;*      THIS TEST LOADS RANDOM #'S INTO EACH MAPPING REGISTER
6254                                  ;;*****************************************************************
6255  033462  000004                  TST71:  SCOPE
6256  033464  112737  000071  001202          MOVB    #71,@#$TSTNM            ;LOAD TEST NUMBER
6257  033472  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
6258  033500  032737  000040  172516          BIT     #BIT5,@#MMR3    ;IS MAP ON?
6259  033506  001053                          BNE     MAPTW0          ;BRANCH IF YES
6260  033510  012700  170200          MAPTST: MOV     #MAPL0,R0       ;SET ADRS OF FIRST MAP REGISTER
6261  033514  012706  000700                  MOV     #SUPSTK,SP      ;SETUP THE SP
6262  033520  012716  000001                  MOV     #1,(SP)         ;SET BIT MASK FOR MAPL0 <15-01>
6263  033524  012702  177700                  MOV     #177700,R2      ;AND ALSO FOR MAPH0 <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     #MAPL0+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     @#MAPL0
6305  033662  005037  170202                  CLR     @#MAPH0
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                  REL77:  .WORD   0
6311                                  ;7777777777777 LAST ADDRESS OF CODE TO BE RELOCATED 77777777777
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
```

```
                                        ;*      SECT2   (A+B)*(A-B)=A**2-B**2
                                        ;*      SECT3   A/B*B=A
                                        ;;*****************************************************************
033704  012767  000001  145414                MOV     #1,$TIMES       ;;DO 1 ITERATION
033712  000004                          TST72:  SCOPE
033714  112737  000072  001202                MOVB    #72,@#$TSTNM            ;LOAD TEST NUMBER
033722  013737  001202  177570                MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
033730  012737  000001  001326                MOV     #1,@#$TIMES     ;SET ITTERATIONS TO 1

                                                .SBTTL  START OF SECTION 8
                                        ;8888888888888 FIRST ADDRESS TO BE RELOCATED 888888888
033736  010700                          REL8:   MOV     PC,R0           ;GET PC
033740  005740                                  TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF REL8
033742  010037  001540                          MOV     R0,@#FRSTAD     ;SAVE
033746  010700                                  MOV     PC,R0           ;GET CURRENT PC
033750  162700  033750                          SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
033754  010037  001534                          MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOR
033760  010737  001212                          MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
033764  062737  000030  001212                ADD     #30,@#$LPERR    ;ADJUST
033772  013737  001212  001210                MOV     @#$LPERR,@#$LPADR
034000  105737  001530                        TSTB    @#NEXEC         ;BR IF TEST CODE TO BE EXECUTED
034004  001402                                BEQ     .+6
034006  000167  002422                        JMP     RELE8
034012  032737  020000  001500                BIT     #FPOPT,@#OPT.CP ;FLOATING POINT AVAILABLE?
034020  001002                                BNE     100$            ;BRANCH IF YES
034022  000167  002424                        JMP     REL88+2
034026  004737  060542                100$:   JSR     PC,@#FLTSGL     ;GET RANDOM OPERANDS
034032  170127  000000                        LDFPS   #0              ;INIT FPS
034036  172537  001302                        LDF     @#$TMP0,AC1     ;LOAD A OPERAND
034042  172437  001306                        LDF     @#$TMP2,AC0     ;LOAD B OPERAND
034046  013737  001256  001414                MOV     @#$REG0,@#$AC1  ;SETUP EXTENDED
034054  013737  001260  001412                MOV     @#$REG1,@#$AC0  ;EXPONENTS
034062  004767  002210                        JSR     PC,FLTADD       ;PERFORM THE ADD
034066  174100                                STF     AC1,AC0         ;SETUP AC0 TO
034070  013737  001414  001412                MOV     @#$AC1,@#$AC0   ;PERFORM THE SQUARE
034076  004767  002124                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
034102  174137  001312                        STF     AC1,@#$TMP4     ;SAVE RESULT
034106  013737  001414  001262                MOV     @#$AC1,@#$REG2  ;AND SOFTWARE EXP
                                        ;
                                        ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
                                        ;DO THE A*A FIRST
034114  013737  001256  001412                MOV     @#$REG0,@#$AC0  ;GET EXT EXPONENT
034122  172437  001302                        LDF     @#$TMP0,AC0     ;LOAD OPERAND A
034126  013737  001412  001414                MOV     @#$AC0,@#$AC1   ;SET OPERAND B EXT EXPONENT
034134  172500                                LDF     AC0,AC1         ;LOAD B OPERAND
034136  004767  002064                        JSR     PC,FLTMPY       ;EXECUTE THE MULTIPLY
034142  174102                                STF     AC1,AC2         ;SAVE RESULT
034144  013737  001414  001416                MOV     @#$AC1,@#$AC2
                                        ;
                                        ;NOW DO THE B*B
034152  172437  001306                        LDF     @#$TMP2,AC0     ;LOAD B OPERAND
034156  172500                                LDF     AC0,AC1
034160  013737  001260  001412                MOV     @#$REG1,@#$AC0  ;AND EXT EXPONENT
034166  013737  001412  001414                MOV     @#$AC0,@#$AC1
034174  004767  002026                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
034200  174103                                STF     AC1,AC3         ;SAVE THE RESULT
```

```
6376  034202  013737  001414  001420          MOV     @#$AC1,@#$AC3
6377
6378                                          ;
6379                                          ;NOW DO THE 2*B*A
6379  034210  012701  001306                  MOV     #$TMP2,R1
6380  034214  172411                          LDF     (R1),AC0        ;LOAD THE B OPERAND
6381  034216  172541                          LDF     -(R1),AC1       ;LOAD THE A OPERAND
6382  034220  013737  001260  001412          MOV     @#$REG1,@#$AC0  ;AND THE EXT EXPONENTS
6383  034226  013737  001256  001414          MOV     @#$REG0,@#$AC1
6384  034234  004767  001766                  JSR     PC,FLTMPY       ;DO THE MULTIPLY
6385  034240  172427  040000                  LDF     #^040000,AC0    ;SETUP TO MULTIPLY BY TWO
6386  034244  012737  000002  001412          MOV     #2,@#$AC0
6387  034252  004767  001750                  JSR     PC,FLTMPY       ;DO THE MULTIPLY
6388                                          ;
6389                                          ;NOW SUM THE RESULTS
6390  034256  013737  001420  001412          MOV     @#$AC3,@#$AC0
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     @#$AC2,@#$AC0
6395  034302  004767  001770                  JSR     PC,FLTADD       ;ADD THIS RESULT
6396  034306  174137  001316                  STF     AC1,@#$TMP6     ;SAVE FINAL RESULT
6397  034312  013737  001414  001264          MOV     @#$AC1,@#$REG3
6398                                          ;
6399                                          ;NOW CHECK BOTH SIDES OF THE EQUATION
6400                                          ;CALCULATE THE NUMBER OF CORRECT BITS
6401                                          ;PUT LARGEST EXPONENT OF A**2 OR B**2 IN $AC2
6402  034320  023737  001416  001420          CMP     @#$AC2,@#$AC3
6403  034326  002003                          BGE     1$              ;BRANCH IF $AC2 ALREADY HAS LARGEST
6404  034330  013737  001420  001416          MOV     @#$AC3,@#$AC2   ;$AC3 WAS LARGER
6405  034336  163737  001414  001416  1$:     SUB     @#$AC1,@#$AC2   ;NOW CALCULATE NUMBER
6406  034344  162737  000023  001416          SUB     #19.,@#$AC2     ;OF CORRECT BITS WITHIN 2
6407  034352  005437  001416                  NEG     @#$AC2          ;MAKE RESULT POSITIVE
6408  034356  172437  001312                  LDF     @#$TMP4,AC0     ;LOAD RESULT OF LEFT HAND SIDE
6409  034362  013737  001262  001412          MOV     @#$REG2,@#$AC0  ;AND EXTENDED EXPONENT
6410  034370  004767  001676                  JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6411  034374  163737  001264  001414          SUB     @#$REG3,@#$AC1  ;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     @#$AC1
6415  034410  023737  001416  001414  3$:     CMP     @#$AC2,@#$AC1   ;ANSWERS WITHIN ALLOWABLE NUMBER?
6416  034416  003401                          BLE     SECT2           ;BRANCH IF YES
6417  034420  104014                  4$:     ERROR   14              ;RESULTS ARE WRONG
6418                                          ;;**************************************************************
6419  034422  170127  000000          -       SECT2:  LDFPS   #0
6420                                          ;DO A+B
6421  034426  172537  001302                  LDF     @#$TMP0,AC1     ;LOAD A OPERAND
6422  034432  172437  001306                  LDF     @#$TMP2,AC0     ;LOAD B OPERAND
6423  034436  013737  001256  001414          MOV     @#$REG0,@#$AC1
6424  034444  013737  001260  001412          MOV     @#$REG1,@#$AC0
6425  034452  004767  001620                  JSR     PC,FLTADD       ;ADD THEM
6426  034456  174102                          STF     AC1,AC2         ;SAVE IN AC2
6427  034460  013737  001414  001416          MOV     @#$AC1,@#$AC2   ;AND EXT EXPONENT
6428                                          ;NOW DO THE A-B
6429  034466  172537  001302                  LDF     @#$TMP0,AC1     ;LOAD OPERAND A
6430  034472  013737  001256  001414          MOV     @#$REG0,@#$AC1  ;AND EXT EXPONENT
6431  034500  172437  001306                  LDF     @#$TMP2,AC0     ;LOAD OPERAND B
```

```
6432  034504  013737  001260  001412          MOV     @#$REG1,@#$AC0
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     @#$AC2,@#$AC0
6437  034526  004767  001474                  JSR     PC,FLTMPY       ;FORM THE PRODUCT
6438  034532  174137  001312                  STF     AC1,@#$TMP4     ;SAVE RESULT
6439  034536  013737  001414  001262          MOV     @#$AC1,@#$REG2  ;AND EXT EXPONENT
6440                                  ;NOW DO THE B*B
6441  034544  172437  001306                  LDF     @#$TMP2,AC0     ;LOAD OPERAND B
6442  034550  013737  001260  001412          MOV     @#$REG1,@#$AC0
6443  034556  172500                          LDF     AC0,AC1         ;B OPERAND IS IN AC0
6444  034560  013737  001412  001414          MOV     @#$AC0,@#$AC1   ;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     @#$AC1,@#$AC2
6448                                  ;NOW DO THE A*A
6449  034602  172437  001302                  LDF     @#$TMP0,AC0     ;LOAD OPERAND A
6450  034606  013737  001256  001412          MOV     @#$REG0,@#$AC0
6451  034614  172500                          LDF     AC0,AC1
6452  034616  013737  001412  001414          MOV     @#$AC0,@#$AC1
6453  034624  004767  001376                  JSR     PC,FLTMPY       ;EXECUTE THE MULTIPLY
6454  034630  013737  001414  001420          MOV     @#$AC1,@#$AC3   ;SAVE EXT EXPO OF A*A
6455                                  ;NOW DO A**2-B**2
6456  034636  172402                          LDF     AC2,AC0         ;GET B*B
6457  034640  013737  001416  001412          MOV     @#$AC2,@#$AC0   ;A*A IN AC1
6458  034646  004767  001420                  JSR     PC,FLTSUB
6459  034652  174137  001316                  STF     AC1,@#$TMP6     ;SAVE IN MEMORY
6460  034656  013737  001414  001264          MOV     @#$AC1,@#$REG3
6461                                  ;NOW COMPUTE THE RESULTS
6462                                  ;CALCULATE THE NUMBER OF CORRECT BITS
6463  034664  023737  001416  001420          CMP     @#$AC2,@#$AC3   ;DETERMINE WHICH EXP IS LARGER
6464  034672  002003                          BGE     2$              ;BRANCH IF AC2 LARGER
6465  034674  013737  001420  001416          MOV     @#$AC3,@#$AC2   ;PUT LARGEST IN AC2
6466  034702  163737  001414  001416  2$:     SUB     @#$AC1,@#$AC2
6467  034710  162737  000025  001416          SUB     #21.,@#$AC2
6468  034716  005437  001416                  NEG     @#$AC2
6469  034722  172437  001312                  LDF     @#$TMP4,AC0     ;GET LEFT HAND SIDE
6470  034726  013737  001262  001412          MOV     @#$REG2,@#$AC0
6471  034734  004767  001332                  JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6472  034740  163737  001264  001414          SUB     @#$REG3,@#$AC1  ;SUB EXT EXPONENTS
6473                                          ;ACTUAL EXPONENTS ARE EQUAL
6474  034746  100002                          BPL     1$              ;MAKE SURE RESULT IS POSITIVE
6475  034750  005437  001414                  NEG     @#$AC1
6476  034754  023737  001416  001414  1$:     CMP     @#$AC2,@#$AC1   ;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     @#$TMP0,AC1     ;LOAD OPERAND A
6482  034772  172437  001306                  LDF     @#$TMP2,AC0     ;AND OPERAND B
6483  034776  013737  001256  001414          MOV     @#$REG0,@#$AC1
6484  035004  013737  001260  001412          MOV     @#$REG1,@#$AC0
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,@#$TMP4     ;SAVE RESULT
```

```
 6488  035026  013737  001414  001262          MOV     @#$AC1,@#$REG2
 6489  035034  172437  001302                  LDF     @#$TMP0,AC0     ;LOAD OPERAND A
 6490  035040  174037  001316                  STF     AC0,@#$TMP6     ;SAVE INCASE TYPE OUT
 6491  035044  013737  001256  001412          MOV     @#$REG0,@#$AC0
 6492  035052  013737  001256  001264          MOV     @#$REG0,@#$REG3
 6493  035060  004767  001206                  JSR     PC,FLTSUB       ;SUBTRACT RIGHT AND LEFT HAND SIDES
 6494  035064  163737  001256  001414          SUB     @#$REG0,@#$AC1  ;SEE IF RESULT OK
 6495  035072  100002                          BPL     1$              ;ENSURE DIFFERANCE IS POSITIVE
 6496  035074  005437  001414                  NEG     @#$AC1
 6497  035100  022737  000026  001414  1$:     CMP     #22.,@#$AC1     ;RESULTS WITHIN 2 BITS?
 6498  035106  003001                          BGT     2$              ;BRANCH IF NO
 6499  035110  000401                          BR      TST73           ;;GO TO NEXT TEST
 6500  035112  104014                  2$:     ERROR   14              ;RESULTS WRONG
 6501
 6502                                          ;;*****************************************************************
 6503                                          ;*TEST 73       FLOATING POINT TEST 2
 6504                                          ;*
 6505                                          ;*      THIS TEST TAKES TWO RANDOM NUMBERS (A AND B) AND
 6506                                          ;*      COMPARES THE RESULTS OF TWO EQUAL CALCULATIONS.
 6507                                          ;*      EACH SECTION EVALUATES A DIFFERENT EQUATION AS DESCRIBED BELOW:
 6508                                          ;*      SECT1   (A+B)**2=A**2+2*A*B+B**2
 6509                                          ;*      SECT2   (A+B)*(A-B)=A**2-B**2
 6510                                          ;*      SECT3   A/B*B=A
 6511                                          ;;*****************************************************************
 6512  035114  000004                  TST73:  SCOPE
 6513  035116  112737  000073  001202          MOVB    #73,@#$TSTNM            ;LOAD TEST NUMBER
 6514  035124  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
 6515  035132  012737  000001  001326          MOV     #1,@#$TIMES
 6516  035140  004737  060534          100$:   JSR     PC,@#FLTDBL     ;GET RANDOM OPERANDS
 6517  035144  170127  000200                  LDFPS   #200            ;INIT FPS
 6518  035150  172537  001302                  LDF     @#$TMP0,AC1     ;LOAD A OPERAND
 6519  035154  172437  001312                  LDF     @#$TMP4,AC0     ;LOAD B OPERAND
 6520  035160  013737  001256  001414          MOV     @#$REG0,@#$AC1  ;SETUP EXTENDED
 6521  035166  013737  001260  001412          MOV     @#$REG1,@#$AC0  ;EXPONENTS
 6522  035174  004767  001076                  JSR     PC,FLTADD       ;PERFORM THE ADD
 6523  035200  174100                          STF     AC1,AC0         ;SETUP AC0 TO
 6524  035202  013737  001414  001412          MOV     @#$AC1,@#$AC0   ;PERFORM THE SQUARE
 6525  035210  004767  001012                  JSR     PC,FLTMPY       ;DO THE MULTIPLY
 6526  035214  174137  001432                  STF     AC1,@#FLTMP0    ;SAVE RESULT
 6527  035220  013737  001414  001262          MOV     @#$AC1,@#$REG2  ;AND SOFTWARE EXP
 6528                                          ;
 6529                                          ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
 6530                                          ;DO THE A*A FIRST
 6531  035226  013737  001256  001412          MOV     @#$REG0,@#$AC0  ;GET EXT EXPONENT
 6532  035234  172437  001302                  LDF     @#$TMP0,AC0     ;LOAD OPERAND A
 6533  035240  013737  001412  001414          MOV     @#$AC0,@#$AC1   ;SET OPERAND B EXT EXPONENT
 6534  035246  172500                          LDF     AC0,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     @#$AC1,@#$AC2
 6538                                          ;
 6539                                          ;NOW DO THE B*B
 6540  035264  172437  001312                  LDF     @#$TMP4,AC0     ;LOAD B OPERAND
 6541  035270  172500                          LDF     AC0,AC1
 6542  035272  013737  001260  001412          MOV     @#$REG1,@#$AC0  ;AND EXT EXPONENT
 6543  035300  013737  001412  001414          MOV     @#$AC0,@#$AC1
```

```
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     @#$AC1,@#$AC3
6547                                  ;
6548                                  ;NOW DO THE 2*B*A
6549  035322  012701  001312              MOV     #$TMP4,R1
6550  035326  172411                      LDF     (R1),AC0        ;LOAD THE B OPERAND
6551  035330  172541                      LDF     -(R1),AC1       ;LOAD THE A OPERAND
6552  035332  013737  001260  001412      MOV     @#$REG1,@#$AC0  ;AND THE EXT EXPONENTS
6553  035340  013737  001256  001414      MOV     @#$REG0,@#$AC1
6554  035346  004767  000654              JSR     PC,FLTMPY       ;DO THE MULTIPLY
6555  035352  172427  040000              LDF     #^O40000,AC0    ;SETUP TO MULTIPLY BY TWO
6556  035356  012737  000002  001412      MOV     #2.,@#$AC0
6557  035364  004767  000636              JSR     PC,FLTMPY       ;DO THE MULTIPLY
6558                                  ;
6559                                  ;NOW SUM THE RESULTS
6560  035370  013737  001420  001412      MOV     @#$AC3,@#$AC0
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     @#$AC2,@#$AC0
6565  035414  004767  000656              JSR     PC,FLTADD       ;ADD THIS RESULT
6566  035420  174137  001442              STF     AC1,@#FLTMP1    ;SAVE FINAL RESULT
6567  035424  013737  001414  001264      MOV     @#$AC1,@#$REG3
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 $AC2
6572  035432  023737  001416  001420      CMP     @#$AC2,@#$AC3
6573  035440  002003                      BGE     1$              ;BRANCH IF $AC2 ALREADY HAS LARGEST
6574  035442  013737  001420  001416      MOV     @#$AC3,@#$AC2   ;$AC3 WAS LARGER
6575  035450  163737  001414  001416  1$: SUB     @#$AC1,@#$AC2   ;NOW CALCULATE NUMBER
6576  035456  162737  000064  001416      SUB     #52.,@#$AC2     ;OF CORRECT BITS WITHIN 2
6577  035464  005437  001416              NEG     @#$AC2          ;MAKE RESULT POSITIVE
6578  035470  172437  001432              LDF     @#FLTMP0,AC0    ;LOAD RESULT OF LEFT HAND SIDE
6579  035474  013737  001262  001412      MOV     @#$REG2,@#$AC0  ;AND EXTENDED EXPONENT
6580  035502  004767  000564              JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
6581  035506  163737  001264  001414      SUB     @#$REG3,@#$AC1  ;GET DIFFERENCE IN EXT EXPONENTS
6582                                                              ;ACTUAL EXP'S ARE EQUAL TO 200
6583  035514  100002                      BPL     3$              ;ENSURE RESULT IS POSITIVE
6584  035516  005437  001414              NEG     @#$AC1
6585  035522  023737  001416  001414  3$: CMP     @#$AC2,@#$AC1   ;ANSWERS WITHIN ALLOWABLE NUMBER?
6586  035530  003401                      BLE     SECT2D          ;BRANCH IF YES
6587  035532  104016                  4$: ERROR   16              ;RESULTS ARE WRONG
6588                                  ;;*****************************************************************
6589  035534  170127  000200          SECT2D: LDFPS   #200
6590                                  ;DO A+B
6591  035540  172537  001302              LDF     @#$TMP0,AC1     ;LOAD A OPERAND
6592  035544  172437  001312              LDF     @#$TMP4,AC0     ;LOAD B OPERAND
6593  035550  013737  001256  001414      MOV     @#$REG0,@#$AC1
6594  035556  013737  001260  001412      MOV     @#$REG1,@#$AC0
6595  035564  004767  000506              JSR     PC,FLTADD       ;ADD THEM
6596  035570  174102                      STF     AC1,AC2         ;SAVE IN AC2
6597  035572  013737  001414  001416      MOV     @#$AC1,@#$AC2   ;AND EXT EXPONENT
6598                                  ;NOW DO THE A-B
6599  035600  172537  001302              LDF     @#$TMP0,AC1     ;LOAD OPERAND A
```

```
 6600  035604  013737  001256  001414          MOV     @#$REG0,@#$AC1  ;AND EXT EXPONENT
 6601  035612  172437  001312                  LDF     @#$TMP4,AC0     ;LOAD OPERAND B
 6602  035616  013737  001260  001412          MOV     @#$REG1,@#$AC0
 6603  035624  004767  000442                  JSR     PC,FLTSUB       ;SUBTRACT THEM
 6604                                  ;NOW DO (A+B)*(A-B)
 6605  035630  172402                          LDF     AC2,AC0         ;GET RESULT OF (A+B)
 6606  035632  013737  001416  001412          MOV     @#$AC2,@#$AC0
 6607  035640  004767  000362                  JSR     PC,FLTMPY       ;FORM THE PRODUCT
 6608  035644  174137  001432                  STF     AC1,@#FLTMP0    ;SAVE RESULT
 6609  035650  013737  001414  001262          MOV     @#$AC1,@#$REG2  ;AND EXT EXPONENT
 6610                                  ;NOW DO THE B*B
 6611  035656  172437  001312                  LDF     @#$TMP4,AC0     ;LOAD OPERAND B
 6612  035662  013737  001260  001412          MOV     @#$REG1,@#$AC0
 6613  035670  172500                          LDF     AC0,AC1         ;B OPERAND IS IN AC0
 6614  035672  013737  001412  001414          MOV     @#$AC0,@#$AC1   ;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     @#$AC1,@#$AC2
 6618                                  ;NOW DO THE A*A
 6619  035714  172437  001302                  LDF     @#$TMP0,AC0     ;LOAD OPERAND A
 6620  035720  013737  001256  001412          MOV     @#$REG0,@#$AC0
 6621  035726  172500                          LDF     AC0,AC1
 6622  035730  013737  001412  001414          MOV     @#$AC0,@#$AC1
 6623  035736  004767  000264                  JSR     PC,FLTMPY       ;EXECUTE THE MULTIPLY
 6624  035742  013737  001414  001420          MOV     @#$AC1,@#$AC3   ;SAVE EXT EXPO OF A*A
 6625                                  ;NOW DO A**2-B**2
 6626  035750  172402                          LDF     AC2,AC0         ;GET B*B
 6627  035752  013737  001416  001412          MOV     @#$AC2,@#$AC0   ;A*A IN AC1
 6628  035760  004767  000306                  JSR     PC,FLTSUB
 6629  035764  174137  001442                  STF     AC1,@#FLTMP1    ;SAVE IN MEMORY
 6630  035770  013737  001414  001264          MOV     @#$AC1,@#$REG3
 6631                                  ;NOW COMPUTE THE RESULTS
 6632                                  ;CALCULATE THE NUMBER OF CORRECT BITS
 6633  035776  023737  001416  001420          CMP     @#$AC2,@#$AC3   ;DETERMINE WHICH EXP IS LARGER
 6634  036004  002003                          BGE     2$              ;BRANCH IF AC2 LARGER
 6635  036006  013737  001420  001416          MOV     @#$AC3,@#$AC2   ;PUT LARGEST IN AC2
 6636  036014  163737  001414  001416  2$:     SUB     @#$AC1,@#$AC2
 6637  036022  162737  000065  001416          SUB     #53.,@#$AC2
 6638  036030  005437  001416                  NEG     @#$AC2
 6639  036034  172437  001432                  LDF     @#FLTMP0,AC0    ;GET LEFT HAND SIDE
 6640  036040  013737  001262  001412          MOV     @#$REG2,@#$AC0
 6641  036046  004767  000220                  JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
 6642  036052  163737  001264  001414          SUB     @#$REG3,@#$AC1  ;SUB EXT EXPONENTS
 6643                                                                  ;ACTUAL EXPONENTS ARE EQUAL
 6644  036060  100002                          BPL     1$              ;MAKE SURE RESULT IS POSITIVE
 6645  036062  005437  001414                  NEG     @#$AC1
 6646  036066  023737  001416  001414  1$:     CMP     @#$AC2,@#$AC1   ;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     @#$TMP0,AC1     ;LOAD OPERAND A
 6652  036104  172437  001312                  LDF     @#$TMP4,AC0     ;AND OPERAND B
 6653  036110  013737  001256  001414          MOV     @#$REG0,@#$AC1
 6654  036116  013737  001260  001412          MOV     @#$REG1,@#$AC0
 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     @#$AC1,@#$REG2
6659  036146  172437  001302                LDF     @#$TMP0,AC0     ;LOAD OPERAND A
6660  036152  174037  001442                STF     AC0,@#FLTMP1    ;SAVE INCASE TYPE OUT
6661  036156  013737  001256  001412        MOV     @#$REG0,@#$AC0
6662  036164  013737  001256  001264        MOV     @#$REG0,@#$REG3
6663  036172  004767  000074                JSR     PC,FLTSUB       ;SUBTRACT RIGHT AND LEFT HAND SIDES
6664  036176  163737  001256  001414        SUB     @#$REG0,@#$AC1  ;SEE IF RESULT OK
6665  036204  100002                        BPL     1$              ;ENSURE DIFFERANCE IS POSITIVE
6666  036206  005437  001414                NEG     @#$AC1
6667  036212  022737  000066  001414  1$:   CMP     #54.,@#$AC1     ;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 AC0 AND AC1
6675                                  ;*      AND LEAVES THE RESULT IN AC1. IT ALSO TAKES CARE OF
6676                                  ;*      THE SOFTWARE EXPONENTS THAT ARE KEPT IN $AC0 AND $AC1.
6677                                  ;;*****************************************************************
6678  036226  063737  001412  001414  FLTMPY: ADD     @#$AC0,@#$AC1   ;ADD SOFTWARE EXPONENTS
6679  036234  171100                        MULF    AC0,AC1         ;DO THE MULTIPLY
6680  036236  012746  100400                MOV     #100400,-(SP)   ;PUT CONTROL WORD ON STACK
6681  036242  004737  060674                JSR     PC,@#EXPEXT     ;CALCULATE EXT EXPONENT
6682  036246  000207                  1$:   RTS     PC              ;RETURN
6683
6684                                  ;;*****************************************************************
6685                                  .SBTTL  FLOATING POINT DIVIDE ROUTINE
6686                                  ;*      THIS ROUTINE DIVIDES THE CONTENTS OF AC1 BY AC0
6687                                  ;*      AND LEAVES THE RESULT IN AC1.
6688                                  ;;*****************************************************************
6689  036250  163737  001412  001414  FLTDIV: SUB     @#$AC0,@#$AC1   ;ADJUST SOFTWARE EXPONENTS
6690  036256  174500                        DIVF    AC0,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$:   RTS     PC              ;RETURN
6694
6695                                  ;;*****************************************************************
6696                                  .SBTTL  FLOATING POINT ADD ROUTINE
6697                                  ;*      THIS ROUTINE ADDS THE CONTENTS OF AC0 TO AC1.
6698                                  ;*      THIS CAN ONLY BE DONE IF THE SOFTWARE EXPONENTS
6699                                  ;*      ARE CLOSE ENOUGH TOGETHER SUCH THAT AN ADJUSTMENT
6700                                  ;*      OF THE REAL EXPONENT LEAVES A NON-ZERO NUMBER.
6701                                  ;;*****************************************************************
6702  036272  010667  000134          FLTSUB: MOV     SP,SUBFLG       ;SET SUBTRACT FLAG
6703  036276  023737  001412  001414  FLTADD: CMP     @#$AC0,@#$AC1   ;CHECK SOFTWARE EXPONENTS
6704  036304  003016                        BGT     1$
6705  036306  001434                        BEQ     2$
6706                                  ;ACCUMULATOR 1 IS LARGER THAN ACCUMULATOR 0
6707  036310  013702  001414                MOV     @#$AC1,R2       ;GET OPERAND B SOFTWARE EXP
6708  036314  163702  001412                SUB     @#$AC0,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                          NEG     R2
 6713  036330  176402                          LDEXP   R2,AC0           ;RELOAD THE EXPONENT
 6714  036332  000422                          BR      2$
 6715  036334  176427  177703          7$:     LDEXP   #-75,AC0          ;FAKE EXPONENT SO HARDWARE
 6716  036340  000417                          BR      2$                ;WILL DETECT OUT OF RANGE
 6717                                  ;
 6718                                  ;ACCUMULATOR 0 IS LARGER THAN ACCUMULATOR 1
 6719  036342  013702  001412          1$:     MOV     @#$AC0,R2         ;GET SOFTWARE EXP OF OPERAND A
 6720  036346  163702  001414                  SUB     @#$AC1,R2         ;GET DIFFERENCE IN EXP'S
 6721  036352  013737  001412  001414          MOV     @#$AC0,@#$AC1     ;MAKE SOFTWARE EXP'S EQUAL
 6722  036360  020227  000071                  CMP     R2,#57. ;EXP WITHIN DBL PREC RANGE?
 6723  036364  002003                          BGE     4$                ;BRANCH IF NO
 6724  036366  005402                          NEG     R2
 6725  036370  176502                          LDEXP   R2,AC1            ;RELOAD THE EXPONENT
 6726  036372  000402                          BR      2$
 6727                                  ;
 6728                                  ;ACCUMULATOR 0 !S MUCH LARGER THAN ACCUMULATOR 1 SO RESULT IS 0
 6729  036374  176527  177703          4$:     LDEXP   #-75,AC1          ;FAKE EXPONENT SO HARDWARE
 6730                                                                    ;WILL DETECT OUT OF RANGE
 6731  036400  005767  000026          2$:     TST     SUBFLG            ;ADD OR SUBTRACT?
 6732  036404  001402                          BEQ     5$                ;BRANCH IF ADD
 6733  036406  173100                          SUBF    AC0,AC1
 6734  036410  000401                          BR      6$
 6735  036412  172100                  5$:     ADDF    AC0,AC1           ;EXECUTE THE ADD
 6736  036414  012746  100400          6$:     MOV     #100400,-(SP)     ;PUT CONTROL WORD ON STACK
 6737  036420  004737  060674                  JSR     PC,@#EXPEXT       ;CALCULATE EXT EXPONENT
 6738  036424  005067  000002          3$:     CLR     SUBFLG            ;INIT SUBTRACT FLAG
 6739  036430  000207                          RTS     PC                ;RETURN
 6740  036432  000000                  SUBFLG: .WORD
 6741  036434  000004                  RELE8:  SCOPE
 6742  036436  010702                          MOV     PC,R2
 6743  036440  062702  000012                  ADD     #12,R2
 6744  036444  012707  043764                  MOV     #RELOC,PC         ;GO RELOCATE PROGRAM CODE
 6745  036450  000000                  REL88:  .WORD   0
 6746                                  ;8888888888888 LAST ADDRESS OF CODE TO BE RELOCATED 88888888888
 6747
 6748
 6749
 6750  036452  005737  001502                  TST     @#KB11E           ;IS THIS A KB11-E OR KB11-EM?
 6751  036456  001002                          BNE     DOMFPT            ;BR IF IT IS
 6752  036460  000167  004060                  JMP     ENDCIS            ;SKIP THIS TEST IF NOT. MFPT AND CIS NOT THERE
 6753  036464                          DOMFPT:
 6754                                  ;;*********************************************************************
 6755                                  ;*TEST 74       CHECK MFPT INSTRUCTION (KB11-E/EM ONLY)
 6756                                  ;*      THE MFPT INSTRUCTION IS NOT AVAILABLE ON THE KB11-B/C/CM BUT
 6757                                  ;*      IF THIS IS A KB11-E/EM THIS TEST IS RUN.  MFPT RETURNS
 6758                                  ;*      DATA TO R0 IN THE FOLLOWING FORMAT:
 6759                                  ;*      BIT 0 - 1  INDICATES 11/44 CPU
 6760                                  ;*      BIT 1 - 1 INDICATES KB11-E/EM CPU (SHOULD ALWAYS COME UP IN THIS TEST)
 6761                                  ;*      BIT 8 - 1 INDICATES CISP PRESENT
 6762                                  ;*      BIT 9 - 1 INDICATES FP PRESENT
 6763                                  ;;*********************************************************************
 6764  036464  012767  000001  142634          MOV     #1,$TIMES         ;;DO 1 ITERATION
 6765  036472  000004                  TST74:  SCOPE
 6766  036474  112737  000074  001202          MOVB    #74,@#$TSTNM              ;LOAD TEST NUMBER
 6767  036502  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY        ;;DISPLAY TEST NUMBER
```

```
6768
6769                                            .SBTTL  START OF SECTION 9
6770                                    ;9999999999999 FIRST ADDRESS TO BE RELOCATED 999999999
6771  036510  010700                    REL9:   MOV     PC,R0           ;GET PC
6772  036512  005740                            TST     -(R0)           ;R0 CONTAINS THE ADDRESS OF REL9
6773  036514  010037  001540                    MOV     R0,@#FRSTAD     ;SAVE
6774  036520  010700                            MOV     PC,R0           ;GET CURRENT PC
6775  036522  162700  036522                    SUB     #.,R0           ;SUBTRACT RELOCATION FACTOR
6776  036526  010037  001534                    MOV     R0,@#FACTOR     ;SAVE RELOCATION FACTOR
6777  036532  010737  001212                    MOV     PC,@#$LPERR     ;SET LOOP ADDRESS
6778  036536  062737  000030  001212          ADD     #30,@#$LPERR    ;ADJUST
6779  036544  013737  001212  001210          MOV     @#$LPERR,@#$LPADR
6780  036552  105737  001530                    TSTB    @#NEXEC         ;BR IF TEST CODE TO BE EXECUTED
6781  036556  001402                            BEQ     .+6
6782  036560  000167  003742                    JMP     RELE9
6783  036564  012703  000002                    MOV     #2,R3           ;R3 IS DATA PATTERN. BIT 1 WILL ALWAYS BE SET
6784  036570  105737  001505                    TSTB    @#CISP          ;CISP FOUND?
6785  036574  001402                            BEQ     1$              ;BR IF NOT
6786  036576  052703  000400                    BIS     #400,R3         ;BIT 8 SHOULD BE SET FOR CISP
6787  036602  032737  020000  001500  1$:       BIT     #20000,@#OPT.CP         ;FP FOUND?
6788  036610  001402                            BEQ     2$              ;BR IF NOT
6789  036612  052703  001000                    BIS     #1000,R3        ;BIT 9 WILL BE SET FOR FP
6790  036616  000007                    2$:     MFPT                    ;EXECUTE INSTRUCTION
6791  036620  020003                            CMP     R0,R3           ;MATCH?
6792  036622  001405                            BEQ     DONE7           ;DONE IF SO
6793  036624  010337  001302                    MOV     R3,@#$TMP0      ;SET UP EXPECTED (GOOD) DATA
6794  036630  010037  001304                    MOV     R0,@#$TMP1      ;SET UP RECIEVED (BAD) DATA
6795  036634  104017                            ERROR   17              ;ERROR PRINTOUT
6796  036636  105737  001505            DONE7:  TSTB    @#CISP                  ;IS CISP PRESENT?
6797  036642  001002                            BNE     DOCIS                   ;BR IF IT IS
6798  036644  000167  003656                    JMP     RELE9                   ;SKIP CIS TEST BUT RELOCATE SECTION FOR MFPT
6799  036650                            DOCIS:
6800                                    ;;*****************************************************************
6801                                    ;*TEST 75        COMMERCIAL INSTRUCTION SET TEST
6802                                    ;;*****************************************************************
6803  036650  000004                    TST75:  SCOPE
6804  036652  112737  000075  001202            MOVB    #75,@#$TSTNM            ;LOAD TEST NUMBER
6805  036660  013737  001202  177570            MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
6806  036666  013767  001550  002272            MOV     @#$LONUM,SAVRNL         ;SAVE SEED LOW
6807  036674  013767  001552  002266            MOV     @#$HINUM,SAVRNH         ;SAVE SEED HIGH
6808  036702  062737  000036  001212            ADD     #30.,@#$LPERR           ;ADJUST LOOP ON ERROR POINTER
6809                                            ;START OF TEST TO MODIFY ABSOLUTE ADDRESSES WHEN RELOCATING
6810  036710  016737  002252  001550            MOV     SAVRNL,@#$LONUM         ;RESTORE SEED LO
6811  036716  016737  002246  001552            MOV     SAVRNH,@#$HINUM         ;RESTORE SEED HI
6812  036724  012700  041266                    MOV     #OFFTAB,R0              ;GET OFFSET TO OFFSET TABLE
6813  036730  063700  001534                    ADD     @#FACTOR,R0             ;ADD FACTOR TO GET ADDRESS OF TABLE
6814  036734  012001                    1$:     MOV     (R0)+,R1                ;GET OFFSET OF STRING
6815  036736  001407                            BEQ     3$                      ;DONE IF OFFSET = 0
6816  036740  010102                            MOV     R1,R2                   ;SAVE R1
6817  036742  063702  001534                    ADD     @#FACTOR,R2             ;ADD FACTOR TO OFFSET
6818  036746  011112                            MOV     (R1),(R2)               ;MOVE ORIGINAL DATA TO ADDRESS
6819  036750  063712  001534                    ADD     @#FACTOR,(R2)           ;ADD FACTOR FOR ADDRESS
6820  036754  000767                            BR      1$                      ;GO MODIFY NEXT ADDRESS
6821                                            ;SET PASS COUNT AND GET RANDOM DATA
6822  036756  016700  002302            3$:     MOV     BUFFAD,R0               ;SET SRC ADDRES FOR DATA
6823  036762  016701  002276                    MOV     BUFFAD,R1               ;SET DST ADDRESS FOR DATA
```

```
6824  036766  062701  000310            ADD     #200.,R1                ;ADJUST
6825  036772  004737  060436  4$:       JSR     PC,@#$RAND              ;GET RANDOM NUMBER
6826  036776  013710  001550            MOV     @#$LONUM,(R0)           ;STORE NUMBER IN SOURCE ONE
6827  037002  042710  100200            BIC     #100200,(R0)            ;MAKE NUMBER BETWEEN 0 AND 177
6828  037006  012021                    MOV     (R0)+,(R1)+             ;STORE NUMBER IN DST FOR TEST CMPC
6829  037010  020067  002164            CMP     R0,DST.1A               ;DONE FILLING SOURCE ONE YET
6830  037014  002766                    BLT     4$                      ;NO GET NEXT RANDOM NUMBER
6831                                                                    ;YES GO TO FIRST TEST
6832                                    ;TEST CMPC 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     FNDSCN                  ;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  FNDSCN:   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
```

```
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                  MOVTCI                          ;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                  MOVTCI                          ;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 DESCIPTORS
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                  MOVC                            ;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                  MOVCI                           ;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                  L3D7                    ;LOAD DESCIPTORS FOR COMPARE
6993  037614  041172          SRC20:  .WORD   SRC.1D          ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
6994  037616  041176          DST20:  .WORD   DST.1D          ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
6995  037620  041254          CHAR5:  .WORD   CHAR
6996  037622  076044                  CMPC                    ;COMPARE STRINGS
6997  037624  001403                  BEQ     DECDAT          ;EQUAL NEXT TEST
6998  037626  004767  001010          JSR     PC,CISER        ;NOT EQUAL GET ERROR DATA
6999  037632  104020                  ERROR   20              ;REPORT ERROR
7000                                  ;DECIMAL ARITHMETIC TESTS
7001                                  ;SETUP DECIMAL DATA
7002  037634  016700  001424  DECDAT: MOV     BUFFAD,R0       ;GET BUFFAD TO R0 FOR INDEX
7003  037640  016705  001420          MOV     BUFFAD,R5
7004  037644  062705  000070          ADD     #56.,R5
7005  037650  005002          1$:     CLR     R2              ;CLR  R2 TO ACCUMULATE NIBBLES
7006  037652  012703  041132          MOV     #MSKTAB,R3      ;GET OFFSET OF MSKTAB
7007  037656  063703  001534          ADD     @#FACTOR,R3     ;ADJUST ADDRESS
7008  037662  011001                  MOV     (R0),R1         ;LOAD R1
7009  037664  005004                  CLR     R4              ;CLR FOR COUNTER
7010  037666  042301          2$:     BIC     (R3)+,R1        ;CLEAR OFF UNDESIRED NIBBLES
7011  037670  022301                  CMP     (R3)+,R1        ;IS NIBBLE LESS THAN 9
7012  037672  002001                  BGE     3$              ;YES DONT SUBTRACT
7013  037674  161301                  SUB     (R3),R1         ;GREATER THAN 9 SUB 6
7014  037676  005723          3$:     TST     (R3)+           ;INC R3 TWICE IF NO SUB
7015  037700  050102                  BIS     R1,R2           ;STORE NIBBLE IN R2
7016  037702  005204                  INC     R4              ;INC NIBBLE COUNT
7017  037704  022704  000004          CMP     #4,R4           ;4 NIBBLES DONE YET
7018  037710  002366                  BGE     2$              ;NO DO AGAIN
7019  037712  010220                  MOV     R2,(R0)+        ;STORE VALID DATA IN SOURCE
7020  037714  020500                  CMP     R5,R0
7021  037716  103354                  BHIS    1$              ;NO DO AGAIN
7022  037720  012701  041206                  MOV     #A.DSC,R1
                                                        ;SET DATA TYPE
7023  037724  063701  001534          ADD     @#FACTOR,R1     ;ADD FACTOR FOR ADDRESS
7024  037730  012702  041232                  MOV     #D.DSC,R2
                                                        ;GET OFFSET OF D.DSC
7025  037734  063702  001534          ADD     @#FACTOR,R2     ;ADJUST TO GET ADDRESS
7026  037740  042711  070000  4$:     BIC     #070000,(R1)    ;CLEAR TYPE BITS
7027  037744  052711  010000          BIS     #10000,(R1)     ;MAKE UNSIGNED ZONED DATA
7028  037750  062701  000004          ADD     #4,R1           ;GET NEXT DATA TYPE SPECIFIER
7029  037754  020102                  CMP     R1,R2           ;TEST FOR DONE
7030  037756  103770                  BLO     4$              ;NOT DONE DO AGAIN
7031                                  ;TEST COMPARE NUMERIC
7032  037760  076152                  CMPNI                   ;COMPARE EQUAL STRINGS
7033  037762  041206          A1:     .WORD   A.DSC           ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7034  037764  041206          A2:     .WORD   A.DSC           ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7035  037766  001403                  BEQ     NUMRIC          ;EQUAL NEXT TEST
7036  037770  004767  000720          JSR     PC,CISER3       ;GET ERROR DATA
7037  037774  104020                  ERROR   20              ;REPORT ERROR
7038                                  ;CALCULATE [(10A+10B)-10C]
7039  037776  076156  NUMRIC: ASHNI                          ;SHIFT A
7040  040000  041206          A3:     .WORD   A.DSC           ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7041  040002  041232          D1:     .WORD   D.DSC           ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7042  040004  000001                  .WORD   1               ;SHIFT COUNT
7043  040006  076156                  ASHNI                   ;SHIFT B
7044  040010  041212          B1:     .WORD   B.DSC           ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7045  040012  041222          E1:     .WORD   E.DSC           ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7046  040014  000001                  .WORD   1               ;SHIFT COUNT
7047  040016  076150                  ADDNI                   ;10A+10B
```

```
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  076027                        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                        CMPN                            ;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: CVTLNI                          ;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                        CVTNLI                          ;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        CMP     LONG.1,LONG.2           ;CHECK FIRST HALF #LONG WORD
7093  040146  001004                        BNE     1$                      ;NOT EQUAL ERROR
7094  040150  026767  001070  001072        CMP     LONG.1+2,LONG.2+2       ;EQUAL CHECK SECOND HALF
7095  040156  001403                        BEQ     NUMPAC                  ;EQUAL NEXT TEST
7096  040160  004767  000474        1$:     JSR     PC,CISER1               ;GET ERROR DATA
7097  040164  104020                        ERROR   20                      ;REPORT ERROR
7098                                        ;CONVERT NUM TO PACK TO NUM
7099  040166  076155                NUMPAC: CVTNPI                          ;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                        CVTPNI                          ;CONVERT BACK TO NUM
7103  040176  041222                E7:     .WORD   E.DSC                   ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
```

```
7104  040200  041226              F7:     .WORD   F.DSC                   ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7105  040202  076156                      ASHNI                           ;TRANSFER A RO E 32 BYTES
7106  040204  041206              A6:     .WORD   A.DSC                   ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7107  040206  041222              E8:     .WORD   E.DSC                   ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7108  040210  000000                      .WORD   0                       ;SHIFT COUNT
7109  040212  076152                      CMPNI                           ;COMPARE RESULTS
7110  040214  041222              E9:     .WORD   E.DSC                   ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7111  040216  041226              F8:     .WORD   F.DSC                   ;THIS LOCATION MODIFIED WHEN TEST RELOCATES
7112  040220  001403                      BEQ     PACDAT                  ;EQUAL NEXT TEST
7113  040222  004767  000450              JSR     PC,CISER2               ;GO GET ERROR DATA
7114  040226  104020                      ERROR   20                      ;REPORT ERROR
7115                                      ;PACKED DECIMAL ARITHMETIC
7116  040230  012701  041206      PACDAT: MOV     #A.DSC,R1               ;SET DATA TYPE
7117  040234  063701  001534              ADD     @#FACTOR,R1             ;ADJUST FOR ADDRESS
7118  040240  012702  041232              MOV     #D.DSC,R2               ;GET OFFSET TO D.DSC
7119  040244  063702  001534              ADD     @#FACTOR,R2             ;ADJUST FOR ADDRESS
```

```
7120  040250  042711  070000          1$:     BIC     #070000,(R1)            ;MAKE UNSIGNED PACKED DATA
7121  040254  052711  060000                  BIS     #060000,(R1)            ;SET TYPE BITS
7122  040260  062701  000004                  ADD     #4,R1                   ;NEXT DATA TYPE SPEC
7123  040264  020102                          CMP     R1,R2                   ;DONE YET
7124  040266  101770                          BLOS    1$                      ;NO DO AGAIN
7125  040270  146777  000666  000712          BICB    HIMASK,@A               ;CLR HI NIBBLE TO MAKE VALID PACKED STRING
7126  040276  146777  000660  000710          BICB    HIMASK,@B               ;CLR HI NIB OF B
7127  040304  146777  000652  000706          BICB    HIMASK,@C               ;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*SCRTCH
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*SCRTCH
7184                                         ;COMPARE RESULTS
7185  040516  076172                        CMPPI                           ;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                        CVTPLI                          ;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$:     JSR     PC,CISER1               ;GET PC,ERROR DATA
7205  040574  104020                        ERROR   20                      ;REPORT ERROR
7206  040576  000167  001724        DONE:   JMP     RELE9                   ;GO ON TO NEXT TEST
7207
7208                                         ;SUBROUTINE-SETUP
7209                                         ;SETS UP CHAR STRING DESCRIPTORS
7210                                         ;USAGE: JSR     PC,SETUP
7211                                         ;NO ARGUMENTS
7212  040602  012767  000310  000362 SETUP:  MOV     #200.,SRC.1D            ;SET SOURCE LENGTH
7213  040610  016767  000450  000356        MOV     BUFFAD,SRC.1A           ;SET SOURCE ADDRESS
7214  040616  012767  000310  000352        MOV     #200.,DST.1D            ;DEST LENGTH
7215  040624  016767  000434  000346        MOV     BUFFAD,DST.1A           ;DEST ADDRESS
7216  040632  062767  000310  000340        ADD     #200.,DST.1A            ;ADJUST FOR DST
7217  040640  000207                        RTS     PC                      ;RETURN
7218                                         ;SUBROUTINES-CISER, CISER1, CISER2 AND CISER3
7219                                         ;GETS SHOULD AND WAS DATA AND ADDRESSES
7220                                         ;USAGE: JSR     PC,CISER(X)
7221                                         ;NO ARGUMENTS
7222  040642  016737  000326  001302 CISER:  MOV     SRC.1A,@#$TMP0          ;SHOULD BE ADDRESS
7223  040650  016737  000324  001304        MOV     DST.1A,@#$TMP1          ;WAS ADDRESS
7224  040656  000207                        RTS     PC                      ;RETURN
7225  040660  016737  000356  001302 CISER1: MOV     LONG.1P,@#$TMP0         ;SHOULD BE ADDRESS
7226  040666  016737  000354  001304        MOV     LONG.2P,@#$TMP1         ;WAS ADDRESS
7227  040674  000207                        RTS     PC                      ;RETURN
7228  040676  016737  000322  001302 CISER2: MOV     E,@#$TMP0               ;SHOULD BE ADDRESS
7229  040704  016737  000320  001304        MOV     F,@#$TMP1               ;WAS ADRRESS
7230  040712  000207                        RTS     PC                      ;RETURN
7231  040714  016737  000270  001302 CISER3: MOV     A,@#$TMP0               ;SHOULD BE ADDRESS
```

```
7232  040722  016737  000262  001304          MOV     A,@#$TMP1               ;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
```

```
                                        ;MASK TABLES
                                        ;FOR MAKING VALID DECIMAL DATA

041132  177760          MSKTAB: .WORD   177760
041134  000011                  .WORD   11
041136  000006                  .WORD   6
041140  177417                  .WORD   177417
041142  000220          NINTAB: .WORD   220
041144  000140                  .WORD   140
041146  170377                  .WORD   170377
041150  004400                  .WORD   4400
041152  003000          SIXTAB: .WORD   3000
041154  007777                  .WORD   7777
041156  070000                  .WORD   70000
041160  100000                  .WORD   100000
041162     360          HIMASK: .BYTE   360
041163     014          SIGN:   .BYTE   014
041164     017          LOMASK: .BYTE   017
041166                  .EVEN
041166  000000          SAVRNL: .WORD   0
041170  000000          SAVRNH: .WORD   0

                                        ;CHARACTER STRING DESCRIPTOR TABLE

041172  000310          SRC.1D: .WORD   200.
041174  041706          SRC.1A: .WORD   BUFF
041176  000310          DST.1D: .WORD   200.
041200  042216          DST.1A: .WORD   BUFF+200.
041202  000031          OBJ.1D: .WORD   25.
041204  041770          OBJ.1A: .WORD   BUFF+50.
041206  000034          A.DSC:  .WORD   28.
041210  041706          A:      .WORD   BUFF
041212  000016          B.DSC:  .WORD   14.
041214  041742          B:      .WORD   BUFF+28.
041216  000012          C.DSC:  .WORD   10.
041220  041760          C:      .WORD   BUFF+42.
041222  000037          E.DSC:  .WORD   31.
041224  041776          E:      .WORD   BUFF+56.
041226  000037          F.DSC:  .WORD   31.
041230  042036          F:      .WORD   BUFF+88.
041232  000037          D.DSC:  .WORD   31.
041234  042076          D:      .WORD   BUFF+120.
041236  000001          SET.1D: .WORD   1
041240  040732          TRANS4: .WORD   TRANS
041242  001020  000000  LONG.1: .WORD   528.,0
041246  000002          LONG.2: .BLKW   2
041252  000001          ONE:    .WORD   1
041254  000040          CHAR:   .WORD   '
041256  040732          TRANS3: .WORD   TRANS
041260  041242          LONG1P: .WORD   LONG.1
041262  041246          LONG2P: .WORD   LONG.2
041264  041706          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   0                       ;TABLE TERMINATER
7398
7399
```

```
                                        ;BUFFER SPACE
                                        ;200 WORDS LONG
                                        ;USED FOR SOURCE AND DESTINATIONS

041706  000310                          BUFF:   .BLKW   200.

042526  000004                          RELE9:  SCOPE
042530  010702                                  MOV     PC,R2
042532  062702  000012                          ADD     #12,R2
042536  012707  043764                          MOV     #RELE9,PC       ;GO RELOCATE PROGRAM CODE
042542  000000                          RELE99: .WORD   0
                                         9999999999999 LAST ADDRESS OF CODE TO BE RELOCATED 99999999999

042544                                  ENDCIS:
                                        ;;*****************************************************************
                                        ;*TEST 76        TELETYPE AND CLOCK TESTS
                                        ;;*****************************************************************
042544  000240                          TST76:  NOP
042546  112737  000076  001202                  MOVB    #76,@#$TSTNM            ;LOAD TEST NUMBER
042554  013737  001202  177570                  MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
042562  113737  001202  177570                  MOVB    @#$TSTNM,@#SWR
042570  005037  001534                  TTYCHK: CLR     @#FACTOR
042574  012704  000100                          MOV     #100,R4         ;SET R4 = CONSTANT 100
042600  032737  000400  001500                  BIT     #TTOPT,@#OPT.CP ;BRANCH IF TTY
042606  001002                                  BNE     1$              ;ON SYSTEM
042610  000167  000204                          JMP     ARBFIN          ;JUMP IF NOT
042614  132777  000200  136422  1$:             BITB    #200,@$TPS      ;CHECK IF TTY IS READY
042622  001774                                  BEQ     1$
042624  012737  001553  001270                  MOV     #NULLS-1,@#$REG5;SET ADDRESS OF ASCII STRING TO TYPE
042632  106277  136406                          ASRB    @$TPS           ;SET IE BIT. SEE TPISR FOR INT SERVICE.
042636  000001                                  WAIT                    ;WAIT FOR INTERRUPT


042640                                  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).
042640  132737  000020  177776  1$:     BITB    #20,@#PSW
042646  001071                          BNE     ARBEX                   ;EXIT TEST IF 'T' BIT SET
042650  030477  136370          2$:     BIT     R4,@$TPS                ;WAIT FOR TTY TO BE NOT
042654  001375                          BNE     2$                      ;BUSY
042656  112737  000300  177776          MOVB    #300,@#PSW              ;SET PRIORITY LEVEL 6
042664  150477  136354          3$:     BISB    R4,@$TPS                ;SET IE BIT
042670  100375                          BPL     3$                      ;AND WAIT FOR READY
042672  032737  001000  001500          BIT     #LKOPT,@#OPT.CP         ;LINE CLOCK AVAILABLE?
042700  001447                          BEQ     ARBFIN                  ;BRANCH IF NO
042702  012737  042772  000064          MOV     #7$,@#TPVEC             ;SET TTY VECTOR
042710  012737  043004  000100          MOV     #8$,@#LKVEC             ;SET CLOCK VECTORS
042716  012737  000340  000102          MOV     #PR7,@#LKVEC+2
042724  005027                          CLR     (PC)+                   ;CLEAR CHECK WORD
042726  000000                  4$:     .WORD   0
042730  000240                          NOP
042732  000240                          NOP
042734  000240                          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                             ;AT THIS TIME BOTH THE CLOCK
7461                                                                          ;ARE READY TO INTERRUPT
7462  042752  105037  177776                  CLRB    @#PSW                   ;SET PRIORITY LEVEL 0
7463                                ;A CLOCK INTERRUPT WILL OCCUR (3$) AND LOC 4$ WILL BE INCREMENTED
7464                                ;AFTER THE CLOCK SERVICE A TTY INTERRUPT WILL OCCUR. THE TTY INT SERV
7465                                ;ICE WILL SHIFT LEFT 4$.
7466
7467  042756  022767  000002  177742          CMP     #2,4$                   ;CHECK THAT THE CLOCK
7468  042764  001415                          BEQ     ARBFIN                  ;& TTY INTERRUPTED IN
7469  042766  104000                          HLT                             ;THE PROPER SEQUENCE
7470  042770  000413                          BR      ARBFIN
7471
7472  042772  005077  136246          7$:     CLR     @$TPS                   ;CLEAR IE BIT
7473  042776  006367  177724                  ASL     4$                      ;SHIFT INDICATOR
7474  043002  000002                          RTI                             ;RETURN
7475
7476  043004  005267  177716          8$:     INC     4$
7477  043010  012737  054300  000100          MOV     #LKSRV,@#LKVEC          ;SET CLOCK VECTORS
7478  043016  000002                          RTI
7479
7480
7481  043020  012737  063434  000064  ARBFIN: MOV     #TPISR,@#TPVEC          ;RESTORE TTY VECTOR
7482  043026  005077  136212                  CLR     @$TPS                   ;CLEAR IE BIT
7483  043032                          ARBEX:
7484                                ;;**************************************************************************
7485                                ;*TEST 77       TURN ON UBE AND MBT
7486                                ;*      TURN ON THE MASS BUS TESTER AND UNIBUS EXERCISER IF PRESENT
7487                                ;;**************************************************************************
7488  043032  000240                  TST77:  NOP
7489  043034  112737  000077  001202          MOVB    #77,@#$TSTNM            ;LOAD TEST NUMBER
7490  043042  013737  001202  177570          MOV     @#$TSTNM,@#DISPLAY      ;;DISPLAY TEST NUMBER
7491  043050  113737  001202  177570          MOVB    @#$TSTNM,@#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     #LKSRV,@#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 UBE?
7503  043124  001011                          BNE     MBTSET                  ;BRANCH IF YES
7504  043126  032737  000040  172516          BIT     #BIT5,@#MMR3            ;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 AVAILABLE?
```

```
7512  043156  001434                          BEQ     STMM            ;BRANCH IF NO
7513  043160  032737  000010  177570          BIT     #SW3,@#SWR      ;INHIBIT MBT?
7514  043166  001030                          BNE     STMM            ;BRANCH IF YES
7515  043170  122737  000060  001560          CMPB    #60,@#SUBPASS   ;FIRST SUB-PASS?
7516  043176  001024                          BNE     STMM            ;BRANCH IF NO
7517  043200  105737  001531                  TSTB    @#MMON          ;MEM MGMT ON?
7518  043204  001021                          BNE     STMM            ;BRANCH IF YES
7519  043206  052777  000047  137104  MBT1:   BIS     #47,@MBTTBL+12          ;CLEAR THE MBT
7520  043214  012777  000007  137076          MOV     #7,@MBTTBL+12           ;SELECT UNIT 7
7521  043222  005077  137062                  CLR     @MBTTBL+2               ;CLEAR THE WORD COUNT
7522  043226  012777  053766  137074          MOV     #MBTSRV,@MBTTBL+22      ;SETUP INTERRUPT VECTOR
7523  043234  012777  000240  137070          MOV     #PR5,@MBTTBL+24         ;SET VECTOR PSW
7524  043242  112777  000161  137036          MOVB    #161,@MBTTBL            ;START MBT
7525                                  ;;*****************************************************************
7526                                  .SBTTL  STMM ROUTINE
7527                                  ;ROUTINE TO SET UP MEMORY MANAGEMENT TO RELOCATE PROGRAM CODE ABOVE 16K
7528                                  ;CHECK IF PROGRAM IS TO BE RELOCATED.
7529                                  ;SW6=1=NO RELOCATION
7530                                  ;;*****************************************************************
7531  043250  112737  000100  001202  STMM:   MOVB    #100,@#$TSTNM   ;LOAD TEST NUMBER
7532  043256  112737  000100  177570          MOVB    #100,@#SWR      ;IN SWICH REG TOO
7533                                                                  ;ANY TIME TEST 100 IS IN ERROR REPORT
7534                                                                  ;THEN ERROR OCCURRED DURING RELOCATION
7535  043264  032737  000100  177570          BIT     #SW6,@#SWR      ;RELOCATION DISABLED?
7536  043272  001402                          BEQ     3$              ;BRANCH IF NO
7537  043274  000167  002622                  JMP     ENDM
7538
7539                                  ;THE PROGRAM IS GOING TO RELOCATE.
7540                                  ;RELOCATION WILL BE PERFORMED IN KERNEL MODE WITH PSW SET AT PRIORITY
7541                                  ;LEVEL 4 (TO PREVENT TTY INTERRUPT-WHICH CHANGES DATA IN PROGRAM)
7542                                  ;THE 'T' BIT IS CLEARED (IF SET). AFTER THE DATA HAS BEEN WRITTEN IT IS
7543                                  ;VERIFIED BEFORE EXECUTION.
7544  043300  013727  177776          3$:     MOV     @#PSW,(PC)+     ;SAVE CURRENT PSW
7545  043304  000000                  OLDPSW: .WORD   0
7546  043306  012737  000200  177776          MOV     #PR4,@#PSW
7547  043314  004767  017642                  JSR     PC,CLRTBIT      ;CO CLEAR 'T' BIT IF SET
7548
7549                                  ;NOW SETUP MEMORY MANAGEMENT REGISTERS.
7550  043320  012700  077406                  MOV     #77406,R0       ;SET CONSTANT=R/W UP 4K WORDS
7551  043324  010037  172300                  MOV     R0,@#KIPDR0     ;SET KIPDR0 THROUGH 7 R/W UP 4K WORDS
7552  043330  010037  172302                  MOV     R0,@#KIPDR1
7553  043334  010037  172304                  MOV     R0,@#KIPDR2
7554  043340  010037  172306                  MOV     R0,@#KIPDR3
7555  043344  010037  172310                  MOV     R0,@#KIPDR4
7556  043350  010037  172312                  MOV     R0,@#KIPDR5
7557  043354  010037  172314                  MOV     R0,@#KIPDR6
7558  043360  010037  172316                  MOV     R0,@#KIPDR7
7559
7560  043364  005037  172340                  CLR     @#KIPAR0        ;SET UP KIPDR0 THROUGH 3 FOR NO RELOCATION
7561  043370  012737  000200  172342          MOV     #200,@#KIPAR1
7562  043376  012737  000400  172344          MOV     #400,@#KIPAR2
7563  043404  012737  000600  172346          MOV     #600,@#KIPAR3
7564  043412  013737  001546  172350          MOV     @#NEXPAR,@#KIPAR4       ;PAR4 MAPS TO BEGINNING OF RELOCATION SPOT
7565  043420  013737  172350  172352          MOV     @#KIPAR4,@#KIPAR5
7566  043426  062737  000600  172352          ADD     #600,@#KIPAR5   ;PAR5 MAPS TO TOP 4K PAGE (NEXPAR START ADDRESS + 12K)
7567  043434  012737  177600  172356          MOV     #177600,@#KIPAR7        ;AND OF COUSE THE I/O PAGE
```

```
                                        ;NOW SETUP USER MEM MGMT REGISTERS
043442  010037  177600          1$:     MOV     R0,@#UIPDR0     ;SET UP USER MEM MGMT REGS
043446  010037  177602                  MOV     R0,@#UIPDR1
043452  010037  177604                  MOV     R0,@#UIPDR2
043456  010037  177606                  MOV     R0,@#UIPDR3
043462  010037  177616                  MOV     R0,@#UIPDR7
043466  016737  136054  177640          MOV     NEXPAR,@#UIPAR0
043474  013737  177640  177642          MOV     @#UIPAR0,@#UIPAR1
043502  062737  000200  177642          ADD     #200,@#UIPAR1
043510  013737  177640  177644          MOV     @#UIPAR0,@#UIPAR2
043516  062737  000400  177644          ADD     #400,@#UIPAR2
043524  013737  177640  177646          MOV     @#UIPAR0,@#UIPAR3
043532  062737  000600  177646          ADD     #600,@#UIPAR3
043540  013737  172356  177656          MOV     @#KIPAR7,@#UIPAR7

043546  010037  172200                  MOV     R0,@#SIPDR0     ;SET UP SUPERVISOR MEM MGMT REGS
043552  010037  172202                  MOV     R0,@#SIPDR1
043556  010037  172204                  MOV     R0,@#SIPDR2
043562  010037  172206                  MOV     R0,@#SIPDR3
043566  010037  172216                  MOV     R0,@#SIPDR7
043572  016737  135750  172240          MOV     NEXPAR,@#SIPAR0
043600  013737  172240  172242          MOV     @#SIPAR0,@#SIPAR1
043606  062737  000200  172242          ADD     #200,@#SIPAR1
043614  013737  172240  172244          MOV     @#SIPAR0,@#SIPAR2
043622  062737  000400  172244          ADD     #400,@#SIPAR2
043630  013737  172240  172246          MOV     @#SIPAR0,@#SIPAR3
043636  062737  000600  172246          ADD     #600,@#SIPAR3
043644  013737  172356  172256          MOV     @#KIPAR7,@#SIPAR7
043652  012737  000001  177572          MOV     #1,@#SR0        ;ENABLE MEM MGMT
043660  012737  000060  172516          MOV     #60,@#SR3       ;SETUP SR3
043666  110637  001531                  MOVB    SP,@#MMON       ;SET MEM MGMT ON IND = ON
043672  005037  000006          RETRY:  CLR     @#ERRVEC+2
043676  012737  046120  000004          MOV     #ENDMEM,@#ERRVEC;SET TIME OUT TRAP VECTOR
043704  013701  000116                  MOV     @#CACHVEC+2,R1  ;SAVE CACHVEC PSW
043710  005037  000116                  CLR     @#CACHVEC+2
043714  012737  046120  000114          MOV     #ENDMEM,@#CACHVEC       ;SET UP CACHE VECTOR FOR HOLE
043722  012702  100000                  MOV     #100000,R2      ;SETUP GENERAL REGISTERS
043726  012700  000000                  MOV     #0,R0           ;DATA WILL BE RELOCATED FROM
                                                                ;ADDRESS IN R0 TO ADDRESS IN R2 AFTER MAPPING
043732  012703  127776                  MOV     #127776,R3      ;SELECT PAR5 + 2K (14K TOTAL REQUIRED)
043736  010013                          MOV     R0,(R3)         ;TRAP TO ENDMEM IF INSUFFICIENT MEMORY
043740  012737  064270  000004          MOV     #ERPRT,@#ERRVEC ;RESTORE ERROR TRAP VECTOR
043746  010137  000116                  MOV     R1,@#CACHVEC+2
043752  012737  063464  000114          MOV     #.PARSRV,@#CACHVEC      ;RESTORE CACHE VECTOR
043760  000137  044160                  JMP     @#IOMON
                                ;;*****************************************************************
                                .SBTTL  RELOCATION ROUTINE
                                ;*      THIS ROUTINE IS USED TO RELOCATE THE 9 SUBTESTS UP TO 28K.
                                ;*      IF RELOCATION BY AN I/O DEVICE IS SELECTED, CONTROL IS PASSED
                                ;*      TO THE I/O MONITOR.
                                ;*      ENTER WITH:
                                ;*              FRSTAD=PHYSICAL ADDRESS OF FIRST CODE
                                ;*              FACTOR=NUMBER OF BYTES ABOVE BASE CODE
                                ;*              R2     =LAST PHYSICAL ADDRESS OF THE SECTION
                                ;*      EXIT TO I/O MONITOR WITH:
                                ;*              OLDBASE=FIRST PHYSICAL ADDRESS TO BE RELOCATED
```

```
                                ;*              NWBASL =FIRST PHYSICAL ADDRESS TO RELOCATE TO
                                ;*              IOWC   =TWO'S COMPLIMENT WORD COUNT
                                ;;*****************************************************************
043764  032737  000100  177570  RELOC:  BIT     #SW6,@#SWR              ;IS RELOCATION DISABLED?
043772  001067                          BNE     EXITRE                  ;BRANCH IF YES
043774  105737  001531                  TSTB    @#MMON                  ;IS MEMORY MGMT ON?
044000  001064                          BNE     EXITRE                  ;BRANCH IF YES
044002  013700  001540                  MOV     @#FRSTAD,R0             ;GET FIRST ADDRESS TO BE RELOCATED
044006  010005                          MOV     R0,R5
                                ;LAST ADDRESS IS IN R2
044010  010203                          MOV     R2,R3                   ;SAVE LAST ADDRESS
044012  010204                          MOV     R2,R4
044014  160004                          SUB     R0,R4                   ;R4 NOW HAS BYTE COUNT
044016  010437  001536                  MOV     R4,@#$FACTOR            ;SAVE BYTE COUNT
044022  005737  001534                  TST     @#FACTOR                ;FIRST RELOC IS TO ENDTAG+2
044026  001004                          BNE     1$                      ;BRANCH IF NOT EXECUTING BASE CODE
044030  010237  044156                  MOV     R2,@#RETPC              ;SAVE RETURN PC TO NEXT SECTION
044034  013702  001542                  MOV     @#FRSTMEM,R2            ;GET FIRST ADDRESS TO RELOCATE TO
044040  060204                  1$:     ADD     R2,R4                   ;R4 NOW CONTAINS LAST MEM ADDRESS
044042  020437  001544                  CMP     R4,@#LSTMEM             ;ENOUGH MEMORY?
044046  101042                          BHI     NOMEM                   ;BRANCH IF NO
044050  160204                          SUB     R2,R4                   ;R4 NOW HAS BYTE COUNT
044052  005037  001534                  CLR     @#FACTOR
044056  032737  000400  177570          BIT     #SW8,@#SWR              ;INHIBIT RELOC BY I/O DEVICE?
044064  001414                          BEQ     RELNIO                  ;BRANCH IF YES
044066  010037  001566                  MOV     R0,@#OLDBASE            ;SAVE START ADDRESS
044072  010237  001570                  MOV     R2,@#NWBASL             ;SAVE NEW BASE ADDRESS
044076  005037  001572                  CLR     @#NWBASH                ;
044102  006204                          ASR     R4                      ;MAKE IT A WORD COUNT
044104  005404                          NEG     R4                      ;GET TWO'S COMPLIMENT
044106  010437  001574                  MOV     R4,@#IOWC               ;SAVE R4 AS WORDCOUNT
044112  000167  000122                  JMP     ENTER2                  ;GO TO I/O MONITOR
                                ;RELOCATE BY CPU-MEMORY MANAGEMENT OFF
044116  012022                  RELNIO: MOV     (R0)+,(R2)+             ;RELOCATE CODE
044120  020003                          CMP     R0,R3                   ;DONE YET?
044122  001375                          BNE     RELNIO                  ;BRANCH IF NO
044124  004737  062534                  JSR     PC,@#CHKDAT             ;GO CHECK DATA
044130  102010                          BVC     EXITRE
044132  010037  001302                  MOV     R0,@#$TMP0              ;SAVE R0 FOR TYPEOUT
044136  010237  001522                  MOV     R2,@#VADR               ;SAVE R2
044142  004737  062432                  JSR     PC,@#CNVADR             ;CONVERT R2 TO A PHYSICAL ADR
044146  104006                          ERROR   6
044150  000401                          BR      NOMEM
044152  010207                  EXITRE: MOV     R2,PC                   ;GO EXECUTE RELOCATED CODE
044154  011707                  NOMEM:  MOV     (PC),PC                 ;GO TO NEXT SECTION
044156  000000                  RETPC:  .WORD   0                       ;CONTAINS PC OF NEXT SECTION
                                ;;*****************************************************************
                                .SBTTL  I/O RELOCATION MONITOR
                                ;*      THIS ROUTINE IS USED TO SCHEDULE I/O DEVICES FOR SUBTEST
                                ;*      RELOCATION AND PROGRAM RELOCATION. THE I/O DEVICE UNIT
                                ;*      NUMBER IS DETERMINED, THE BUS ADDRESS CALCULATED, THE WORD
                                ;*      COUNT CALCULATED AND PASSED TO THE DEVICE HANDLER.
                                ;;*****************************************************************
044160  012737  044166  001212  IOMON:  MOV     #1$,@#$LPERR    ;SETUP ERROR LOOP
044166  012737  000000  001566  1$:     MOV     #0,@#OLDBASE
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$:   MOV     #174000,@#IOWC          ;SET WORD COUNT TO 2K
7688  044240  005037  001306          ENTER2: CLR     @#$TMP2
7689  044244  012737  177776  001604        MOV     #-2,@#RNTBINX           ;SETUP RUN TABLE INDEX
7690  044252  005037  001302                CLR     @#$TMP0
7691  044256  005002                        CLR     R2                      ;CLEAR LEGAL DEV FLAG
7692  044260  032737  000040  177570  41$:   BIT     #SW5,@#SWR              ;INHIBIT ROUND ROBIN?
7693  044266  001416                        BEQ     50$                     ;BRANCH IF NO
7694  044270  005737  001302                TST     @#$TMP0                 ;FLAG SET?
7695  044274  001027                        BNE     43$                     ;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     @#$TMP0                 ;SET FLAG
7700  044322  000414                        BR      43$                     ;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$:   MOV     @#DEVINDX,R3            ;GET INDEX
7707  044360  012737  000401  001304        MOV     #401,@#$TMP1            ;INIT UNIT MASK
7708  044366  012704  000010                MOV     #10,R4          ;SET SOB COUNT
7709  044372  133763  001304  001642  44$:   BITB    @#$TMP1,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    @#$TMP1+1,SYSSIZE+1(R3) ;HAS IT BEEN USED?
7713  044412  001516                        BEQ     11$                     ;BRANCH IF NO
7714  044414  006337  001304          52$:  ASL     @#$TMP1                 ;SELECT NEXT UNIT
7715  044420  077414                        SOB     R4,44$                  ;CONTINUE
7716  044422  005737  001302                TST     @#$TMP0                 ;INHIBIT ROUND ROBIN?
7717  044426  001013                        BNE     45$                     ;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                MOV     #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
```

```
 7736  044514  104400  044522                 TYPE    ,65$                ;;TYPE ASCIZ STRING
 7737  044520  000407                         BR      64$                 ;;GET OVER THE ASCIZ
 7738                                  ;;65$:  .ASCIZ  /UNAVAILABLE/<CRLF>
 7739  044540                          64$:
 7740  044540  000637                         BR      ENTER2
 7741  044542  105737  001532          46$:    TSTB    @#QV                        ;ACT11?
 7742  044546  001016                         BNE     51$                         ;BRANCH IF YES
 7743  044550  005227  177777                 INC     #-1
 7744  044554  001013                         BNE     51$
 7745  044556  104400  044564                 TYPE    ,67$                ;;TYPE ASCIZ STRING
 7746  044562  000410                         BR      66$                 ;;GET OVER THE ASCIZ
 7747                                  ;;67$:  .ASCIZ  ?NO I/O DEVICES?<CRLF>
 7748  044604                          66$:
 7749  044604  105737  001531          51$:    TSTB    @#MMON                      ;MGMT ON?
 7750  044610  001012                         BNE     61$                         ;BRANCH IF YES
 7751  044612  013700  001566                 MOV     @#OLDBASE,R0                ;RESTORE R0
 7752  044616  013702  001570                 MOV     @#NWBASL,R2                 ;RESTORE R2
 7753  044622  013703  001536                 MOV     @#$FACTOR,R3                ;GET RELOCATION FACTOR
 7754  044626  060003                         ADD     R0,R3                       ;FORM LAST ADDRESS
 7755  044630  010005                         MOV     R0,R5                       ;SETUP R5
 7756  044632  000167  177260                 JMP     RELNIO                      ;GO RELOCATE WITH CP
 7757  044636  012702  100000          61$:    MOV     #100000,R2                  ;SETUP REGISTERS
 7758  044642  005000                         CLR     R0                          ;WITH FROM AND TO ADDRESS
 7759  044644  000137  045672                 JMP     @#RELOCP                    ;RELOCATE VIA CP
 7760  044650  105763  001736          11$:    TSTB    RP3HSTAT(R3)                ;IS HANDLER BUSY?
 7761  044654  100405                         BMI     8$                          ;BRANCH IF NO
 7762  044656  005737  001302                 TST     @#$TMP0                     ;ROUND ROBIN?
 7763  044662  001372                         BNE     11$                         ;BRANCH IF NO
 7764  044664  000167  177370                 JMP     41$
 7765  044670  005763  001736           8$:    TST     RP3HSTAT(R3)                ;DID HANDLER FAIL?
 7766  044674  100005                         BPL     62$                         ;BRANCH IF NO
 7767  044676  005737  001302                 TST     @#$TMP0                     ;ROUND ROBIN
 7768  044702  001402                         BEQ     62$                         ;BRANCH IF YES
 7769  044704  000137  045652                 JMP     @#15$
 7770  044710  153763  001305  001643  62$:    BISB    @#$TMP1+1,SYSSIZE+1(R3)     ;SET UNIT USED BIT
 7771  044716  005002                         CLR     R2
 7772  044720  006037  001304           30$:    ROR     @#$TMP1                     ;ENCODE THE BIT POSITION
 7773  044724  005202                         INC     R2                          ;INTO A UNIT NUMBER
 7774  044726  103374                         BCC     30$
 7775  044730  005302                         DEC     R2
 7776  044732  010237  001602                 MOV     R2,@#UNITNO                 ;SAVE UNIT NUMBER
 7777  044736  013763  001574  001756  10$:    MOV     @#IOWC,RP3HWC(R3)           ;GIVE WORD COUNT TO HANDLER
 7778  044744  010304                         MOV     R3,R4                       ;
 7779  044746  072427  000003                 ASH     #3,R4                       ;ENCODE DEVICE FOR RUNTABLE
 7780  044752  053704  001602                 BIS     @#UNITNO,R4                 ;ENCODE UNIT NUMBER
 7781  044756  006304                         ASL     R4
 7782  044760  062737  000002  001604         ADD     #2,@#RNTBINX                ;INCREMENT RUN TABLE INDEX
 7783  044766  013702  001604                 MOV     @#RNTBINX,R2                ;GET RUN TABLE INDEX
 7784  044772  110462  001663                 MOVB    R4,RUNTBL+1(R2)             ;ENTER DEV & UNIT IN TABLE
 7785  044776  013763  001602  002052         MOV     @#UNITNO,RP3UNIT(R3)        ;GIVE HANDLER UNIT NUMBER
 7786  045004  012737  000240  000012         MOV     #PR5,@#RESVEC+2             ;SETUP RESERVED VECTOR PSW
 7787  045012  016337  002066  000010         MOV     RP3HANA(R3),@#RESVEC        ;SETUP RESERVED VECTOR
 7788  045020  006303                         ASL     R3                          ;ADJUST INDEX
 7789  045022  013763  001566  001772         MOV     @#OLDBASE,RP3OLD(R3)        ;GIVE HANDLER OLD BASE ADDRESS
 7790  045030  013763  001570  002022         MOV     @#NWBASL,RP3NWL(R3)         ;GIVE HANDLER
 7791  045036  013763  001572  002024         MOV     @#NWBASH,RP3NWH(R3)         ;NEW BASE ADDRESS
```

```
7792  045044  005063  001774                CLR     RP3OLD+2(R3)            ;ENSURE OLD BASE HIGH IS CLR
7793  045050  000010                        CALLHANDLER
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        13$:    MOVB    @#LTICKS+1,R5           ;GET SECOND COUNT
7803  045120  062705  000002                ADD     #2,R5                   ;INCREMENT BY TWO
7804  045124  162705  000074                SUB     #60.,R5                 ;ENSURE RESULT IS 59 OR LESS
7805  045130  100002                        BPL     31$                     ;
7806  045132  062705  000074                ADD     #60.,R5                 ;COUNT WAS LESS THAN 58-RESTORE
7807  045136  012700  000010        31$:    MOV     #10,R0                  ;SET SOB COUNT
7808  045142  005002                        CLR     R2
7809  045144  005003                        CLR     R3
7810  045146  005004                        CLR     R4
7811  045150  066203  001736        14$:    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     @$LPERR                 ;RESTART RELOCATION
7823  045212  005704                32$:    TST     R4                      ;ANY DEVICE ERRORS?
7824  045214  001402                        BEQ     82$                     ;BRANCH IF NO
7825  045216  000167  000430                JMP     15$                     ;ERROR
7826  045222  105737  001531        82$:    TSTB    @#MMON                  ;MEM MGMT ON?
7827  045226  001012                        BNE     25$                     ;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        25$:    MOV     #70000,R0               ;GET LAST ADR + 2 OF GOOD DATA
7837  045260  012702  110000                MOV     #110000,R2              ;GET LAST ADR + 2 OF DATA TO BE CHECKED
7838  045264  013737  172352  172350        MOV     @#KIPAR5,@#KIPAR4       ;SET UP PAR4 FOR TOP 4K BANK
7839  045272  012705  002344                MOV     #$ERRTB,R5              ;DON'T CHECK BELOW $ERRTB
7840  045276  004737  062534        22$:    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  83$:  BIT     #SW9,@#SWR              ;LOOP ON ERROR?
7846  045324  001473                        BEQ     100$+2                  ;BRANCH IF NO
7847  045326  000167  000244                JMP     20$                     ;GO DO FUNCTION AGAIN
```

```
7848  045332  005001                  81$:    CLR     R1
7849  045334  010037  001302                  MOV     R0,@#$TMP0
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$:    MOVB    RUNTBL+1(R4),R4         ;GET DEVICE THAT FAILED
7861  045376  042704  177400                  BIC     #177400,R4              ;ENSURE HIGH BYTE CLEAR
7862  045402  006204                          ASR     R4                      ;THROW AWAY LSB
7863  045404  005005                          CLR     R5                      ;ENSURE R5 CLEAR
7864  045406  073427  177775                  ASHC    #-3,R4                  ;GET UNIT NUMBER IN R5
7865  045412  010500                          MOV     R5,R0
7866  045414  072027  177763                  ASH     #-15,R0
7867  045420  042700  177770                  BIC     #177770,R0
7868  045424  010037  001310                  MOV     R0,@#$TMP3
7869  045430  010403                          MOV     R4,R3                   ;AND DEVICE INDEX IN R4 & R3
7870  045432  010337  001306                  MOV     R3,@#$TMP2
7871  045436  012737  000001  001304          MOV     #1,@#$TMP1                      ;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     @#$TMP1                 ;SELECT NEXT UNIT
7875  045456  000772                          BR      19$                     ;CONTINUE
7876  045460  012737  045576  001212  18$:    MOV     #20$,@#$LPERR           ;SETUP LOOP RETURN
7877  045466  005701                          TST     R1                      ;DEVICE ERROR?
7878  045470  001010                          BNE     100$                    ;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 RP03
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$:     MOV     #RP3HSTAT,R5    ;DON'T CHECK BELOW RP3HSTAT
7932  045746  004737  062534            JSR     PC,@#CHKDAT                 ;CHECK DATA
7933  045752  102012                    BVC     EXIT
7934  045754  010037  001302            MOV     R0,@#$TMP0
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            MOVB    R0,@#$TSTNM+1   ;PUT IN DSPLAY REG HIGH BYTE
7954  046064  012706  001200            MOV     #KERSTK,SP      ;SET KERNEL STACK PTR
7955  046070  005037  177776            CLR     @#PSW
7956  046074  016746  175204            MOV     OLDPSW,-(SP)    ;RESTORE OLD PSW
7957  046100  012746  010000            MOV     #LOOP,-(SP)
7958  046104  105737  001530            TSTB    @#NEXEC         ;BRANCH IF TEST CODE TO
7959  046110  001402                    BEQ     1$              ;BE EXECUTED
```

```
 7960  046112  012716  043250            MOV     #STMM,(SP)
 7961  046116  000002            1$:     RTI                             ;RESTART PROGRAM AT LOOP
 7962
 7963                            ;WHEN RELOCATION ABOVE 28K IS COMPLETE PROGRAM TRAPS TO ENDMEM.
 7964  046120  022626            ENDMEM: CMP     (SP)+,(SP)+             ;POP STACK TWICE
 7965  046122  005037  177572    ENDM:   CLR     @#SR0                   ;DISABLE MEM MGMT
 7966  046126  042737  000020  172516    BIC     #BIT4,@#MMR3            ;CLEAR 22 BIT MODE
 7967
 7968                            ;*************************************************************************
 7969                            ;AT THIS TIME A 'SUB-PASS' HAS BEEN COMPLETED.
 7970                            ;PROGRAM NOW EXECUTING IN KERNEL MODE AT PC AS SHOWN (NO RELOCATION)
 7971  046134  012737  000700  001546    MOV     #700,@#NEXPAR           ;RESET NEXT VALUE FOR PAR REGISTERS
 7972  046142  005737  003574            TST     @#PROT
 7973  046146  001403                    BEQ     2$
 7974  046150  012737  001600  001546    MOV     #1600,@#NEXPAR
 7975  046156  105037  001531    2$:     CLRB    @#MMON                  ;SET MEM MGMT ON IND = OFF
 7976                            ;;*********************************************************************
 7977                            .SBTTL  END OF SUB-PASS ROUTINE
 7978                            ;*      THIS ROUTINE SETSUP THE PSW AND MAINTENANCE REGISTERS
 7979                            ;*      FOR THE NEXT SUB-PASS. IT THEN STARTS THE PRINTER
 7980                            ;*      (IF NOT ON ACT11) FOR TYPING THE END OF SUB-PASS MESSAGE.
 7981                            ;;*********************************************************************
 7982  046162                    END:
 7983  046162  012737  064270  000004    END1:   MOV     #ERPRT,@#ERRVEC
 7984  046170  012737  063464  000114            MOV     #.PARSRV,@#CACHVEC      ;RESTORE CACHE PARITY VECTOR
 7985  046176  010137  000116            MOV     R1,@#CACHVEC+2  ;AND PSW
 7986  046202  005037  177776            CLR     @#PSW           ;CLEAR MODE BITS IN PSW
 7987  046206  004767  014750            JSR     PC,CLRTBIT      ;GO CLEAR 'T' BIT IF SET
 7988  046212  012706  001200            MOV     #KERSTK,SP      ;SET KERNEL STACK PTR
 7989  046216  032777  000100  133020    BIT     #100,@$TPS      ;CHECK IF OUTPUT DEVICE IS BUSY
 7990  046224  001374                    BNE     .-6             ;IS AVAILABLE
 7991  046226  105237  001560    1$:     INCB    @#SUBPASS
 7992  046232  113702  001560            MOVB    @#SUBPASS,R2
 7993  046236  162702  000060            SUB     #60,R2
 7994  046242  022702  000006            CMP     #6,R2                   ;END OF TEST?
 7995  046246  001013                    BNE     2$              ;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     @#$MAINT                ;CLEAR SOFTWARE VALUE
 7999  046266  005046                    CLR     -(SP)
 8000  046270  012746  046376            MOV     #$EOP,-(SP)
 8001  046274  000002                    RTI
 8002  046276  006302            2$:     ASL     R2
 8003  046300  012737  001520  000014    MOV     #$RTRN,@#TBITVEC        ;SET 'T' TRAP VECTOR
 8004  046306  012737  001557  001270    MOV     #SUBPASS-1,@#$REG5
 8005  046314  106277  132724            ASRB    @$TPS
 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),@#$MAINT
 8009  046336  016237  064400  177750    MOV     MRGTAB(R2),@#MAINT
 8010  046344  105737  001532    3$:     TSTB    @#QV            ;QV PASS?
 8011  046350  001011                    BNE     RTI1            ;BRANCH IF YES
 8012  046352  122777  000200  132664    CMPB    #200,@$TPS      ;IS PRINTER READY?
 8013  046360  001371                    BNE     3$              ;BRANCH IF NO
 8014  046362  012737  065313  001270    MOV     #MSG20-1,@#$REG5
 8015  046370  106277  132650            ASRB    @$TPS           ;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 YYYYY''
8025                                  ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
8026                                  ;*IF THERES A MONITOR GO TO IT
8027                                  ;*IF THERE ISN'T JUMP TO LOOP
8028
8029  046376                          $EOP:
8030  046376  004737  056506                 JSR     PC,@#TYPTIME
8031  046402  005067  132574                 CLR     $TSTNM          ;;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   1
8040  046436  046426                         $EOPCT
8041  046440  104400  046446                 TYPE    ,65$            ;;TYPE ASCIZ STRING
8042  046444  000407                         BR      64$             ;;GET OVER THE ASCIZ
8043                                 ;;65$:  .ASCIZ  <12><15>/END PASS #/
8044  046464                         64$:
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 ASCIZ STRING
8049  046476  000421                         BR      66$             ;;GET OVER ASCIZ
8050                                 ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
8051  046542                         66$:
8052  046542  016746  132446                 MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
8053                                                                  ;;TOTAL NUMBER OF ERRORS
8054  046546  104410                         TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
8055  046550  104400  001337                 TYPE    ,$CRLF          ;;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$:     JSR     PC,@#$RAND              ;GO GET RANDOM NUMBER
8085  046664  013700  001552                    MOV     @#$HINUM,R0             ;GET HI NUMBER
8086  046670  013701  001550                    MOV     @#$LONUM,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$:     MOV     @#RP311,R2              ;GET RUN TABLE INDEX
8094  046724  016203  001662                    MOV     RUNTBL(R2),R3           ;GET DEVICE ID
8095  046730  042703  000777                    BIC     #777,R3                 ;ID ONLY
8096  046734  050300                            BIS     R3,R0                   ;COMBINE WITH CYL ADR
8097  046736  010062  001662                    MOV     R0,RUNTBL(R2)           ;PUT BACK IN TABLE
8098  046742  072127  177775                    ASH     #-3,R1                  ;GEN TRK-SECT ADR
8099  046746  010103                            MOV     R1,R3                   ;SAVE
8100  046750  042701  160377            6$:     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$:     BIS     R3,R1                   ;COMBINE TRK-SECT
8111  047012  010162  001700                    MOV     R1,RUNTRAK(R2)          ;PUT IN TABLE
8112  047016  010037  002104                    MOV     R0,@#RP3HDC             ;SAVE DESIRED CYL
8113  047022  010137  002152                    MOV     R1,@#RP3DA              ;SAVE DSK ADR
8114  047026  112737  177775  002132            MOVB    #-3,@#RP3TRY            ;INIT TRY COUNT
8115  047034  032737  000040  172516            BIT     #BIT5,@#MMR3            ;MAP ON?
8116  047042  001405                            BEQ     7$                      ;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
```

```
8128  047120  010037  002052                  MOV     R0,@#RP3UNIT
8129  047124  104422                          RESREG
8130  047126  005777  133006          RP3WTRY:TST     @RP3DS                  ;IS DRIVE READY?
8131  047132  100375                          BPL     RP3WTRY                 ;BRANCH IF NO
8132  047134  053777  002052  133002          BIS     @#RP3UNIT,@RP3CS        ;SET UNIT BITS
8133  047142  004737  047174                  JSR     PC,@#LDRP3              ;LOAD RP3 REGISTERS
8134  047146  012777  050512  133002          MOV     #RP3SRV,@RP3VEC         ;SET VECTOR
8135  047154  005077  133000                  CLR     @RP3PSW
8136  047160  005037  002120                  CLR     @#RP3FUN                ;SET FUNCTION TO WRITE
8137  047164  013777  001612  132752          MOV     @#RP310,@RP3CS          ;LOAD FUNCT AND GO
8138  047172  000002                          RTI                             ;RETURN
8139  047174  013777  002102  132750  LDRP3:  MOV     @#RP3HDA,@RP3DA         ;LOAD DSK ADR
8140  047202  013777  002104  132744          MOV     @#RP3HDC,@RP3DC         ;LOAD CYL ADR
8141  047210  013777  001756  132730          MOV     @#RP3HWC,@RP3WC         ;LOAD WORD COUNT
8142  047216  013777  001772  132724          MOV     @#RP3OLD,@RP3BA         ;LOAD BUS ADR
8143  047224  000207                          RTS     PC                      ;RETURN
8144
8145                                  ;;*****************************************************************
8146                                  .SBTTL  RK11/RK05 HANDLER
8147                                  ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
8148                                  ;;*****************************************************************
8149  047226  104420                  RKDRV:  SAVREG
8150  047230  105037  001740                  CLRB    @#RKHSTAT               ;CLEAR DONE FLAG IN HANDLER STAT
8151  047234  032737  000400  001740          BIT     #BIT8,@#RKHSTAT         ;REPEAT FLAG SET?
8152  047242  001403                          BEQ     5$                      ;BRANCH IF NO
8153  047244  104422                          RESREG
8154  047246  000137  051300                  JMP     @#RKRPT
8155  047252  013737  001604  001620  5$:     MOV     @#RNTBINX,@#RK11        ;SAVE RUN TABLE INDEX
8156  047260  105037  001740                  CLRB    @#RKHSTAT               ;CLEAR DONE FLAG IN HANDLER STAT
8157  047264  032737  000020  177570          BIT     #SW4,@#SWR              ;RANDOM DSK ADDRESS?
8158  047272  001403                          BEQ     6$                      ;BRANCH IF YES
8159  047274  005000                          CLR     R0                      ;CLEAR REGISTERS
8160  047276  005001                          CLR     R1
8161  047300  000404                          BR      7$                      ;FOR ADDRESS CHECKING
8162  047302  004737  060436          6$:     JSR     PC,@#$RAND              ;GET RANDOM NUMBER
8163  047306  013700  001552                  MOV     @#$HINUM,R0             ;GET HIGH NUMBER
8164  047312  072027  177775          7$:     ASH     #-3,R0                  ;ADJUST TO FORM
8165                                                                          ;CYLINDER ADDRESS
8166  047316  010001                          MOV     R0,R1                   ;SAVE IN R1
8167  047320  042701  160037          4$:     BIC     #160037,R1              ;GET RID OF SURF-SECT BITS
8168  047324  022701  014300                  CMP     #14300,R1               ;IS IT A LEGAL CYLINDER?
8169  047330  100003                          BPL     3$                      ;BRANCH IF YES
8170  047332  062701  014340                  ADD     #14340,R1               ;ADD MAXIMUM CYLINDER
8171  047336  000770                          BR      4$                      ;TRY AGAIN
8172  047340  072127  177773          3$:     ASH     #-5,R1                  ;ADJUST CYLINDER ADDRESS
8173  047344  013702  001620                  MOV     @#RK11,R2               ;GET RUN TABLE INDEX
8174  047350  016203  001662                  MOV     RUNTBL(R2),R3           ;GET RUN TABLE ENTRY
8175  047354  042703  000777                  BIC     #777,R3                 ;SAVE ID AND UNIT NO.
8176  047360  050103                          BIS     R1,R3                   ;INSERT CYLINDER ADDR
8177  047362  010362  001662                  MOV     R3,RUNTBL(R2)           ;ENTER CYLINDER ADR IN RUN TABLE
8178  047366  072027  177770                  ASH     #-10,R0                 ;GENER SECTOR-SURF ADDRESS
8179  047372  042700  177740                  BIC     #177740,R0              ;GET RID OF EXTRA BITS
8180  047376  010003                          MOV     R0,R3                   ;SAVE
8181  047400  042700  000020                  BIC     #BIT4,R0                ;GET RID OF SURFACE BIT
8182  047404  022700  000012                  CMP     #12,R0                  ;IS SECTOR ADDRESS LEGAL?
8183  047410  100004                          BPL     1$                      ;BRANCH IF YES
```

```
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                                  .SBTTL  RH70/RP04 HANDLER
8220                                  ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
8221                                  ;;*******************************************************************
8222  047622  104420                  RP4DRV: SAVREG
8223  047624  105037  001746              CLRB    @#RP4HSTA               ;CLEAR DONE FLAG
8224  047630  032737  000400  001746      BIT     #BIT8,@#RP4HST          ;REPEAT FLAG SET?
8225  047636  001403                      BEQ     6$                      ;BRANCH IF NO
8226  047640  104422                      RESREG
8227  047642  000137  052156              JMP     @#RP4RPT
8228  047646  013737  001604  001622  6$: MOV     @#RNTBINX,@#RP411       ;SAVE RUN TABLE INDEX
8229  047654  105037  001746              CLRB    @#RP4HSTA               ;CLEAR DONE FLAG
8230  047660  032737  000020  177570      BIT     #SW4,@#SWR              ;RANDOM DSK ADDRESS?
8231  047666  001403                      BEQ     1$                      ;BRANCH IF YES
8232  047670  005000                      CLR     R0
8233  047672  005001                      CLR     R1
8234  047674  000410                      BR      4$
8235  047676  004737  060436      1$:     JSR     PC,@#$RAND              ;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. ADR.
8239  047716  042700  177000      4$:     BIC     #177000,R0              ;GET RID OF UNUSED BITS
```

```
 8240  047722  022700  000631          CMP     #631,R0                 ;LEGAL CYLINDER
 8241  047726  100003                  BPL     5$                      ;BRANCH IF YES
 8242  047730  062700  000631          ADD     #631,R0                 ;MAKE IT LEGAL
 8243  047734  000770                  BR      4$
 8244
 8245  047736  013702  001622  5$:     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         MOVB    #-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                ;SET FUNCTION TO WRITE
 8273  050106  112777  000161  132066         MOVB    #161,@RP4CS1            ;LOAD FUNCTION AND GO
 8274  050114  000002                         RTI                             ;RETURN
 3275
 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                 CLRB    @#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$:     MOV     @#RNTBINX,@#RS11        ;SAVE RUN TABLE INDEX
 8296  050224  032737  000020  177570         BIT     #SW4,@#SWR              ;RANDOM DSK ADR?
 8297  050232  001403                         BEQ     1$                      ;BRANCH IF YES
 8298  050234  005000                         CLR     R0
 8299  050236  005001                         CLR     R1
 8300  050240  000407                         BR      4$
 8301  050242  004737  060436         1$:     JSR     PC,@#$RAND              ;GET RANDOM NUMBER
```

```
8302  050246  013700  001552          MOV     @#$HINUM,R0
8303  050252  072027  177774          ASH     #-4,R0
8304  050256  010001                  MOV     R0,R1                   ;SAVE RANDOM NUMBER
8305  050260  042700  170077      4$: 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$: MOV     @#RS11,R2               ;GET RUN TABLE INDEX
8311  050304  072027  177772          ASH     #-6,R0                  ;ADJUST TRACK ADR
8312  050310  110062  001662          MOVB    R0,RUNTBL(R2)           ;SAVE TRAK ADR IN RUN TBL
8313  050314  042701  177700      6$: 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      2$: MOV     R1,RUNTRAK(R2)          ;SAVE IN RUN TRAK TABLE
8319  050340  072027  000006          ASH     #6,R0                   ;ADJUST TRACK ADDR
8320  050344  050100                  BIS     R1,R0                   ;COMBINE SECTOR TRAK
8321  050346  010037  002116          MOV     R0,@#RSHDA              ;SAVE AS DSK ADR
8322  050352  112737  177775  002136  MOVB    #-3,@#RSTRY             ;SET TRY COUNT
8323  050360  104422                  RESREG
8324  050362  004737  050422  RSWTRY: JSR     PC,@#LDRS               ;GO LOAD REGISTERS
8325  050366  012777  052712  131670  MOV     #RSSRV,@RSVEC           ;SET INTERRUPT VECTOR
8326  050374  005077  131666          CLR     @RSPSW
8327  050400  005037  002130          CLR     @#RSFUN                 ;SET FUNCTION TO WRITE
8328  050404  105777  131650      1$: TSTB    @RSDS                   ;IS DRIVE READY?
8329  050410  001775                  BEQ     1$                      ;BRANCH IF NO
8330  050412  112777  000161  131622  MOVB    #161,@RSCS1             ;LOAD FUNCTION AND GO
8331  050420  000002                  RTI
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     PC                      ;RETURN
8339
8340                                  ;;*****************************************************************
8341                                  .SBTTL  RP11/RP03 SERVICE ROUTINE
8342                                  ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
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     R0,-(SP)                ;SAVE R0
8373  050622  013700  001614                  MOV     @#RP311,R0              ;GET RUNTABLE INDEX
8374  050626  052760  100000  001700          BIS     #BIT15,RUNTRAK(R0)      ;SET ERROR BIT
8375  050634  012600                          MOV     (SP)+,R0                ;RESTORE R0
8376  050636  000002                          RTI                             ;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$:     RTI                             ;RETURN
8383                                  ;WRITE WAS OK- NOW DO A WRITE CHECK
8384  050664  112737  177775  002132  RP31:   MOVB    #-3,@#RP3TRY            ;INIT TRY COUNT
8385  050672  012737  000107  001612          MOV     #107,@#RP310            ;SET FUNCTION
8386  050700  053737  001774  001612          BIS     @#RP3OLD+2,@#RP310      ;SET BAE BITS
8387  050706  053737  002052  001612          BIS     @#RP3UNIT,@#RP310       ;SET UNIT BITS
8388  050714  004737  047174          RP32:   JSR     PC,@#LDRP3              ;LOAD RP3 REGISTERS
8389  050720  013777  001612  131216          MOV     @#RP310,@RP3CS          ;LOAD FUNCTION AND GO
8390  050726  000002                          RTI                             ;RETURN
8391
8392                                  ;FUNCTION JUST EXECUTED WAS A WRITE CHECK
8393  050730  032737  000400  001736  RP3WCK: BIT     #BIT8,@#RP3HSTAT                ;REPEAT FLAG SET?
8394  050736  001340                          BNE     RP3LOOP                 ;BRANCH IF YES
8395  050740  005777  131200                  TST     @RP3CS                  ;ANY ERRORS?
8396  050744  100031                          BPL     1$                      ;BRANCH IF NO
8397  050746  005737  001614                  TST     @#RP311                 ;FIRST 2K?
8398  050752  001422                          BEQ     4$                      ;BRANCH IF YES
8399  050754  105737  002132          5$:     TSTB    @#RP3TRY                ;TRIED 3 TIMES?
8400  050760  001714                          BEQ     RP3ERR                  ;BRANCH IF YES
8401  050762  005337  002120                  DEC     @#RP3FUN                ;RESTORE FUNCTION
8402  050766  112777  000001  131150          MOVB    #BIT0,@RP3CS            ;CLEAR THE DRIVE
8403  050774  105777  131144                  TSTB    @RP3CS                  ;CONTROLLER READY?
8404  051000  100375                          BPL     .-4                     ;BRANCH IF NO
8405  051002  105237  002132                  INCB    @#RP3TRY                ;INCREMENT TRY COUNT
8406  051006  013746  177776                  MOV     @#PSW,-(SP)
8407  051012  012746  050714                  MOV     #RP32,-(SP)
8408  051016  000002                          RTI                             ;GO TRY AGAIN
8409  051020  032777  000010  131114  4$:     BIT     #BIT3,@RP3ER            ;WRITE CHECK ERROR?
8410  051026  001752                          BEQ     5$                      ;BRANCH IF NO
8411
8412                                  ;WRITE CHECK OK- NOW DO A READ
8413  051030  112737  177775  002132  1$:     MOVB    #-3,@#RP3TRY            ;RESTORE TRY COUNT
```

```
8414  051036  032737  000040  172516          BIT     #BIT5,@#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$:     MOV     R0,-(SP)                ;SAVE R0
8421  051066  013700  002024                  MOV     @#RP3NWH,R0             ;GET BAE BITS
8422  051072  072027  000004                  ASH     #4,R0                   ;ADJUST
8423  051076  010037  002024                  MOV     R0,@#RP3NWH             ;SAVE
8424  051102  012600                          MOV     (SP)+,R0                ;RESTORE R0
8425  051104  012737  000105  001612          MOV     #105,@#RP310            ;SET FUNCTION
8426  051112  053737  002024  001612          BIS     @#RP3NWH,@#RP310        ;SET BAE BITS
8427  051120  053737  002052  001612          BIS     @#RP3UNIT,@#RP310       ;SET UNIT NUMBER
8428  051126  013777  002102  131016  RP33:   MOV     @#RP3HDA,@RP3DA         ;LOAD DSK ADR
8429  051134  013777  002104  131012          MOV     @#RP3HDC,@RP3DC         ;LOAD CYL
8430  051142  013777  001756  130776          MOV     @#RP3HWC,@RP3WC         ;LOAD WORD COUNT
8431  051150  013777  002022  130772          MOV     @#RP3NWL,@RP3BA         ;LOAD BUS ADR
8432  051156  013777  001612  130760          MOV     @#RP310,@RP3CS          ;LOAD FUNCTION AND GO
8433  051164  000002                          RTI                             ;RETURN
8434
8435                          ;FUNCTION JUST EXECUTED WAS A READ
8436  051166  032737  000400  001736  RP3READ:BIT     #BIT8,@#RP3HSTAT                ;REPEAT FLAG SET?
8437  051174  001221                          BNE     RP3LOOP                 ;BRANCH IF YES
8438  051176  005777  130742                  TST     @RP3CS                  ;ANY ERRORS?
8439  051202  100022                          BPL     1$                      ;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 OF NO
8446  051232  105237  002132                  INCB    @#RP3TRY                ;INCREMENT TRY COUNT
8447  051236  013746  177776                  MOV     @#PSW,-(SP)
8448  051242  012746  051126                  MOV     #RP33,-(SP)
8449  051246  000002                          RTI                             ;GO TRY AGAIN
8450  051250  032737  000040  172516  1$:     BIT     #BIT5,@#MMR3            ;MAP ON?
8451  051256  001404                          BEQ     2$                      ;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$:     MOVB    #200,@#RP3HSTA          ;SET DONE FLAG
8456  051276  000002                          RTI                             ;RETURN
8457                          ;;*************************************************************************
8458                          .SBTTL  RK11/RK05 SERVICE ROUTINE
8459                          ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8460                          ;;*************************************************************************
8461  051300  000005                  RKRPT:  RESET
8462  051302  005337  002122                  DEC     @#RKFUN                 ;RESTORE FUNCTION
8463  051306  022737  000001  002122          CMP     #1,@#RKFUN              ;WHAT IS IT?
8464  051314  001475                          BEQ     RK1                     ;BRANCH IF WC
8465  051316  002402                          BLT     1$                      ;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                          BNE     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 ERROR
 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  000002                  1$:     RTI                             ;RETURN
 8502                                  ;WRITE WAS OK, NOW DO A WRITE CHECK
 8503  051510  112737  177775  002133  RK1:    MOVB    #-3,@#RKTRY             ;RESTORE TRY COUNT
 8504  051516  012767  000507  130072          MOV     #507,RK10               ;SET FUNCTION TO WRITE
 8505  051524  053767  002000  130064          BIS     @#RKOLD+2,RK10          ;SET BA EXT BITS
 8506  051532  013777  002106  130434  RK2:    MOV     @#RKHDA,@RKDA           ;LOAD DISK ADDRESS
 8507  051540  013777  001760  130422          MOV     @#RKHWC,@RKWC           ;LOAD WORD COUNT
 8508  051546  013777  001776  130416          MOV     @#RKOLD,@RKBA           ;LOAD BUS ADDRESS
 5509  051554  016777  130036  130404          MOV     RK10,@RKCS              ;START FUNCTION
 8510  051562  000002                          RTI                             ;RETURN
 8511
 8512                                  ;FUNCTION JUST EXECUTED WAS A WRITE CHECK. ANY ERRORS?
 8513  051564  032737  000400  001740  RKWRCK:BIT     #BIT8,@#RKHSTAT                 ;REPEAT FLAG SET?
 8514  051572  001334                          BNE     RKLOOP                  ;BRANCH IF YES
 8515  051574  005777  130366                  TST     @RKCS                   ;ANY ERRORS?
 8516  051600  100033                          BPL     1$                      ;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                  DEC     @#RP4FUN                        ;SET FUNCTION BACK TO WC
 8522  051622  012777  000001  130336          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                 ;TRYED 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 REGSTER
8575  052126  005726                  TST     (SP)+                   ;POP THE STACK
8576  052130  112737  000200  001740  2$:     MOVB    #200,@#RKHSTAT          ;SET DON E FLAG
8577  052136  000002                  RTI                             ;RETURN
8578  052140  005067  000010          TIMER:  CLR     1$
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                                    .SBTTL  RH70/RP04 SERVICE ROUTINE
8586                                    ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8587                                    ;;*****************************************************************
8588  052156  000005                    RP4RPT: RESET
8589  052160  005337  002126                    DEC     @#RP4FUN                ;RESTORE FUNCTION
8590  052164  022737  000001  002126            CMP     #1,@#RP4FUN             ;WHAT IS IT?
8591  052172  001501                            BEQ     RP41                    ;BRANCH IF WC
8592  052174  002560                            BLT     RP43                    ;BRANCH IF READ
8593  052176  000137  050064                    JMP     @#RP4WTRY               ;GO TO WRITE
8594  052202  005237  002126            RP4SRV: INC     @#RP4FUN                ;FIND OUT WHAT FUNCTION
8595  052206  022737  000002  002126            CMP     #2,@#RP4FUN             ;WAS JUST EXECUTED
8596  052214  001504                            BEQ     RP4WCK
8597  052216  100566                            BMI     RP4READ
8598
8599                                    ;WRITE FUNCTION WAS JUST EXECUTED.
8600  052220  032737  000400  001746            BIT     #BIT8,@#RP4HSTAT        ;REPEAT FLAG SET?
8601  052226  001050                            BNE     RP4LOOP                 ;BRANCH IF YES
8602  052230  032777  040000  127762            BIT     #BIT14,@RP4DS           ;ANY ERRORS
8603  052236  001457                            BEQ     RP41                    ;BRANCH IF NO
8604  052240  105737  002135                    TSTB    @#RP4TRY                ;TRIED 3 TIMES?
8605  052244  001426                            BEQ     RP4ERR                  ;BRANCH IF YES
8606  052246  052777  000040  127740            BIS     #BIT5,@RP4CS2           ;CLEAR ALL ERRORS
8607  052254  004737  050116                    JSR     PC,@#LDRP4              ;RELOAD THE UNIT NO
8608  052260  105237  002135                    INCB    @#RP4TRY                ;INCREMENT TRY COUNT
8609  052264  013746  177776                    MOV     @#PSW,-(SP)             ;SETUP THE STACK TO
8610  052270  012746  050064                    MOV     #RP4WTRY,-(SP)          ;TRY WRITE AGAIN
8611  052274  032737  000400  001746            BIT     #BIT8,@#RP4HSTAT        ;REPEAT FLAG SET?
8612  052302  001006                            BNE     2$                      ;BRANCH IF YES
8613  052304  012777  000007  127670            MOV     #7,@RP4CS1              ;RECALIBRATE
8614  052312  105777  127702            1$:     TSTB    @RP4DS                  ;DRIVE READY?
8615  052316  100375                            BPL     1$                      ;BRANCH IF NO
8616  052320  000002                    2$:     RTI
8617  052322  012737  100200  001746    RP4ERR: MOV     #100200,@#RP4HSTA       ;SET ERROR & DONE BIT
8618  052330  010046                            MOV     R0,-(SP)                ;SAVE R0
8619  052332  013700  001622                    MOV     @#RP411,R0              ;GET RUN TABLE INDEX
8620  052336  052760  100000  001700            BIS     #BIT15,RUNTRAK(R0)      ;SET ERROR BIT
8621  052344  012600                            MOV     (SP)+,R0                ;RESTORE R0
8622  052346  000002                            RTI                             ;RETURN
8623
8624  052350  012737  100200  001746    RP4LOOP:MOV     #100200,@#RP4HSTAT      ;SET DONE AND ERROR BITS
8625  052356  032777  040000  127634            BIT     #BIT14,@RP4DS           ;ANY ERRORS?
8626  052364  001003                            BNE     1$                              ;BRANCH IF YES
8627  052366  042737  100000  001746            BIC     #BIT15,@#RP4HSTAT       ;CLEAR ERROR BIT
8628  052374  000002                    1$:     RTI                             ;RETURN
8629                                    ;WRITE OK...NOW DO A WRITE CHECK.
8630  052376  112737  177775  002135    RP41:   MOVB    #-3,@#RP4TRY            ;INITIALIZE TRY COUNT
8631  052404  105777  127610            RP42:   TSTB    @RP4DS                  ;IS DRIVE READY?
8632  052410  001775                            BEQ     RP42                    ;BRANCH IF NO
8633  052412  004737  050116                    JSR     PC,@#LDRP4
8634  052416  112777  000151  127556            MOVB    #151,@RP4CS1            ;LOAD FUNCTION AND GO
8635  052424  000002                            RTI
8636
8637                                    ;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$:     MOVB    #-3,@#RP4TRY            ;INITIALIZE TRY COUNT
 8659  052536  105777  127456          RP43:   TSTB    @RP4DS                  ;IS DRIVE READY?
 8660  052542  001775                          BEQ     RP43                    ;BRANCH IF NO
 8661  052544  004737  050116                  JSR     PC,@#LDRP4              ;LOAD REGISTERS
 8662  052550  013777  002044  127432          MOV     @#RP4NWH,@RP4BAE        ;LOAD EXTENDED ADR BITS
 8663  052556  013777  002042  127422          MOV     @#RP4NWL,@RP4BA         ;LOAD BUS ADR
 8664  052564  112777  000171  127410          MOVB    #171,@RP4CS1            ;LOAD FUNCTION AND GO
 8665  052572  000002                          RTI                             ;RETURN
 8666
 8667                                  ;FUNCTION JUST EXECUTED WAS A READ.
 8668  052574  032737  000400  001746  RP4READ:BIT     #BIT8,@#RP4HSTAT        ;REPEAT FLAG SET?
 8669  052602  001262                          BNE     RP4LOOP                 ;BRANCH IF YES
 8670  052604  032777  040000  127406          BIT     #BIT14,@RP4DS           ;ANY ERRORS?
 8671  052612  001421                          BEQ     1$                      ;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                                  .SBTTL  RH70/RS04 SERVICE ROUTINE
 8685                                  ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
 8686                                  ;;*****************************************************************
 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,@#RSFUN              ;WAS JUST EXECUTED
8695  052724  001472                          BEQ     RSWCK
8696  052726  100554                          BMI     RSREAD
8697
8698                                  ;WRITE FUNCTION WAS JUST EXECUTED
8699  052730  032737  000400  001750          BIT     #BIT8,@#RSHSTAT ;REPEAT FLAG SET?
8700  052736  001036                          BNE     RSLOOP                  ;BRANCH IF YES
8701  052740  032777  040000  127312          BIT     #BIT14,@RSDS            ;ANY ERRORS?
8702  052746  001445                          BEQ     RS41                    ;BRANCH IF NO
8703  052750  105737  002136                  TSTB    @#RSTRY                 ;TRIED 3 TIMES?
8704  052754  001414                          BEQ     RSERR                   ;BRANCH IF YES
8705  052756  052777  000040  127270          BIS     #BIT5,@RSCS2            ;CLEAR ALL ERRORS
8706  052764  004737  050422                  JSR     PC,@#LDRS               ;LOAD UNIT #
8707  052770  105237  002136                  INCB    @#RSTRY                 ;INCREMENT TRY COUNT
8708  052774  013746  177776                  MOV     @#PSW,-(SP)             ;SETUP THE STACK TO
8709  053000  012746  050362                  MOV     #RSWTRY,-(SP)           ;TRY THE WRITE AGAIN
8710  053004  000002                          RTI
8711  053006  012737  100200  001750  RSERR:  MOV     #100200,@#RSHSTAT       ;SET ERROR AND DONE BIT
8712  053014  010046                          MOV     R0,-(SP)                ;SAVE R0
8713  053016  013700  001624                  MOV     @#RS11,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,@#RSHSTAT       ;SET DONE AND ERROR BITS
8719  053042  032777  040000  127210          BIT     #BIT14,@RSDS            ;ANY ERRORS?
8720  053050  001003                          BNE     1$                      ;BRANCH IF YES
8721  053052  042737  100000  001750          BIC     #BIT15,@#RSHSTAT        ;CLEAR ERROR BIT
8722  053060  000002                  1$:     RTI                             ;RETURN
8723                                  ;WRITE OK...NOW DO A WRITE CHECK
8724  053062  112737  177775  002136  RS41:   MOVB    #-3,@#RSTRY             ;INIT TRY COUNT
8725  053070  105777  127164          RS42:   TSTB    @RSDS                   ;IS DRIVE READY?
8726  053074  001775                          BEQ     RS42                    ;BRANCH IF NO
8727  053076  004737  050422                  JSR     PC,@#LDRS               ;LOAD RS REGISTERS
8728  053102  112777  000151  127132          MOVB    #151,@RSCS1             ;LOAD FUNCTION AND GO
8729  053110  000002                          RTI                             ;RETURN
8730
8731                                  ;FUNCTION JUST EXECUTED WAS A WRITE CHECK
8732  053112  032737  000400  001750  RSWCK:  BIT     #BIT8,@#RSHSTAT         ;REPEAT FLAG SET?
8733  053120  001345                          BNE     RSLOOP                  ;BRANCH IF YES
8734  053122  032777  040000  127130          BIT     #BIT14,@RSDS            ;ANY ERRORS?
8735  053130  001421                          BEQ     1$                      ;BRANCH IF NO
8736  053132  105737  002136          3$:     TSTB    @#RSTRY                 ;TRIED 3 TIMES?
8737  053136  001723                          BEQ     RSERR                   ;BRANCH IF YES
8738  053140  005337  002130                  DEC     @#RSFUN                 ;SET FUNCTION BACK TO WC
8739  053144  052777  000040  127102          BIS     #BIT5,@RSCS2            ;CLEAR THE ERROR
8740  053152  004737  050422                  JSR     PC,     @#LDRS
8741  053156  105237  002136                  INCB    @#RSTRY                 ;INCREMENT THE TRY COUNT
8742  053162  013746  177776                  MOV     @#PSW,-(SP)
8743  053166  012746  053070                  MOV     #RS42,-(SP)
8744  053172  000002                          RTI                             ;TRY AGAIN
8745
8746  053174  032777  040000  127052  1$:     BIT     #BIT14,@RSCS2           ;WRITE CHECK ERROR?
8747  053202  001404                          BEQ     2$                      ;BRANCH IF NO
8748  053204  005737  001624                  TST     @#RS11                  ;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     #BIT5,@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$:     MOVB    #200,@#RSHSTAT          ;SET DONE FLAG
8777  053350  000002                          RTI                             ;RETURN
8778                                          ;;*****************************************************************
8779                                          .SBTTL  UNIBUS EXERCISER SERVICE ROUTINE
8780                                          ;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8781                                          ;;*****************************************************************
8782  053352  104420                  UBESRV: SAVREG
8783  053354  004737  063752                  JSR     PC,@#LDKT               ;GO TO LOW CORE
8784  053360  012704  002276                  MOV     #UBETBL+6,R4            ;GET ADDRESS OF UBECR1
8785  053364  005774  000000                  TST     @(R4)                   ;WAS THERE AN ERROR?
8786  053370  100437                          BMI     UBE2                    ;BRANCH IF YES
8787  053372  012746  000003                  MOV     #3,-(SP)                ;PUT DEVICE ID IN STACK
8788  053376  004737  063142                  JSR     PC,@#GIVEMAP            ;GIVE UP MAP REG
8789  053402  012767  002344  126312          MOV     #$ERRTB,UBESAV          ;INITIALIZE UBE
8790  053410  005067  126310                  CLR     UBESAV+2
8791  053414  012767  002344  126304          MOV     #$ERRTB,UBEADR
8792  053422  005067  126302                  CLR     UBEADR+2
8793  053426  012746  001726                  MOV     #UBEADR,-(SP)
8794  053432  004737  062676                  JSR     PC,@#GETMAP
8795  053436  013754  001730                  MOV     @#UBEADR+2,@-(R4)       ;LOAD UBECR2
8796  053442  013754  001726                  MOV     @#UBEADR,@-(R4)         ;LOAD UBEBA
8797  053446  012754  172400                  MOV     #172400,@-(R4)          ;LOAD UBECC
8798  053452  004737  064050                  JSR     PC,@#RESKT              ;GO BACK TO ORIGINAL CORE
8799  053456  104422                          RESREG
8800  053460  012777  064545  126610          MOV     #64545,@UBETBL+6        ;RESTART UBE
8801  053466  000002                          RTI                             ;RETURN
8802
8803                                          ;UBE ERROR-IS IT LAST MEMORY?
8804  053470  005037  001322          UBE2:   CLR     @#$TMP10
8805  053474  162704  000004                  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     @(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  000003          1$:     MOV     #3,-(SP)                ;PUT DEVICE ID ON STACK
8823  053606  004737  063142                  JSR     PC,@#GIVEMAP
8824  053612  005726                          TST     (SP)+
8825  053614  004737  062214                  JSR     PC,@#UBEINIT
8826  053620  004737  064050                  JSR     PC,@#RESKT
8827  053624  104422                          RESREG
8828  053626  012777  064545  126442          MOV     #64545,@UBETBL+6
8829  053634  000002                          RTI
8830
8831  053636  010637  001322          MHOLE:  MOV     SP,@#$TMP10
8832  053642  013737  001212  001324  UBEERR: MOV     @#$LPERR,@#$TMP11       ;SAVE LOOP ARROR ADR
8833  053650  012737  053712  001212          MOV     #UBE3,@#$LPERR          ;SET LOOP ADR
8834  053656  012703  000022                  MOV     #22,R3
8835  053662  005737  001322                  TST     @#$TMP10
8836  053666  001002                          BNE     1$
8837  053670  104007                          ERROR   7
8838  053672  000407                          BR      UBE3
8839  053674  013737  001524  001226  1$:     MOV     @#PA1500,@#$GDDAT
8840  053702  013737  001526  001230          MOV     @#PA2116,@#$BDDAT
8841  053710  104012                          ERROR   12
8842
8843                                  ;RESTART UBE IN SAME MEMORY
8844  053712  013737  001324  001212  UBE3:   MOV     @#$TMP11,@#$LPERR       ;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,@#LDKT               ;GO TO LOW CORE
 8863  053774  005037  001322          CLR     @#$TMP10
 8864  054000  012704  002306          MOV     #MBTTBL,R4              ;GET ADDRESS OF ADDRESS OF CS1 REG
 8865  054004  032734  040000          BIT     #BIT14,@(R4)+           ;ANY ERRORS?
 8866  054010  001007                  BNE     1$                      ;BRANCH IF YES
 8867  054012  004737  064050  2$:     JSR     PC,@#RESKT              ;GO BACK TO ORIGINAL CORE
 8868  054016  104422                  RESREG
 8869  054020  112777  000161  126260  MOVB    #161,@MBTTBL
 8870  054026  000002                  RTI                             ;RESTART MBT AND RETURN
 8871  054030  062704  000010  1$:     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)+,@#PA1500         ;GET BUS ADR
 8876  054054  013437  001526          MOV     @(R4)+,@#PA2116         ;GET BUS ADR EXT
 8877  054060  162737  000004  001524  SUB     #4,@#PA1500             ;ADJUST BUS ADR
 8878  054066  005637  001526          SBC     @#PA2116
 8879  054072  023737  001524  001610  CMP     @#PA1500,@#MXMMLO       ;IS IT LAST MEMORY?
 8880  054100  001015                  BNE     MEMHOLE                 ;BRANCH IF NO
 8881  054102  023737  001526  001606  CMP     @#PA2116,@#MXMMHI       ;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,@#$TMP10
 8890  054140  013737  001212  001324  MBTERR: MOV     @#$LPERR,@#$TMP11       ;SAVE LOOP ADDRESS
 8891  054146  012737  054210  001212          MOV     #1$,@#$LPERR            ;SET NEW LOOP ADR
 8892  054154  012703  000020                  MOV     #20,R3                  ;PUT DEVICE ID IN R3
 8893  054160  005737  001322                  TST     @#$TMP10
 8894  054164  001002                          BNE     2$
 8895  054166  104007                          ERROR   7
 8896  054170  000407                          BR      1$
 8897  054172  013737  001524  001226  2$:     MOV     @#PA1500,@#$GDDAT
 8898  054200  013737  001526  001230          MOV     @#PA2116,@#$BDDAT
 8899  054206  104013                          ERROR   13
 8900  054210  013737  001324  001212  1$:     MOV     @#$TMP11,@#$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
 3908  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,@#RESKT              ;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                          ;;*****************************************************************
```

```
                                        .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.
                                        ;;*****************************************************************
054300  104420                  LKSRV:  SAVREG
054302  004737  063752                  JSR     PC,@#LDKT               ;GO TO LOW CORE
054306  105237  001630                  INCB    @#LTICKS                ;INCREMENT TICK COUNT
054312  122737  000074  001630          CMPB    #60.,@#LTICKS           ;ONE SECOND YET?
054320  001014                          BNE     1$                      ;BRANCH IF NO
054322  105237  001631                  INCB    @#LTICKS+1              ;INCREMENT SECOND COUNT
054326  105037  001630                  CLRB    @#LTICKS                ;CLEAR SECOND COUNT
054332  122737  000074  001631          CMPB    #60.,@#LTICKS+1         ;ONE MINUTE YET?
054340  001004                          BNE     1$                      ;BRANCH IF NO
054342  105037  001631                  CLRB    @#LTICKS+1
054346  005237  001626                  INC     @#MTICKS                ;INCREMENT MINUTE COUNT
054352  004737  064050          1$:     JSR     PC,@#RESKT              ;RESTORE THE KT
054356  104422                          RESREG
054360  012737  000100  177546          MOV     #BIT6,@#LKS             ;CLEAR READY BIT IN CLOCK
054366  000002                          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

054370                          $SCOPE:
054370  032737  040000  177570          BIT     #SW14,@#SWR     ;;LOOP ON PRESENT TEST?
054376  001077                          BNE     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
054400  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
054402  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
054406  012737  054426  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
054414  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
054420  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
054424  000453                          BR      $SVLAD          ;;GO TO THE NEXT TEST
054426  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
054430  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
054434  000413                          BR      7$              ;;LOOP ON THE PRESENT TEST
054436                          6$:;#####END OF CODE FOR THE XOR TESTER#####
054436  105767  124542          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
054442  001421                          BEQ     3$              ;;BR IF NO
054444  126767  124547  124532          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
054452  101015                          BHI     3$              ;;BR IF NO
054454  032737  001000  177570          BIT     #BIT09,@#SWR    ;;LOOP ON ERROR?
054462  001404                          BEQ     4$              ;;BR IF NO
```

```
 8974  054464  016767  124522  124516  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 8975  054472  000441                  4$:     BR      $OVER
 8976  054474  105067  124504          4$:     CLRB    $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    $ERFLG,@#$TSTNM+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                                  .SBTTL  ERROR HANDLER ROUTINE
 9001
 9002                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 9003                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 9004                                  ;*AND GO TO $ERRTYP ON ERROR
 9005                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 9006                                  ;*SW15=1         HALT ON ERROR
 9007                                  ;*               HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
 9008                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
 9009                                  ;*SW10=1         BELL ON ERROR
 9010                                  ;*SW09=1         LOOP ON ERROR
 9011                                  ;*CALL
 9012                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 9013
 9014  054622                          $ERROR:
 9015  054622  116737  124356  001203          MOVB    $ERFLG,@#$TSTNM+1
 9016  054630  105267  124350          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 9017  054634  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 9018  054636  016737  124340  177570          MOV     $TSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 9019  054644  005737  177570                  TST     @#SWR           ;;HALT ON ERROR = 1?
 9020  054650  100001                          BPL     8$              ;;BRANCH IF NO
 9021  054652  000000                          HALT                    ;;YES--HALT
 9022  054654  032737  002000  177570  8$:     BIT     #BIT10,@#SWR    ;;BELL ON ERROR?
 9023  054662  001402                          BEQ     1$              ;;NO - SKIP
 9024  054664  104400  001332                  TYPE    ,$BELL          ;;RING BELL
 9025  054670  005267  124320          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 9026  054674  011667  124320                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 9027  054700  162767  000002  124312          SUB     #2,$ERRPC
 9028  054706  117767  124306  124302          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
 9029  054714  032737  020000  177570          BIT     #BIT13,@#SWR    ;;SKIP TYPEOUT IF SET
```

```
                                        BNE     2$                      ;;SKIP TYPEOUTS
                                        JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
                                        TYPE    ,$CRLF
                                2$:     TST     @#SWR                   ;;HALT ON ERROR
                                        BPL     9$                      ;;SKIP IF CONTINUE
                                        HALT                            ;;HALT ON ERROR!
                                9$:     CMP     #$ENDAD,42              ;;ACT-11?
                                        BNE     3$                      ;;BRANCH IF NO
                                        HALT                            ;;YES
                                3$:     BIT     #BIT09,@#SWR            ;;LOOP ON ERROR SWITCH SET?
                                        BEQ     4$                      ;;BR IF NO
                                        MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
                                4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
                                        BEQ     5$                      ;;BR IF NONE
                                        MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
                                5$:
                                        RTI                             ;;RETURN
                                ;;******************************************************************
                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                ;*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
                                ;;******************************************************************

                                $ERRTYP:SAVREG
                                        TYPE    ,$CRLF                  ;;"CARRIAGE RETURN" & "LINE FEED"
                                        JSR     PC,@#TYPTIME            ;GO TYPE THE TIME
                                        TYPE    ,MSG3
                                        TYPE    ,$CRLF
                                        MOV     $ERRPC,-(SP)            ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;TYPE THE VIRTUAL PC
                                        TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                        TYPE    ,8$
                                        MOV     @#VADR,R0               ;SAVE VADR
```

```
 9086  055046  013737  001220  001522          MOV     @#$ERRPC,@#VADR         ;SAVE THE VIR PC FOR CONVERSION
 9087  055054  122737  000014  001216          CMPB    #14,@#$ITEMB
 9088  055062  003403                          BLE     51$
 9089  055064  105737  001216                  TSTB    @#$ITEMB                ;ERROR ZERO?
 9090  055070  001005                          BNE     42$                     ;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,@#$DB2O              ;CONVERT TO ASCII
 9099  055132  062716  000003                  ADD     #3,(SP)                 ;GET RID OF 3 MS DIGITS
 9100  055136  012667  000002                  MOV     (SP)+,30$               ;SAVE POINTER TO ASCII
 9101  055142  104400                          TYPE                            ;TYPE IT
 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 $MAINT FOR TYPEOUT
 9108                                                                  ;;TYPE THE MAINTENANCE REG
 9109  055170  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 9110  055172  104400  056260                  TYPE    ,8$
 9111  055176  116746  124000                  MOVB    $TSTNM,-(SP)
 9112  055202  105066  000001                  CLRB    1(SP)
 9113  055206  104402                          TYPOC                   ;TYPE THE TEST NUMBER
 9114  055210  104400  056260                  TYPE    ,8$
 9115  055214  013746  001560                  MOV     @#SUBPASS,-(SP)
 9116  055220  162716  000060                  SUB     #60,(SP)
 9117  055224  104402                          TYPOC
 9118  055226  104400  056260                  TYPE    ,8$
 9119  055232  016746  123742                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
 9120                                                                  ;;TYPE THE PASS COUNT
 9121  055236  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 9122  055240  104400  001337                  TYPE    ,$CRLF
 9123  055244  005000                          CLR     R0
 9124  055246  153700  001216                  BISB    @#$ITEMB,R0             ;PICK UP THE INDEX
 9125  055252  001431                          BEQ     6$                      ;EXIT IF ZERO
 9126  055254  022700  000007          1$:     CMP     #7,R0           ;IS THIS ERROR 7?
 9127  055260  001551                          BEQ     15$             ;BRANCH IF YES
 9128  055262  005300                          DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
 9129  055264  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
 9130  055266  006300                          ASL     R0
 9131  055270  006300                          ASL     R0
 9132  055272  062700  002344                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
 9133  055276  012067  000004                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
 9134  055302  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
 9135  055304  104400                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
 9136  055306  000000          2$:     .WORD   0                       ;;"ERROR MESSAGE" POINTER GOES HERE
 9137  055310  104400  001337                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
 9138  055314  012067  000004          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
 9139  055320  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
 9140  055322  104400                          TYPE                    ;;TYPE THE "DATA HEADER"
 9141  055324  000000          4$:     .WORD   0                       ;;"DATA HEADER" POINTER GOES HERE
```

```
 9142  055326  104400  001337          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
 9143  055332  012001                  MOV     (R0)+,R1                ;;PICKUP "DATA TABLE" POINTER
 9144  055334  001004                  BNE     7$              ;;GO TYPE THE DATA
 9145  055336  104422          6$:     RESREG
 9146  055340  104400  001337          TYPE    ,$CRLF          ;;"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                          ;DATA FORMAT 0
 9161  055406  005202                  INC     R2              ;INCREMENT FORMAT POINTER
 9162  055410  013146                  MOV     @(R1)+,-(SP)    ;PUSH DATA TO BE TYPED
 9163  055412  104402                  TYPOC
 9164  055414  005711          13$:    TST     (R1)            ;ANY MORE DATA?
 9165  055416  001747                  BEQ     6$              ;BRANCH IF NO
 9166  055420  104400  056260          TYPE    ,8$             ;TYPE TWO SPACES
 9167  055424  000751                  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,@#$DB2O      ;CONVERT TO ASCII
 9174  055444  062716  000003          ADD     #3,(SP)         ;DELETE LEADING ZEROS
 9175  055450  012667  000002          MOV     (SP)+,12$       ;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 ASCIZ ADR
 9192  055520  017767  000004  000002  MOV     @25$,25$        ;GET ADR OF ASCIZ
 9193  055526  104400                  TYPE
 9194  055530  000000          25$:    .WORD
 9195  055532  000730                  BR      13$             ;CONTINUE
 9196                          ;;*****************************************************************
 9197                          ;DATA FORMAT 4
```

```
 9198  055534  005202                 40$:    INC     R2
 9199  055536  012167  000002                 MOV     (R1)+,44$       ;GET ADDRESS OF DATA
 9200  055542  104424                         FL2O                    ;CONVERT TO FLOATING FORMAT
 9201  055544  000000                 44$:    .WORD
 9202  055546  012667  000002                 MOV     (SP)+,45$       ;GET ADDRESS OF ASCIZ STRING
 9203  055552  104400                         TYPE                    ;TYPE THE DATA
 9204  055554  000000                 45$:    .WORD
 9205  055556  000716                         BR      13$
 9206                                 ;;*****************************************************************
 9207                                 ;DATA FORMAT 5
 9208  055560  005202                 60$:    INC     R2              ;INCREMENT FORMAT POINTER
 9209  055562  012167  000002                 MOV     (R1)+,61$       ;GET ADDRESS OF DATA
 9210  055566  104426                         FLD2O                   ;CONVERT TO FLOATING ASCIZ
 9211  055570  000000                 61$:    .WORD
 9212  055572  012667  000002                 MOV     (SP)+,62$       ;GET ADDRESS OF ASCIZ STRING
 9213  055576  104400                         TYPE                    ;TYPE THE DATA
 9214  055600  000000                 62$:    .WORD
 9215  055602  000704                         BR      13$
 9216                                 ;;*****************************************************************
 9217                                 ;ERROR 7 DECODE
 9218  055604  010300                 15$:    MOV     R3,R0           ;SAVE R3
 9219  055606  062700  064440                 ADD     #MSGINX,R0      ;GEN ADRS OF ASCIZ
 9220  055612  011067  000002                 MOV     (R0),16$
 9221  055616  104400                         TYPE
 9222  055620  000000                 16$:    .WORD
 9223  055622  104400  055630                 TYPE    ,65$            ;;TYPE ASCIZ STRING
 9224  055626  000404                         BR      64$             ;;GET OVER THE ASCIZ
 9225                                 ;;65$:  .ASCIZ  /FAILED/<CRLF>
 9226  055640                         64$:
 9227  055640  010300                         MOV     R3,R0           ;SAVE DEVICE ID
 9228  055642  022700  000010                 CMP     #10,R0          ;MASS BUS DEVICE?
 9229  055646  003403                         BLE     17$             ;BRANCH IF YES
 9230  055650  104400  064627                 TYPE    ,MSG12
 9231  055654  000411                         BR      18$
 9232                                 ;;*****************************************************************
 9233                                 ;MASS BUS ERR
 9234  055656  022703  000020         17$:    CMP     #20,R3          ;MBT ERROR?
 9235  055662  001426                         BEQ     26$             ;BRANCH IF MBT ERROR
 9236  055664  002435                         BLT     27$             ;BRANCH IF UBE ERROR
 9237  055666  104400  064742                 TYPE    ,MSG13
 9238  055672  022700  000012                 CMP     #12,R0          ;WAS IT RS?
 9239  055676  001140                         BNE     29$             ;BRANCH IF NO
 9240                                 ;;*****************************************************************
 9241                                 ;UNIBUS ERROR OR RS04 ERROR
 9242  055700  062700  064414         18$:    ADD     #REGINX,R0      ;FORM ADR OF REG TABLE
 9243  055704  011000                         MOV     (R0),R0         ;GET ADR OF REG TABLE
 9244  055706  022703  000002                 CMP     #2,R3           ;RP3 OR RK?
 9245  055712  001404                         BEQ     20$             ;BRANCH IF RK
 9246  055714  100406                         BMI     21$             ;BRANCH IF NOT RP03
 9247  055716  012704  000007                 MOV     #7,R4           ;SET RP03 SOB COUNT
 9248  055722  000423                         BR      22$
 9249  055724  012704  000006         20$:    MOV     #6,R4           ;SET RK05 SOB COUNT
 9250  055730  000420                         BR      22$
 9251  055732  012704  000011         21$:    MOV     #11,R4          ;SET RS04 SOB COUNT
 9252  055736  000415                         BR      22$
 9253                                 ;;*****************************************************************
```

```
                                ;MBT ERROR
055740  104400  065131        26$:    TYPE    ,MSG16
055744  012704  000011                MOV     #11,R4          ;SET MBT SOB COUNT
055750  062700  064414        28$:    ADD     #REGINX,R0
055754  011000                        MOV     (R0),R0         ;GET ADR OF MBT TABLE
055756  000405                        BR      22$             ;GO TYPE REGISTERS
                                ;UNIBUS  EXERCISER ERROR
055760  104400  065240        27$:    TYPE    ,MSG17
055764  012704  000004                MOV     #4,R4           ;SET UBE SOB COUNT
055770  000767                        BR      28$             ;GO TYPE UBE REGISTERS
055772  013046                22$:    MOV     @(R0)+,-(SP)    ;GET DATA IN REG
055774  104402                        TYPOC                   ;TYPE IT
055776  104400  056260                TYPE    ,8$             ;TYPE TWO SPACES
056002  077405                        SOB     R4,22$          ;CONTINUE
                                ;;*****************************************************************
                                ;THIS CODE TYPES A PHYSICAL BUS ADDRESS IF THE ERROR WAS AN RP03, RK05, OR UBE
056004  022703  000022                CMP     #22,R3          ;UBE ERROR?
056010  001454                        BEQ     73$             ;BRANCH IF YES
056012  022703  000002                CMP     #2,R3           ;RK05?
056016  002445                        BLT     32$             ;BRANCH IF NOT RK OR RP03
056020  001005                        BNE     70$             ;BRANCH IF RP03
                                ;RK05 ERROR
056022  104400  065434                TYPE    ,MSG22
056026  012700  002166                MOV     #RKCS,R0        ;GET ADR OF ADR OF RKCS REG
056032  000404                        BR      71$
                                ;RP03 ERROR
056034  012700  002144        70$:    MOV     #RP3CS,R0       ;GET ADR OF ADR OF RP3CS REG
056040  104400  065444                TYPE    ,MSG23
                                ;GET, CALCULATE, & TYPE PHYSICAL BUS ADDRESS
056044  013001                71$:    MOV     @(R0)+,R1       ;GET BUS ADR EXTENDED BITS
056046  005720                        TST     (R0)+           ;ADJUST R0
056050  013037  001732                MOV     @(R0)+,@#ERRBA  ;GET BUS ADRESS THAT FAILED
056054  072127  177774                ASH     #-4,R1          ;GET BITS 4&5 INTO BITS 0&1
056060  042701  177774                BIC     #177774,R1      ;GET RID OF UNUSED BITS
056064  010137  001734                MOV     R1,@#ERRBA+2    ;SAVE EXTENDED BITS
056070  162737  000002  001732  74$:    SUB     #2,@#ERRBA      ;DECREMENT BUS ADR
056076  005637  001734                SBC     @#ERRBA+2
056102  004737  062324                JSR     PC,@#PHYMAP     ;GO CONVERT TO 22 BIT PHYSICAL
056106  012746  001524                MOV     #PA1500,-(SP)
056112  004737  057544                JSR     PC,@#$DB2O      ;CONVERT TO ASCIZ STRING
056116  062716  000003                ADD     #3,(SP)         ;GET RID OF LEADING ZEROS
056122  012667  000002                MOV     (SP)+,72$
056126  104400                        TYPE
056130  000000                72$:    .WORD
056132  104400  001337        32$:    TYPE    ,$CRLF
056136  000167  177174                JMP     6$              ;EXIT
                                ;GET UBE VIRTUAL ADDRESS
056142  012700  002272        73$:    MOV     #UBETBL+2,R0    ;GET ADR OF UBE TABLE +2
056146  013037  001732                MOV     @(R0)+,@#ERRBA  ;GET BUS ADR THAT FAILED
056152  013037  001734                MOV     @(R0)+,@#ERRBA+2;GET BAE BITS
056156  042737  177774  001734        BIC     #177774,@#ERRBA+2;MASK OFF ADR BITS
056164  162737  000002  001732        SUB     #2,@#ERRBA
056172  005637  001734                SBC     @#ERRBA+2
056176  000734                        BR      74$             ;GO CONVERT & TYPE PHYSICAL ADR
                                ;;*****************************************************************
                                ;RP04 ERROR
```

```
 9310  056200  062700  064414          29$:    ADD     #REGINX,R0
 9311  056204  011000                          MOV     (R0),R0          ;FORM ADR OF RP04 TABLE
 9312  056206  012704  000011                  MOV     #11,R4           ;SET SOB COUNT
 9313  056212  013046                  31$:    MOV     @(R0)+,-(SP)     ;GET DATA TO BE TYPED
 9314  056214  104402                          TYPOC                    ;TYPE DATA
 9315  056216  104400  056260                  TYPE    ,8$
 9316  056222  077405                          SOB     R4,31$           ;CONTINUE
 9317  056224  104400  001337                  TYPE    ,$CRLF
 9318  056230  104400  001337                  TYPE    ,$CRLF
 9319  056234  012704  000004                  MOV     #4,R4            ;SET SOB COUNT
 9320  056240  104400  065052                  TYPE    ,MSG14
 9321  056244  013046                  50$:    MOV     @(R0)+,-(SP)     ;GET DTA TO BE TYPED
 9322  056246  104402                          TYPOC                    ;TYPE IT
 9323  056250  104400  056260                  TYPE    ,8$
 9324  056254  077405                          SOB     R4,50$           ;CONTINUE
 9325  056256  000725                          BR      32$
 9326  056260  020040      000         8$:     .ASCIZ  /  /             ;;TWO(2) SPACES
 9327          056264                          .EVEN
 9328                                  ;;*****************************************************************
 9329
 9330                                  .SBTTL  TYPE ROUTINE
 9331
 9332                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
 9333                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
 9334                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
 9335                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
 9336                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
 9337                                  ;*
 9338                                  ;*CALL:
 9339                                  ;*1) USING A TRAP INSTRUCTION
 9340                                  ;*      TYPE    ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
 9341                                  ;*OR
 9342                                  ;*      TYPE
 9343                                  ;*      MESADR
 9344                                  ;*
 9345                                  ;*2) USING A JSR INSTRUCTION
 9346                                  ;*      MOV     PS,-(SP)         ;;PUSH PROCESSOR STATUS WORD ON THE STACK
 9347                                  ;*      JSR     PC,$TYPE         ;;CALL TYPE ROUTINE
 9348                                  ;*      MESADDR                  ;;FIRST ADRESS OF MESSAGE
 9349
 9350  056264  105767  122763          $TYPE:  TSTB    $TPFLG           ;;IS THERE A TERMINAL?
 9351  056270  100002                          BPL     1$               ;;BR IF YES
 9352  056272  000000                          HALT                     ;;HALT HERE IF NO TERMINAL
 9353  056274  000407                          BR      3$               ;;LEAVE
 9354  056276  010046                  1$:     MOV     R0,-(SP)         ;;SAVE R0
 9355  056300  017600  000002                  MOV     @2(SP),R0        ;;GET ADDRESS OF ASCIZ STRING
 9356  056304  112046                  2$:     MOVB    (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 9357  056306  001005                          BNE     4$               ;;BR IF IT ISN'T THE TERMINATOR
 9358  056310  005726                          TST     (SP)+            ;;IF TERMINATOR POP IT OFF THE STACK
 9359  056312  012600                          MOV     (SP)+,R0         ;;RESTORE R0
 9360  056314  062716  000002          3$:     ADD     #2,(SP)          ;;ADJUST RETURN PC
 9361  056320  000002                          RTI                      ;;RETURN
 9362  056322  122716  000011          4$:     CMPB    #HT,(SP)         ;;BRANCH IF <HT>
 9363  056326  001426                          BEQ     8$
 9364  056330  122716  000200                  CMPB    #CRLF,(SP)       ;;BRANCH IF NOT
 9365  056334  001004                          BNE     5$
```

```
 9366  056336  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
 9367  056340  104400  001337                  TYPE    ,$CRLF
 9368  056344  000757                          BR      2$              ;;GET NEXT CHARACTER
 9369  056346  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
 9370  056352  126726  122674          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
 9371  056356  001352                          BNE     2$              ;;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     $TYPEX          ;;BRANCH IF YES
 9390  056436  105777  122602                  TSTB    @$TPS           ;WAIT UNTIL PRINTER IS READY
 9391  056442  100372                          BPL     $TYPEC
 9392  056444  116677  000002  122574          MOVB    2(SP),@$TPB     ;;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                                  .SBTTL  ROUTINE TO TYPE THE ELAPSED RUN TIME OF THE PROGRAM
 9405                                  ;*      THIS ROUTINE CONVERTS THE CONTENTS OF LOCATIONS ''LTICKS''
 9406                                  ;*      AND ''MTICKS'' TO SECONDS AND MINUTES/HOURS RESPECTIVELY
 9407                                  ;*      AND TYPES THEM IN THE FOLLOWING FORMAT:
 9408                                  ;*              HHH:MM:SS
 9409                                  ;;**********************************************************************
 9410  056506  104420                  TYPTIME:SAVREG
 9411  056510  004737  063752                  JSR     PC,@#LDKT       ;GO BACK TO LOW CORE
 9412  056514  113701  001631                  MOVB    @#LTICKS+1,R1   ;GET SECOND COUNT
 9413  056520  005000                          CLR     R0
 9414  056522  071027  000012                  DIV     #10.,R0
 9415  056526  062701  000060                  ADD     #60,R1
 9416  056532  110137  056744                  MOVB    R1,@#TIMEBUF+10
 9417  056536  010001                          MOV     R0,R1
 9418  056540  005000                          CLR     R0
 9419  056542  071027  000006                  DIV     #6,R0
 9420  056546  062701  000060                  ADD     #60,R1
 9421  056552  110137  056743                  MOVB    R1,@#TIMEBUF+7
```

```
 9422  056556  013701  001626          MOV     @#MTICKS,R1     ;GET MINUTE COUNT
 9423  056562  005000                  CLR     R0
 9424  056564  071027  000012          DIV     #10.,R0         ;GET HOURS AND MINUTES
 9425  056570  062701  000060          ADD     #60,R1          ;MAKE REMAINDER ASCII
 9426  056574  110167  000141          MOVB    R1,TIMEBUF+5    ;PUT IN BUFFER
 9427  056600  010001                  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
```

```
 9435  056626  005000                CLR     R0
```

```
9436  056630  071027  000012                  DIV     #10.,R0
9437  056634  062701  000060                  ADD     #60,R1
9438  056640  110167  000072                  MOVB    R1,TIMEBUF+2
9439  056644  005700                          TST     R0
9440  056646  001422                          BEQ     2$
9441  056650  010001                          MOV     R0,R1
9442  056652  005000                          CLR     R0
9443  056654  071027  000010                  DIV     #10,R0
9444  056660  062701  000060                  ADD     #60,R1
9445  056664  110167  000045                  MOVB    R1,TIMEBUF+1
9446  056670  005700                          TST     R0
9447  056672  001410                          BEQ     2$
9448  056674  010001                          MOV     R0,R1
9449  056676  005000                          CLR     R0
9450  056700  071027  000012                  DIV     #10.,R0
9451  056704  062701  000060                  ADD     #60,R1
9452  056710  110167  000020                  MOVB    R1,TIMEBUF
9453  056714  104400  056734          2$:     TYPE    ,TIMEBUF
9454  056720  104400  001337                  TYPE    ,$CRLF
9455  056724  004737  064050                  JSR     PC,@#RESKT      ;GO BACK TO ORIGINAL MEMORY
9456  056730  104422                          RESREG
9457  056732  000207                          RTS     PC
9458  056734     001     001     001  TIMEBUF:.BYTE   1,1,1,72,1,1,72,60,60,0
9459  056737     072     001     001
9460  056742     072     060     060
9461  056745     000
9462                                          .EVEN
9463                                  ;;*****************************************************************
9464                                  .SBTTL  ROUTINE TO TYPE THE AVAILABLE DEVICES AND UNIT NUMBERS
9465                                  ;*      THIS ROUTINE SEARCHES THE SYSTEM SIZE TABLE FOR NON-
9466                                  ;*      ZERO ENTRIES.WHEN IT FINDS ONE, IT TYPES THE NAME OF THE
9467                                  ;*      DEVICE AND THE UNIT NUMBERS THAT WERE FOUND TO BE
9468                                  ;*      AVAILABLE FOR THAT DEVICE.
9469                                  ;;*****************************************************************
9470  056746  104400  067176          TYPSIZ: TYPE    ,SWITCH
9471  056752  104400  070517                  TYPE    ,MSG30          ;NOTE SWITCH REG BIT 8 REVERSAL
9472  056756  104400  070364                  TYPE    ,MSG4
9473  056762  012700  000010                  MOV     #10,R0          ;SET SOB COUNT
9474  056766  005001                          CLR     R1
9475  056770  105761  001642          1$:     TSTB    SYSSIZE(R1)     ;DEVICE AVAILABLE?
9476  056774  001004                          BNE     2$              ;BRANCH IF YES
9477  056776  062701  000002          7$:     ADD     #2,R1           ;INCREMENT INDEX
9478  057002  077006                          SOB     R0,1$           ;CONTINUE
9479  057004  000207                          RTS     PC              ;RETURN
9480  057006  010102                  2$:     MOV     R1,R2           ;GET INDEX
9481  057010  062702  064440                  ADD     #MSGINX,R2      ;GET ADR OF MESSAGE ADR
9482  057014  011267  000002                  MOV     (R2),3$         ;GET ADDRESS OF MESSAGE
9483  057020  104400                          TYPE
9484  057022  000000                  3$:     .WORD
9485  057024  112767  000060  000034        MOVB    #60,4$          ;INIT UNIT NO. BUFFER (ASCII)
9486  057032  116102  001642                  MOVB    SYSSIZE(R1),R2  ;GET WORD WITH AVAILABLE UNITS
9487  057036  012703  000010                  MOV     #10,R3          ;SET SOB COUNT
9488  057042  006002                  6$:     ROR     R2              ;GET UNITS
9489  057044  103002                          BCC     5$              ;BRANCH IF NOT A UNIT
9490  057046  104400  057066                  TYPE    ,4$
9491  057052  005267  000010          5$:     INC     4$
```

```
                                        SOB     R3,6$           ;CONTINUE
                                        TYPE    ,$CRLF
                                        BR      7$
057066  000     054     040  4$:        .BYTE   0,54,40,0       ;NUMBER,COMMA,SPACE,TERMINATOR
057071  000
                        ;;**************************************************************************

                        .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                        ;*      .BYTE   M               ;;M=1 OR 0
                        ;*                                      ;;1=TYPE LEADING ZEROS
                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                        ;*
                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                        ;*$TYPOS OR $TYPOC
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                        ;*
                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
000001  000211         MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
000207                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
                        BR      $TYPON
000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
                        MOV     R3,-(SP)        ;;SAVE R3
                        MOV     R4,-(SP)        ;;SAVE R4
                        MOV     R5,-(SP)        ;;SAVE R5
000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
                        NEG     R4
000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
000125                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
                        CLR     R3              ;;CLEAR THE OUTPUT WORD
                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
                        BR      3$              ;;GO DO MSB
                2$:     ROL     R5              ;;FORM THIS DIGIT
                        ROL     R5
                        ROL     R5
                        MOV     R5,R3
                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
```

```
 9548  057214  105367  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
 9575                          ;;**************************************************************
 9576
 9577                          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 9578
 9579                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 9580                          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 9581                          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 9582                          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 9583                          ;*REPLACED WITH SPACES.
 9584                          ;*CALL:
 9585                          ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
 9586                          ;*      TYPDS                   ;;GO TO THE ROUTINE
 9587
 9588  057320                  $TYPDS:
 9589  057320  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 9590  057322  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 9591  057324  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 9592  057326  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 9593  057330  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 9594  057332  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 9595  057336  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 9596  057342  100004                          BPL     1$              ;;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                          BMI     7$              ;;BR IF YES
 9613  057416  106316                  5$:     ASLB    (SP)            ;;MSD?
 9614  057420  103003                          BCC     6$              ;;BR IF NO
 9615  057422  116663  000001  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                  $DBLK:  .BLKW   4
 9643                                  ;;**********************************************************************
 9644
 9645                                  .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
 9646
 9647                                  ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
 9648                                  ;*UNSIGNED OCTAL ASCIZ NUMBER.
 9649                                  ;*CALL
 9650                                  ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
 9651                                  ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
 9652                                  ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
 9653
 9654
 9655  057544  104420                  $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
 9656  057546  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
 9657  057552  012705  057701                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
 9658  057556  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
 9659  057562  012703  177770                  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$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
 9664  057576  010002                        MOV     R0,R2           ;;GET THIS DIGIT
 9665  057600  005304                        DEC     R4              ;;COUNT THIS CHARACTER
 9666  057602  003016                        BGT     3$              ;;BR IF NOT THE LAST DIGIT
 9667  057604  001414                        BEQ     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)        ;;ASCIZ 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       MOVB    #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  057702                        $SAVREG:
 9705  057702  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 9706  057704  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 9707  057706  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 9708  057710  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 9709  057712  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 9710  057714  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 9711  057716  016646  000022               MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
 9712  057722  016646  000022               MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
 9713  057726  016646  000022               MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
 9714  057732  016646  000022               MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
 9715  057736  000002                        RTI
```

```
                                                    ;*RESTORE R0-R5
                                                    ;*CALL:
                                                    ;*      RESREG
057740                                              $RESREG:
057740  012666  000022                                      MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
057744  012666  000022                                      MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
057750  012666  000022                                      MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
057754  012666  000022                                      MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
057760  012605                                              MOV     (SP)+,R5        ;;POP STACK INTO R5
057762  012604                                              MOV     (SP)+,R4        ;;POP STACK INTO R4
057764  012603                                              MOV     (SP)+,R3        ;;POP STACK INTO R3
057766  012602                                              MOV     (SP)+,R2        ;;POP STACK INTO R2
057770  012601                                              MOV     (SP)+,R1        ;;POP STACK INTO R1
057772  012600                                              MOV     (SP)+,R0        ;;POP STACK INTO R0
057774  000002                                              RTI
                                                    ;;*****************************************************************
                                                    .SBTTL  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.
                                                    ;;*****************************************************************
057776  104420                                      $FL20:  SAVREG
060000  017600  000000                                      MOV     @(SP),R0        ;GET ADDRESS OF DATA
060004  062716  000002                                      ADD     #2,(SP)         ;ADJUST RETURN PC
060010  016001  000002                                      MOV     2(R0),R1        ;PUT SECOND DATA WORD IN R1
060014  011000                                              MOV     (R0),R0         ;PUT FIRST DATA WORD IN R0
060016  012704  001367                                      MOV     #$FLBUFF+23,R4  ;GET ADDRESS OF BUFFER END IN R4
060022  112744  000000                                      MOVB    #0,-(R4)        ;PUT TERMINATOR IN BUFFER
060026  012705  000005                                      MOV     #5,R5           ;SET SOB COUNT FOR FRACTION DIGITS
060032  010103                                      1$:     MOV     R1,R3           ;GET LSB'S OF FRACTION
060034  042703  177770                                      BIC     #^C7,R3         ;SAVE LS 3 BITS
060040  062703  000060                                      ADD     #60,R3          ;MAKE THEM ASCII
060044  110344                                              MOVB    R3,-(R4)        ;STORE IN BUFFER
060046  073027  177775                                      ASHC    #-3,R0          ;SHIFT NUMBER TO NEXT 3 BITS
060052  077511                                              SOB     R5,1$           ;CONTINUE FOR 7 DIGITS
060054  010103                                              MOV     R1,R3           ;GET NEXT DIGITS
060056  042703  177776                                      BIC     #^C1,R3         ;ONLY WANT 1 BIT
060062  062703  000060                                      ADD     #60,R3          ;MAKE THEM ASCII
060066  110344                                              MOVB    R3,-(R4)        ;STORE IN BUFFER
060070  112744  000040                                      MOVB    #40,-(R4)       ;PUT SPACE IN BUFFER
060074  073027  177777                                      ASHC    #-1,R0
060100  012705  000002                                      MOV     #2,R5           ;SET SOB COUNT
060104  010103                                      3$:     MOV     R1,R3           ;GET LOW WORD
060106  042703  177770                                      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
9805                            ;;******************************************************************
9806                            .SBTTL  CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCIZ
9807                            ;*
9808                            ;*THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
9809                            ;*ASCIZ STRING IN THE FOLLOWING FORMAT:
9810                            ;*
9811                            ;*              U  VVV  WWW XXXXXX YYYYYY ZZZZZZ
9812                            ;*
9813                            ;*      WHERE   U = SIGN BIT
9814                            ;*              V = 8-BIT EXPONENT (RIGHT JUSTIFIED)
9815                            ;*              W = FRACTION BITS<57:51> (RIGHT JUSTIFIED)
9816                            ;*              X = FRACTION BITS <50:35>
9817                            ;*              Y = FRACTION BITS <34:19>
9818                            ;*              Z = FRACTION BITS <18:03>
9819                            ;*
9820                            ;*IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
9821                            ;*NUMBER IN THE WORD FOLLOWING THE CALL.
9822                            ;*IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
9823                            ;;******************************************************************
9824  060264  104420            $FLD20: SAVREG
9825  060266  017667  000000  000006    MOV     @(SP),1$        ;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$,R1           ;GET ADDRESS OF DATA TO CONVERT
 9834  060324  062701  000004                  ADD     #4,R1           ;ADJUST TO LOWER 32 BITS
 9835  060330  012102                          MOV     (R1)+,R2        ;SAVE THE DATA
 9836  060332  012103                          MOV     (R1)+,R3
 9837  060334  012701  000002                  MOV     #2,R1           ;SET LOOP COUNT
 9838  060340  012704  000005          3$:     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     $BUFF,4(SP)     ;OF BUFFER ON STACK
 9856  060434  000006                          RTT                     ;RETURN
 9857
 9858                          ;;*****************************************************************
 9859
 9860                          .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
 9861
 9862                          ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
 9863                          ;*WITH A RANGE OF 0 TO 2(+33)-1.
 9864                          ;*CALL:
 9865                          ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
 9866                          ;*      RETURN                  ;;RETURN HERE THE RANDOM
 9867                          ;*                              ;;NUMBER WILL BE IN
 9868                          ;*                              ;;$HINUM,$LONUM
 9869
 9870  060436                  $RAND:
 9871  060436  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 9872  060440  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 9873  060442  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 9874  060444  016700  121100                  MOV     $LONUM,R0       ;;SET R0 WITH LOW
 9875  060450  016701  121076                  MOV     $HINUM,R1       ;;SET R1 WITH HIGH
 9876  060454  012702  177771                  MOV     #-7,R2          ;;SET SHIFT COUNT
 9877  060460  006300                  1$:     ASL     R0              ;;SHIFT R0 LEFT AND
 9878  060462  006101                          ROL     R1              ;;ROTATE CARRY INTO R1 AND
 9879  060464  005202                          INC     R2              ;;CHECK FOR DONE
 9880  060466  001374                          BNE     1$              ;;CONTINUE SHIFT LOOP
 9881  060470  066700  121054                  ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
 9882  060474  005501                          ADC     R1              ;;PROPOGATE CARRY
 9883  060476  066701  121050                  ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
```

```
                        ADD     #1057,R0        ;;ADD LOW CONSTANT
                        ADC     R1              ;;PROPUGATE CARRY
                        ADD     #47401,R1       ;;ADD HIGH CONSTANT
                        MOV     R0,$LONUM       ;;SAVE R0
                        MOV     R1,$HINUM       ;;SAVE R1
                        MOV     (SP)+,R2        ;;POP STACK INTO R2
                        MOV     (SP)+,R1        ;;POP STACK INTO R1
                        MOV     (SP)+,R0        ;;POP STACK INTO R0
                        RTS     PC              ;;RETURN
                ;;******************************************************************
                .SBTTL  FLOATING POINT NUMBER GENERATOR
                ;*      THIS ROUTINE GENERATES TWO RANDOM FLOATING POINT NUMBERS
                ;*      IN EITHER SINGLE OR DOUBLE PRECISION. FOR SINGLE PRECISION
                ;*      THE NUMBERS ARE STORED IN $TMP0 AND $TMP2. DOUBLE PRECISION
                ;*      NUMBERS ARE STORED IN $TMP0 AND $TMP4.
                ;*      IN EITHER SINGLE OR DOUBLE THE EXTENDED EXPONENT IS STORED
                ;*      IN $REG0 AND $REG1.
                ;;******************************************************************
060534  012767  000002  000130  FLTDBL: MOV     #2,SOBDBL       ;SET LOOP FOR 2, FOUR WORD NUMBERS
060542  016700  000124          FLTSGL: MOV     SOBDBL,R0       ;SET WORD LENGTH LOOP
060546  012702  001302                  MOV     #$TMP0,R2       ;GET ADDRESS TO STORE WORDS IN
060552  012701  000002          2$:     MOV     #2,R1           ;SET NUMBER OF WORDS TO 2
060556  004767  177654          1$:     JSR     PC,$RAND        ;GET RANDOM NUMBER
060562  022701  000002                  CMP     #2,R1           ;FIRST TIME?
060566  001404                          BEQ     3$              ;BRANCH IF YES
060570  022767  000002  000074          CMP     #2,SOBDBL       ;DOUBLE PRECISION?
060576  001407                          BEQ     4$              ;BRANCH IF YES
060600  016703  120746          3$:     MOV     $HINUM,R3       ;GET EXPONENT PART
060604  042703  000177                  BIC     #177,R3         ;CHECK FOR MINUS ZERO
060610  022703  100000                  CMP     #BIT15,R3
060614  001760                          BEQ     1$              ;BRANCH IF MINUS ZERO
060616  016722  120730          4$:     MOV     $HINUM,(R2)+    ;SAVE HINUM
060622  016722  120722                  MOV     $LONUM,(R2)+    ;SAVE LONUM
060626  077125                          SOB     R1,1$           ;CONTINUE
060630  077030                          SOB     R0,2$           ;CONTINUE FOR DOUBLE PREC
060632  012746  001302                  MOV     #$TMP0,-(SP)    ;PUT ADDRESS OF NUMBER ON STACK
060636  012746  001002                  MOV     #1002,-(SP)     ;PUT CONTROL WORD ON STACK
060642  022767  000002  000022          CMP     #2,SOBDBL       ;DOUBLE PREC?
060650  001002                          BNE     5$              ;BRANCH IF NO
060652  012716  001004                  MOV     #1004,(SP)      ;CHANGE CONTROL WORD
060656  004767  000012          5$:     JSR     PC,EXPEXT       ;CALCULATE EXT EXPONENTS
060662  012767  000001  000002          MOV     #1,SOBDBL       ;INIT SOBDBL FOR SINGLE PREC
060670  000207                          RTS     PC              ;RETURN
060672  000001                  SOBDBL: .WORD   1
                ;;******************************************************************
                .SBTTL  FLOATING POINT EXPONENT EXTENSION
                ;*      THIS ROUTINE CONVERTS THE ACTUAL EXPONENT OF A FLOATING POINT
                ;*      NUMBER INTO AN ACTUAL EXPONENT OF 200 AND AN EXTENDED
                ;*      EXPONENT EQUAL TO THE DIFFERANCE BETWEEN THE ORIGINAL
                ;*      ACTUAL EXPONENT AND 200.
                ;*
                ;*      THE ROUTINE IS ENTERED WITH A CONTROL WORD ON THE STACK.
                ;*      BIT 15 OF THE CONTROL WORD INDICATES WHETHER THE NUMBER
                ;*      IS IN MEMORY (<15>=0) OR IN AN ACCUMULATOR (<15>=1).
                ;*      IF THE NUMBER IS IN AN ACCUMULATOR, BITS <9:8> INDICATE
                ;*      THE ACCUMULATOR NUMBER. IF THE NUMBER(S) IS IN MEMORY,
```

```
                                ;*      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).
                                ;;*****************************************************************
060674  012605                  EXPEXT: MOV     (SP)+,R5        ;SAVE RETURN PC
060676  012600                          MOV     (SP)+,R0        ;GET CONTROL WORD
060700  100437                          BMI     1$              ;BRANCH IF ACC CONVERSION
060702  012601                          MOV     (SP)+,R1        ;GET START ADDRESS
060704  162700  000400                  SUB     #400,R0
060710  012702  001302                  MOV     #$TMP0,R2       ;GET OFFSET FROM $TMPC
060714  160102                          SUB     R1,R2
060716  005402                          NEG     R2
060720  006202                          ASR     R2
060722  062702  001256                  ADD     #$REG0,R2       ;GEN ADDRESS OF EXT WORD
060726  011103                  3$:     MOV     (R1),R3         ;GET DATA
060730  042703  100177                  BIC     #100177,R3      ;GET EXPONENT
060734  072327  177771                  ASH     #-7,R3          ;RIGHT JUSTIFY EXPONENT
060740  162703  000200                  SUB     #200,R3         ;CONVERT TO 2'S COMPLIMENT
060744  010312                          MOV     R3,(R2)         ;ADD TO EXTENDED EXPONENT
060746  042711  077600                  BIC     #77600,(R1)     ;MAKE ACTUAL
060752  052711  040000                  BIS     #BIT14,(R1)     ;EXPONENT 200
060756  162700  000400                  SUB     #400,R0         ;ANY MORE WORDS?
060762  100435                          BMI     2$              ;BRANCH IF NO
060764  110003                          MOVB    R0,R3           ;GET WORD LENGTH
060766  006303                          ASL     R3
060770  060301                          ADD     R3,R1           ;SELECT NEXT NUMBER ADDRESS
060772  062702  000002                  ADD     #2,R2           ;SELECT NEXT EXTENDED ADDRESS
060776  000753                          BR      3$              ;CONTINUE
                                ;ACCUMULATOR CONVERSION
061000  072027  177776          1$:     ASH     #-2,R0          ;GET ACCUMULATOR NUMBER
061004  042700  177477                  BIC     #177477,R0      ;
061010  010002                          MOV     R0,R2           ;GENERATE
061012  072227  177773                  ASH     #-5,R2          ;ADDRESS OF
061016  062702  001412                  ADD     #$AC0,R2        ;EXTENDED EXPONENT
061022  042767  000300  000004          BIC     #300,5$         ;GENERATE INSTRUCTION
061030  050067  000000                  BIS     R0,5$           ;TO GET EXPONENT
061034  175003                  5$:     STEXP   AC0,R3          ;GET EXPONENT
061036  060312                          ADD     R3,(R2)         ;ADD TO EXTENDED EXPONENT
061040  005003                          CLR     R3
061042  042767  000300  000004          BIC     #300,4$         ;GENERATE INSTRUCTION
061050  050067  000000                  BIS     R0,4$           ;TO LOAD EXPONENT BACK TO ACC
061054  176403                  4$:     LDEXP   R3,AC0          ;LOAD EXPONENT OF 200
061056  010546                  2$:     MOV     R5,-(SP)        ;RESTORE RETURN PC
061060  000207                          RTS     PC              ;RETURN
                                ;;*****************************************************************


                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;POWER DOWN ROUTINE
061062  012737  061210  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
061070  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
061076  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
061100  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
061102  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
```

```
 9996  061104  010346                          MOV     R3,-(SP)          ;;PUSH R3 ON STACK
 9997  061106  010446                          MOV     R4,-(SP)          ;;PUSH R4 ON STACK
 9998  061110  010546                          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
 9999  061112  010667  000076                  MOV     SP,$SAVR6         ;;SAVE SP
10000  061116  012737  061130  000024          MOV     #$PWRUP,@#PWRVEC  ;;SET UP VECTOR
10001  061124  000000                          HALT
10002  061126  000776                          BR      .-2               ;;HANG UP
10003
10004                                  ;POWER UP ROUTINE
10005  061130  016706  000060          $PWRUP: MOV     $SAVR6,SP         ;;GET SP
10006  061134  005067  000054                  CLR     $SAVR6            ;;WAIT LOOP FOR THE TTY
10007  061140  005267  000050          1$:     INC     $SAVR6            ;;WAIT FOR THE INC
10008  061144  001375                          BNE     1$                ;;OF  WORD
10009  061146  012605                          MOV     (SP)+,R5          ;;POP STACK INTO R5
10010  061150  012604                          MOV     (SP)+,R4          ;;POP STACK INTO R4
10011  061152  012603                          MOV     (SP)+,R3          ;;POP STACK INTO R3
10012  061154  012602                          MOV     (SP)+,R2          ;;POP STACK INTO R2
10013  061156  012601                          MOV     (SP)+,R1          ;;POP STACK INTO R1
10014  061160  012600                          MOV     (SP)+,R0          ;;POP STACK INTO R0
10015  061162  012737  061062  000024          MOV     #$PWRDN,@#PWRVEC  ;;SET UP THE POWER DOWN VECTOR
10016  061170  012737  000340  000026          MOV     #340,@#PWRVEC+2   ;;PRIO:7
10017  061176  104400                          TYPE                      ;REPORT THE POWER FAILURE
10018  061200  061216                  $PWRMG: .WORD   $POWER            ;;POWER FAIL MESSAGE POINTER
10019  061202  012716                          MOV     (PC)+,(SP)        ;;RESTART AT START
10020  061204  003542                  $PWRAD: .WORD   START             ;;RESTART ADDRESS
10021  061206  000002                          RTI
10022  061210  000000                  $ILLUP: HALT                      ;;THE POWER UP SEQUENCE WAS STARTED
10023  061212  000776                          BR      .-2               ;; BEFORE THE POWER DOWN WAS COMPLETE
10024  061214  000000                  $SAVR6: 0                         ;;PUT THE SP HERE
10025  061216  005015  047520  042527  $POWER: .ASCIZ  <15><12>'POWER'
10026  061224  000122
10027                                          .EVEN
10028                                  ;;*****************************************************************
10029
10030                                  .SBTTL  TTY INPUT ROUTINE
10031
10032                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
10033                                  ;*CALL:
10034                                  ;*      RDCHR                     ;;INPUT A SINGLE CHARACTER FROM THE TTY
10035                                  ;*      RETURN HERE               ;;CHARACTER IS ON THE STACK
10036                                  ;*                                ;;WITH PARITY BIT STRIPPED OFF
10037                                  ;
10038
10039  061226  011646                  $RDCHR: MOV     (SP),-(SP)        ;;PUSH DOWN THE PC
10040  061230  016666  000004  000002          MOV     4(SP),2(SP)       ;;SAVE THE PS
10041  061236  105777  117776          1$:     TSTB    @$TKS             ;;WAIT FOR
10042  061242  100375                          BPL     1$                ;;A CHARACTER
10043  061244  117766  117772  000004          MOVB    @$TKB,4(SP)       ;;READ THE TTY
10044  061252  042766  177600  000004          BIC     #^C<177>,4(SP)    ;;GET RID OF JUNK IF ANY
10045  061260  000002                          RTI                       ;;GO BACK TO USER
10046                                  ;;*****************************************************************
10047                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
10048                                  ;*CALL:
10049                                  ;*      RDLIN                     ;;INPUT A STRING FROM THE TTY
10050                                  ;*      RETURN HERE               ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
10051                                  ;*                                ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
```

```
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$:     TYPE    ,$QUES          ;;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    ,$LF            ;;TYPE A LINE FEED
10069  061344  012603                          MOV     (SP)+,R3        ;;RESTORE R3
10070  061346  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
10071  061350  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
10072  061356  012766  061370  000004          MOV     #$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 ASCIZ 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     #^C7777,(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$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
10127  061530  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
10128  061532  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
10129  061534  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
10130  061536  000002                  RTI                     ;;RETURN
10131
10132  061540  005726          5$:     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     PC,$SIZE
10144                          ;*      RETURN
10145                          ;*$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 & PC
10166  061572  013746  000006                  MOV     @#ERRVEC+2,-(SP)
10167  061576  013746  000114                  MOV     @#114,-(SP)       ;;SAVE PRESENT PARITY VECTOR PS & PC
10168  061602  013746  000116                  MOV     @#116,-(SP)
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                  $KT11:  .WORD   200               ;;SET TO USE MEMORY MANAGEMENT
10174  061626  100065                          BPL     $CORE             ;;BR IF NO
10175  061630  012737  061774  000004          MOV     #$KTNEX,@#ERRVEC  ;;SET FOR TIMEOUT
10176  061636  005737  177572                  TST     @#SR0             ;;KT11 ARE YOU THERE?
10177  061642  052767  100000  177754          BIS     #100000,$KT11     ;;YES--SET KT11 KEY
10178  061650  005046                          CLR     -(SP)             ;;INITIALIZE FOR 'PAR' LOADING
10179  061652  012702  172340                  MOV     #KIPAR0,R2        ;;ADDRESS OF FIRST 'PAR'
10180  061656  012703  000010                  MOV     #^D8,R3           ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
10181  061662  012762  077406  177740  1$:     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                          SOB     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 REG.
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     #$KTOUT,@#ERRVEC  ;;SET FOR TIME OUT
10193  061740  012737  062106  000114          MOV     #$MTMOUT,@#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  $CORE:  MOV     #$CROUT,@#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     PC
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                     ;;IF NOT, CONTINUE
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
10235                                .SBTTL  TRAP DECODER
10236
10237                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE ''TRAP'' INSTRUCTION
10238                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
10239                                ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
10240                                ;*GO TO THAT ROUTINE.
10241
10242  062144  010046                $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
10243  062146  016600  000002                MOV     2(SP),R0        ;;GET TRAP ADDRESS
10244  062152  005740                        TST     -(R0)           ;;BACKUP BY 2
10245  062154  111000                        MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
10246  062156  016000  062164                MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
10247  062162  000200                        RTS     R0              ;;GO TO ROUTINE
10248
10249
10250                                .SBTTL  TRAP TABLE
10251
10252                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
10253                                ;*BY THE ''TRAP'' INSTRUCTION.
10254
10255                                ;       ROUTINE
10256                                ;       -------
10257  062164                        $TRPAD:
10258  062164  056264                        $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
10259  062166  057116                        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
10260  062170  057072                        $TYPOS  ;;CALL=TYPOS    TRAP+4(104404)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
10261  062172  057132                        $TYPON  ;;CALL=TYPON    TRAP+6(104406)  TYPE OCTAL NUMBER (AS PER LAST CALL)
10262  062174  057320                        $TYPDS  ;;CALL=TYPDS    TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
10263  062176  061226                        $RDCHR  ;;CALL=RDCHR    TRAP+12(104412) TTY TYPEIN CHARACTER ROUTINE
10264  062200  061262                        $RDLIN  ;;CALL=RDLIN    TRAP+14(104414) TTY TYPEIN STRING ROUTINE
10265  062202  061400                        $RDDEC  ;;CALL=RDDEC    TRAP+16(104416) READ A DECIMAL NUMBER FROM TTY
10266  062204  057702                        $SAVREG ;;CALL=SAVREG   TRAP+20(104420) SAVE R0-R5 ROUTINE
10267  062206  057740                        $RESREG ;;CALL=RESREG   TRAP+22(104422) RESTORE R0-R5 ROUTINE
10268  062210  057776                        $FL20   ;;CALL=FL20     TRAP+24(104424)
10269  062212  060264                        $FLD20  ;;CALL=FLD20    TRAP+26(104426)
10270                                ;;*****************************************************************
10271                                .SBTTL  UNIBUS EXERCISER INITIALIZATION ROUTINE
10272                                ;*THIS ROUTINE INITIALIZES THE BASE ADDRESS FOR THE
10273                                ;*UNIBUS EXERCISER AND LOADS UP THE EXERCISER REGISTERS.
10274                                ;;*****************************************************************
10275  062214  012767  002344  117500 UBEINIT:MOV     #$ERRTB,UBESAV          ;BASE ADDRESS OF UBE TRANSFER
```

```
10276  062222  005067  117476                  CLR     UBESAV+2
10277  062226  012767  002344  117472          MOV     #$ERRTB,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  062324  104420                  PHYMAP: SAVREG
10304  062326  013703  001732                  MOV     @#ERRBA,R3      ;GET BUS ADDRESS <15:00>
10305  062332  013702  001734                  MOV     @#ERRBA+2,R2    ;GET BUS ADDRESS <17:16>
10306  062336  042702  177774                  BIC     #177774,R2      ;
10307  062342  032737  000040  172516          BIT     #BIT5,@#MMR3    ;MAP ON?
10308  062350  001005                          BNE     1$              ;BRANCH IF YES
10309  062352  010337  001524                  MOV     R3,@#PA1500     ;PHY ADR=BUS ADR
10310  062356  010237  001526                  MOV     R2,@#PA2116
10311  062362  000421                          BR      MAPEND
10312  062364  010305                  1$:     MOV     R3,R5           ;SAVE ADR BITS <15:00>
10313  062366  073227  000005                  ASHC    #5,R2           ;GET MAP REG SELECT BITS
10314  062372  042702  000003                  BIC     #3,R2           ;
10315  062376  062702  170200                  ADD     #MAPL0,R2       ;FORM ADDRESS OF MAP REG
10316  062402  012237  001524                  MOV     (R2)+,@#PA1500  ;GET CONTENTS OF MAP REG LO
10317  062406  011237  001526                  MOV     (R2),@#PA2116   ;GET CONTENTS OF MAP REG HI
10318  062412  042705  160000                  BIC     #160000,R5      ;FORM PHYSICAL ADDRESS
10319  062416  060537  001524                  ADD     R5,@#PA1500     ;THAT TIMED OUT
10320  062422  005537  001526                  ADC     @#PA2116        ;
10321  062426  104422                  MAPEND: RESREG
10322  062430  000207                          RTS     PC
10323
10324                                  ;;*****************************************************************
10325                                  .SBTTL  CONVERT A VIRTUAL ADDRESS TO A PHYSICAL ADDRESS
10326                                  ;*      THIS ROUTINE CONVERTS A 16-BIT VIRTUAL ADDRESS TO A
10327                                  ;*      22-BIT PHYSICAL ADDRESS. THE VIRTUAL ADDRESS IS
10328                                  ;*      ASSUMED TO BE IN LOCATION 'VADR'' AND THE PHYSICAL
10329                                  ;*      ADDRESS IS PLACED IN LOCATIONS 'PA2116'' AND 'PA1500''.
10330                                  ;*
10331                                  ;*      IF MEMORY MANAGEMENT IS OFF THE PHYSICAL ADDRESS IS
```

```
10332                                   ;*      GENERATED BY SUBTRACTING THE CONTENTS OF LOCATION
10333                                   ;*      'FACTOR' FROM THE VIRTUAL ADDRESS. THIS LOCATION
10334                                   ;*      CONTAINS THE BYTE OFFSET BETWEEN THE RELOCATED CODE
10335                                   ;*      AND THE NON-RELOCATED CODE.
10336                                   ;*
10337                                   ;*      IF MEMORY MANAGEMENT IS ON, THE CONTENTS OF THE
10338                                   ;*      APPROPRIATE PAR REGISTER IS ADDED(AFTER ADJUSTMENT)
10339                                   ;*      TO THE LEAST SIGNIFICANT 13 BITS OF THE VIRTUAL ADDRESS.
10340                                   ;;*****************************************************************
10341  062432  104420                  CNVADR: SAVREG
10342  062434  013703  001522                  MOV     @#VADR,R3               ;GET VIRTUAL ADDRESS TO CONVERT
10343  062440  105737  001531                  TSTB    @#MMON                  ;IS MEMORY MGMT ON?
10344  062444  001426                          BEQ     1$                      ;BRANCH IF NO
10345  062446  005002                          CLR     R2
10346  062450  073227  000003                  ASHC    #3,R2                   ;GET PAR SELECT BITS
10347  062454  072327  177775                  ASH     #-3,R3                  ;RETURN VIR ADDR TO ORIGINAL
10348  062460  042703  160000                  BIC     #160000,R3              ;MAKE SURE SIGN DIDN'T EXTEND
10349  062464  006102                          ROL     R2                      ;MAKE R2 EVEN FOR WORD ADDRESSING
10350  062466  062702  172340                  ADD     #KIPAR0,R2              ;GET ADDRESS OF PAR
10351  062472  011205                          MOV     (R2),R5                 ;GET PAR DATA
10352  062474  005004                          CLR     R4                      ;SETUP R4
10353  062476  073427  000006                  ASHC    #6,R4                   ;SHIFT PAR DATA
10354  062502  060305                          ADD     R3,R5                   ;FORM PHYSICAL ADDRESS
10355  062504  005504                          ADC     R4
10356  062506  010437  001526          2$:     MOV     R4,@#PA2116             ;SAVE PHYSICAL
10357  062512  010537  001524                  MOV     R5,@#PA1500             ;ADDRESS
10358  062516  104422                          RESREG
10359  062520  000207                          RTS     PC                      ;RETURN
10360  062522  163703  001534          1$:     SUB     @#FACTOR,R3             ;FORM PHYSICAL ADDRESS
10361  062526  005004                          CLR     R4
10362  062530  010305                          MOV     R3,R5
10363  062532  000765                          BR      2$                      ;RETURN
10364
10365                                   ;;*****************************************************************
10366                                   .SBTTL  ROUTINE TO CHECK RELOCATED DATA
10367                                   ;*ROUTINE TO CHECK DATA RELOCATED
10368                                   ;*CALL: R0=     HIGHEST ADDRESS +2 OF SOURCE DATA
10369                                   ;*      R2=     HIGHEST ADDRESS +2 OF DEST DATA
10370                                   ;*      R5=     LOWEST ADDRESS OF THE SOURCE DATA
10371                                   ;*
10372                                   ;*      THIS ROUTINE USES A COMPARE INSTRUCTION TO CHECK
10373                                   ;*      THE DATA THAT WAS RELOCATED. IF A PARITY ERROR OCCURS
10374                                   ;*      DURING THIS CHECK A SPECIAL ERROR MESSAGE IS TYPED
10375                                   ;*      INSTEAD OF THE UNEXPECTED TRAP MESSAGE.
10376                                   ;;*****************************************************************
10377  062534  012703  004000          CHKDAT: MOV     #2048.,R3               ;COUNTER
10378  062540  012737  062632  000114          MOV     #2$,@#CACHVEC           ;SETUP PARITY VECTOR
10379  062546  024042                  7$:     CMP     -(R0),-(R2)             ;CHECK DATA
10380  062550  001026                          BNE     99$
10381  062552  005112                          COM     (R2)                    ;COMPLEMENT DEST DATA
10382  062554  005112                          COM     (R2)                    ;TWICE
10383  062556  021210                          CMP     (R2),(R0)               ;CHECK DATA
10384  062560  001022                          BNE     99$
10385  062562  020005                  1$:     CMP     R0,R5                   ;BRANCH IF ALL DATA CHECKED
10386  062564  001414                          BEQ     3$
10387  062566  105737  001531                  TSTB    @#MMON                  ;MEMORY MANAGMENT ON?
```

```
10388  062572  001765                    BEQ     7$                      ;BR BACK IF NOT
10389  062574  077314                    SOB     R3,7$                   ;REPEAT 4096 TIMES
10390  062576  012703  010000            MOV     #4096.,R3               ;SET UP COUNTER AGAIN
10391  062602  162737  000200  172350    SUB     #200,@#KIPAR4           ;MAP TO NEXT LOWER 4K OF SPACE
10392  062610  012702  120000            MOV     #120000,R2              ;START AT TOP OF 4K SPACE + 2
10393  062614  000754                    BR      7$                      ;CKECK MORE
10394  062616  012737  063464  000114  3$: MOV     #.PARSRV,@#CACHVEC      ;RESTORE CACHVEC
10395  062624  000207                    RTS     PC                      ;RETURN
10396  062626  000262                  99$: SEV
10397  062630  000207                    RTS     PC
10398  062632  013737  177744  001306  2$: MOV     @#MEMERR,@#$TMP2                ;SAVE ERROR REG
10399  062640  013737  177740  001310      MOV     @#LOADRS,@#$TMP3                ;SAVE ERROR ADR
10400  062646  013737  177742  001312      MOV     @#HIADRS,@#$TMP4
10401  062654  010237  001522              MOV     R2,@#VADR
10402  062660  010037  001302              MOV     R0,@#$TMP0
10403  062664  104005                      ERROR   5
10404  062666  012737  177777  177744      MOV     #-1,@#MEMERR                    ;CLEAR ERROR REG
10405  062674  000754                      BR      99$                             ;RETURN
10406
10407                                  ;;*****************************************************************
10408                                  .SBTTL  ROUTINE TO GET A MAP REGISTER
10409                                  ;*THIS ROUTINE TAKES AN 18 BIT RANDOM NUMBER, FINDS TWO
10410                                  ;*CONSECUTIVE MAP REGISTERS THAT ARE NOT IN USE, LOADS THE
10411                                  ;*REGISTERS WITH THE PHYSICAL ADDRESS MINUS THE RANDOM NUMBER
10412                                  ;*AND THE NUMBER + 4K, AND RETURNS A NEW BUS ADDRESS, BASED
10413                                  ;*ON THE RANDOM NUMBER.
10414                                  ;*
10415                                  ;*      MAP REGISTERS 0 AND 1 ARE NOT USED IF THE PROGRAM IS
10416                                  ;*      RUNNING ON ACT11. THIS ALLOWS 'MOTHER'' TO ACCESS THE
10417                                  ;*      END OF PASS HOOKS.
10418                                  ;*
10419                                  ;*      THE MAP TABLE (MAPTBL) CONTAINS 4 BYTES, ONE FOR EACH
10420                                  ;*      UNIBUS DEVICE. IF THE UBE IS PRESENT IT USES THE
10421                                  ;*      4TH BYTE. WHEN A REGISTER IS ASSIGNED TO A DEVICE,
10422                                  ;*      THE LOWER 4 ADDRESS BITS OF THAT REGISTER ARE PLACED
10423                                  ;*      IN THE TABLE. WHEN A DEVICE REQUESTS A REGISTER PAIR
10424                                  ;*      THIS TABLE IS THEN SEARCHED TO SEE IF THE REGISTER
10425                                  ;*      PAIR IS IN USE.
10426                                  ;*      ENTER WITH:
10427                                  ;*              4(SP)=DEVICE ID
10428                                  ;*              2(SP)=ADDRESS OF THE PHYSICAL ADDRESS
10429                                  ;;*****************************************************************
10430  062676  016600  000004  GETMAP: MOV     4(SP),R0                ;GET DIVICE ID
10431  062702  016601  000002          MOV     2(SP),R1                ;GET ADR OF PHY ADR
10432  062706  013746  177776          MOV     @#PSW,-(SP)             ;SAVE CURRENT PRIORITY
10433  062712  005116                  COM     (SP)                    ;MAKE IT READY FOR RESTORE
10434  062714  042716  177437          BIC     #^CPR7,(SP)             ;
10435  062720  000237                  SPL     7                       ;IF THIS IS RK CALL, LOCK OUT UBE
10436  062722  104420                  SAVREG
10437  062724  012137  001226          MOV     (R1)+,@#$GDDAT          ;SAVE PHYSICAL
10438  062730  012137  001230          MOV     (R1)+,@#$BDDAT          ;ADDRESS
10439  062734  004737  060436  2$:     JSR     PC,@#$RAND              ;GET RANDOM NUMBER
10440  062740  013702  001552          MOV     @#$HINUM,R2             ;GET HIGH RANDOM NUMBER
10441  062744  013703  001550          MOV     @#$LONUM,R3             ;GET LOW RANDOM NUMBER
10442  062750  073227  177764          ASHC    #-14,R2                 ;CONVERT TO 20 BIT NUMBER
10443  062754  042702  177760          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                  BPL     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     #MAPL0,R4               ;GENERATE MAP ADDRESS
10465  063056  042703  160000          BIC     #160000,R3              ;GET LS 13 BITS OF RAND NO.
10466  063062  013701  001226          MOV     @#$GDDAT,R1             ;GET PHYSICAL
10467  063066  013702  001230          MOV     @#$BDDAT,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                          ;;*****************************************************************
10496                          .SBTTL  ROUTINE TO CLEAR 'T' BIT
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                    RETPSW: .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 MODE
10510  063214  016746  177750                    MOV     RETPSW,-(SP)    ;PUSH ORIG PSW ONTO STACK
10511  063220  000766                            BR      RESPSW
10512
10513                                    ;;*****************************************************************
10514                                    .SBTTL  KEYBOARD INT SERV ROUTINE
10515                                    ;*THIS ROUTINE HANDLES INTERRUPTS FROM THE KEYBOARD
10516                                    ;*
10517                                    ;*TYPING A CONTROL 'C' WILL CAUSE THE PROCESSOR TO  HALT
10518                                    ;*
10519                                    ;*TYPING A CARRAGE RETURN WILL CAUSE A CARRIAGE RETURN-LINE FEED
10520                                    ;*TO BE TYPED.
10521                                    ;*
10522                                    ;*TYPING A CONTROL 'O' WILL INHIBIT ANY FURTHER TYPEOUT. THE SECOND CONTROL 'O'
10523                                    ;*WILL ENABLE TYPEOUT AGAIN AND ECHO A CR-LF.
10524                                    ;*
10525                                    ;*ANY OTHER CHARACTER WILL JUST BE ECHOED.
10526                                    ;;*****************************************************************
10527
10528          000003                            CNTRLC=3
10529          000017                            CNTRLO=17
10530
10531  063222  017746  116014           TKISR:  MOV     @$TKB,-(SP)     ;GET CHARACTER
10532  063226  042716  177600                    BIC     #177600,(SP)    ;STRIP UNUSED BITS
10533  063232  022716  000003                    CMP     #CNTRLC,(SP)    ;BRANCH IF NOT CONTROL C (^C)
10534  063236  001010                            BNE     1$
10535  063240  012737  001336  001270          MOV     #$CRLF-1,@#$REG5 ;ECHO CR LF
10536  063246  106277  115772                    ASRB    @$TPS
10537  063252  005726                            TST     (SP)+           ;POP CHARACTER OFF THE STACK
10538  063254  000000                            HALT
10539  063256  000002                            RTI                     ;RETURN
10540
10541  063260  122716  000015           1$:     CMPB    #15,(SP)        ;BRANCH IF NOT <CR>
10542  063264  001007                            BNE     2$
10543  063266  012737  001336  001270          MOV     #$CRLF-1,@#$REG5 ;ECHO CR LF
10544  063274  106277  115744                    ASRB    @$TPS
10545  063300  005726                            TST     (SP)+           ;POP CHARACTEROFF STACK
10546  063302  000002                            RTI                     ;RETURN
10547
10548  063304  122716  000017           2$:     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     #$CRLF-1,@#$REG5 ;ECHO CR LF
10554  063330  106277  115710                    ASRB    @$TPS
10555  063334  000002                    7$:     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    @$TPS                   ;CLEAR IE BIT
10573  063416  000002                          RTI                             ;RETURN
10574  063420  105777  115620          1$:     TSTB    @$TPS           ;PRINTER READY?
10575  063424  100375                          BPL     .-4                     ;BRANCH IF NO
10576  063426  112677  115614                  MOVB    (SP)+,@$TPB             ;MOVE CHAR TO PRINTER
10577  063432  000002                          RTI                             ;RETURN
10578
10579
10580                                  ;;*******************************************************************
10581                                  .SBTTL  TELETYPE INTERRUPT SERVICE ROUTINE
10582                                  ;*THIS ROUTINE TYPES A MESSAGE POINTED TO BY THE ADR STORED
10583                                  ;*IN LOCATION $REG5. THIS ROUTINE IS INTERRUPT DRIVEN.
10584                                  ;;*******************************************************************
10585  063434  005237  001270          TPISR:  INC     @#$REG5         ;STEP MESSAGE ADDRESS PTR
10586  063440  117746  115624                  MOVB    @$REG5,-(SP)            ;GET CHAR TO BE TYPED
10587  063444  001356                          BNE     ECHO                    ;GO TYPE CHAR IF NOT '0'
10588  063446  005726                          TST     (SP)+                   ;POP STACK
10589  063450  005077  115570                  CLR     @$TPS                   ;CLEAR IE BIT
10590  063454  012737  001554  001270          MOV     #NULLS,@#$REG5
10591  063462  000002                          RTI                             ;RETURN
10592
10593                                  ;;*******************************************************************
10594                                  .SBTTL  PARITY ERROR SERVICE
10595                                  ;*      THIS ROUTINE FIELDS UNEXPECTED TRAPS TO 114. IT IS ASSUMED
10596                                  ;*      THAT THE ERROR WAS IN CACHE AND WAS CAUSED BY THE ''OTHER
10597                                  ;*      WORD'' RATHER THAN THE ''WANTED WORD'' WHICH MEANS THAT THE
10598                                  ;*      BAD DATA IS STILL IN THE CACHE. SO, TO CLEAR THE BAD DATA
10599                                  ;*      THE ERROR ADDRESS IS REFERENCED CAUSING THE CACHE TO GO
10600                                  ;*      TO MAIN MEMORY TO GET THE DATA. THIS PREVENTS AN
10601                                  ;*      ARBITRARY REFERENCE TO THE BAD WORD FROM TRAPPING.
10602                                  ;*
10603                                  ;*      AFTER THE ERROR IS REPORTED, BITS 2 AND 3 OF THE MEMORY
10604                                  ;*      ERROR REGISTER ARE TESTED TO SEE IF THE BAD DATA IS IN
10605                                  ;*      MAIN MEMORY. IF IT IS, THE PROGRAM RESTARTS SINCE THE
10606                                  ;*      GOOD DATA IS NOW LOST FOREVER. OTHERWISE THE PROGRAM
10607                                  ;*      RETURNS TO THE ADDRESS POINTED TO BY ''$LPERR''.
10608                                  ;;*******************************************************************
10609  063464  012737  063744  000114  .PARSRV:MOV     #RT1,@#CACHVEC  ;PUT NEW ADDRESS IN PARITY VECTOR
10610  063472  016637  000002  001302          MOV     2(SP),@#$TMP0   ;SAVER ERROR PSW
10611  063500  011637  001522                  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,@#$TMP3      ;PUT LOW ADR IN MEMORY
10616  063526  013737  177742  001312          MOV     @#HIADRS,@#$TMP4        ;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,@#$TMP2        ;SAVE ERROR REG FOR TYPEOUT
10629  063606  013737  001212  001266          MOV     @#$LPERR,@#$REG4        ;SAVE LOOP ADDRESS
10630  063614  012737  063624  001212          MOV     #2$,@#$LPERR            ;SET RETURN ADDRESS IF LOOPING
10631  063622  104004                          ERROR   4
10632  063624  013737  001266  001212  2$:     MOV     @#$REG4,@#$LPERR        ;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$:  .ASCIZ  /FATAL PARITY ERROR-RESTARTING/<CRLF>
10642  063732                          64$:
10643  063732  000005                          RESET                   ;CLEAR THE WORLD
10644  063734  000137  003542                  JMP     @#START
10645  063740  012716  063746          1$:     MOV     #X,(SP)         ;PUT ADDRESS ON STACK TO GET ORIGINAL
10646                                                                  ;PSW BACK
10647  063744  000002                  RT1:    RTI                     ;GET OLD PSW
10648  063746  000177  115240          X:      JMP     @$LPERR         ;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,@#$SAVPSW ;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  105737  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     #$SAVPAR,R0     ;GET ADDRESS OF SAVE BUFF
10685  064064  012001                          MOV     (R0)+,R1        ;GET OLD PAR DATA
10686  064066  012002                          MOV     (R0)+,R2
10687  064070  012003                          MOV     (R0)+,R3
10688  064072  012005                          MOV     (R0)+,R5
10689  064074  012700  172340                  MOV     #KIPAR0,R0      ;GET ADDRESS OF PAR0
10690  064100  010120                          MOV     R1,(R0)+        ;RELOCATE BACK
10691  064102  010220                          MOV     R2,(R0)+
10692  064104  010320                          MOV     R3,(R0)+
10693  064106  010510                          MOV     R5,(R0)
10694  064110  013737  001516  177776          MOV     @#$SAVPSW,@#PSW
10695  064116  010446                          MOV     R4,-(SP)
10696  064120  000207                  1$:     RTS     PC
10697                                   ;;*****************************************************************
10698                                   .SBTTL  KT ABORT SUBROUTINE
10699                                   ;;*****************************************************************
10700  064122  016637  000002  001302  KTABRT: MOV     2(SP),@#$TMP0           ;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,@#$TMP2          ;SAVE MMR0
10704  064150  013737  177576  001310          MOV     @#MMR2,@#$TMP3          ;SAVE MMR2
10705  064156  013737  001212  001266          MOV     @#$LPERR,@#$REG4        ;SAVE LOOP ADDRESS
10706  064164  012737  064174  001212          MOV     #1$,@#$LPERR            ;SET RETURN ADR IF LOOPING
10707  064172  104003                          ERROR   3
10708  064174  013737  001266  001212  1$:     MOV     @#$REG4,@#$LPERR        ;RESTORE LOOP ADR
10709  064202  042737  170000  177572          BIC     #170000,@#MMR0          ;CLEAR ERRORS
10710  064210  013716  001212                  MOV     @#$LPERR,(SP)           ;GET LOOP ADDRESS
10711  064214  000002                          RTI                             ;RETURN
10712
10713                                   ;;*****************************************************************
10714                                   .SBTTL  RESERVED INSTRUCTION ROUTINE
10715                                   ;;*****************************************************************
10716  064216  016637  000002  001302  RESERR: MOV     2(SP),@#$TMP0           ;SAVE PSW
10717  064224  011637  001522                  MOV     (SP),@#VADR             ;SAVE ERROR PC
10718  064230  162737  000002  001522          SUB     #2,@#VADR
10719  064236  013737  001212  001266          MOV     @#$LPERR,@#$REG4        ;SAVE LOOP ADR
10720  064244  012737  064254  001212          MOV     #1$,@#$LPERR            ;SET RETURN ADR IF LOOPING
10721  064252  104002                          ERROR   2
10722  064254  013737  001266  001212  1$:     MOV     @#$REG4,@#$LPERR        ;RESTORE LOOP ADR
10723  064262  013716  001212                  MOV     @#$LPERR,(SP)           ;GET LOOP ADDRESS
```

```
10724  064266  000002                          RTI                                 ;RETURN
10725
10726                                          ;;*****************************************************************
10727                                          .SBTTL  TRAP TO 4 SERVICE ROUTINE
10728                                          ;;*****************************************************************
10729  064270  016637  000002  001302  ERPRT:  MOV     2(SP),@#$TMP0               ;SAVE ERROR PSW
10730  064276  011637  001522                  MOV     (SP),@#VADR                 ;SAVE ERROR PC
10731  064302  162737  000002  001522          SUB     #2,@#VADR
10732  064310  012706  000700                  MOV     #SUPSTK,SP                  ;RESTORE SP
10733  064314  013737  177766  001306          MOV     @#CPUERR,@#$TMP2            ;GET ERROR REG
10734  064322  013737  001212  001266          MOV     @#$LPERR,@#$REG4            ;SAVE LOOP ADR
10735  064330  012737  064340  001212          MOV     #1$,@#$LPERR                ;SET RETURN ADR IF LOOPING
10736  064336  104001                          ERROR   1
10737  064340  013737  001266  001212  1$:     MOV     @#$REG4,@#$LPERR            ;SET LOOP ADR
10738  064346  005037  177766                  CLR     @#CPUERR
10739  064352  013746  001302                  MOV     @#$TMP0,-(SP)    ;SETUP STACK TO RETURN
10740  064356  013746  001212                  MOV     @#$LPERR,-(SP)
10741  064362  000002                          RTI                                 ;RETURN
10742
```

```
10743                                   ;THE BELOW TABLE REPRESENTS THE 'NEW' PSW SET BY THE PROGRAM ON
10744                                   ;SUCCESSIVE SUB-PASSES.
10745                                   ;NOTE THE BELOW TABLE MAY BE MODIFIED TO CAUSE THE PROGRAM TO RUN
10746                                   ;UNDER USER DEFINED PARAMETERS BY PATCHING IN THE DESIRED PASS PARAMETER
10747                                   ;FOR EXAMPLE TO CAUSE THE PROGRAM TO RUN WITHOUT SETTING THE 'T' BIT
10748                                   ;IN ALL PASSES PATCH OUT THE 'T' BIT IN THE TABLE.
10749  064364  000000                   PSWTAB: 000000                  ;
10750  064366  000020                           000020                  ;T-BIT TRAPPING
10751  064370  140000                           140000                  ;USER MODE
10752  064372  144020                           144020                  ;USER MODE, REG SET #1, T-BIT TRAPPING
10753  064374  040000                           040000                  ;SUPERVISOR MODE
10754  064376  044020                           044020                  ;SUPERVISOR MODE, REG SET #1, T-BIT TRAPPING
10755
10756                                   ;THE BELOW TABLE IS USED TO SET MEMORY MARGINS
10757  064400  000000                   MRGTAB: .WORD   0                       ;NO MARGINS
10758  064402  000004                           .WORD   4                       ;EARLY STROBE
10759  064404  000006                           .WORD   6                       ;LATE STROBE
10760  064406  000010                           .WORD   10                      ;LOW DRIVE CURRENT
10761  064410  000000                           .WORD   0                       ;NO MARGINS
10762  064412  000012                           .WORD   12                      ;HIGH DRIVE CURRENT
10763                                   ;MESSAGES
10764                                           .EVEN
10765  064414  002140                   REGINX: RP3DS
10766  064416  002162                           RKDS
10767  064420  000000                           .WORD
10768  064422  000000                           .WORD
10769  064424  002202                           RP4CS1
10770  064426  002242                           RSCS1
10771  064430  000000                           .WORD
10772  064432  000000                           .WORD
10773  064434  002306                           MBTTBL
10774  064436  002270                           UBETBL
10775  064440  064577                   MSGINX: .WORD   MSG5
10776  064442  064605                           .WORD   MSG6
10777  064444  065414                           .WORD   MSG21
10778  064446  065414                           .WORD   MSG21
10779  064450  064613                           .WORD   MSG10
10780  064452  064621                           .WORD   MSG11
10781  064454  065414                           .WORD   MSG21
10782  064456  065414                           .WORD   MSG21
10783  064460  065110                           .WORD   MSG15
10784  064462  065453                           .WORD   MSG24
10785  064464  046200  053517  046040   MSG1:   .ASCIZ  <CRLF>'LOW LIM?'
10786  064472  046511  000077
10787  064476  044510  044107  046040   MSG2:   .ASCIZ  'HIGH LIM?'
10788  064504  046511  000077
10789  064510  051105  047522  050122   MSG3:   .ASCIZ  /ERRORPC PHYSC PC    PSW   MAINT   TEST NO SUB-PASS CNT/
10790  064516  020103  044120  051531
10791  064524  020103  041520  020040
10792  064532  020040  051520  020127
10793  064540  020040  040515  047111
10794  064546  020124  020040  042524
10795  064554  052123  047040  020117
10796  064562  052523  026502  040520
10797  064570  051523  041440  052116
10798  064576     000
```

```
10799                                   ;MSG4 HAS BEEN MOVED TO END OF PROGRAM
10800
10801  064577     122  030120  004463  MSG5:   .ASCIZ  ?RP03   ?
10802  064604     000
10803  064605     122  030113  004465  MSG6:   .ASCIZ  ?RK05   ?
10804  064612     000
10805  064613     122  030120  004464  MSG10:  .ASCIZ  ?RP04   ?
10806  064620     000
10807  064621     122  030123  004464  MSG11:  .ASCIZ  ?RS04   ?
10808  064626     000
10809  064627     104  053122  052123  MSG12:  .ASCIZ  /DRVSTA  ERRREG  CSREG   WRDCNT  BUSADR  DSKADR  CYLADR(RP03)  PHYS BUSA
10810  064634  020101  042440  051122
10811  064642  042522  020107  041440
10812  064650  051123  043505  020040
10813  064656  053440  042122  047103
10814  064664  020124  041040  051525
10815  064672  042101  020122  042040
10816  064700  045523  042101  020122
10817  064706  041440  046131  042101
10818  064714  024122  050122  031460
10819  064722  020051  050040  054510
10820  064730  020123  052502  040523
10821  064736  051104  000200
10822  064742  041440  030523  020040  MSG13:  .ASCIZ  / CS1    WRDCNT  BUSADR  BADREX  DSKADR   CS2     CS3    DRVSTA  ERRREG/
10823  064750  020040  051127  041504
10824  064756  052116  020040  052502
10825  064764  040523  051104  020040
10826  064772  040502  051104  054105
10827  065000  020040  051504  040513
10828  065006  051104  020040  041440
10829  065014  031123  020040  020040
10830  065022  041440  031523  020040
10831  065030  020040  051104  051526
10832  065036  040524  020040  051105
10833  065044  051122  043505  000200
10834  065052  042504  041523  046131  MSG14:  .ASCIZ  /DESCYL   ER2     ER3    RPCC/<CRLF>
10835  065060  020040  042440  031122
10836  065066  020040  020040  042440
10837  065074  031522  020040  020040
10838  065102  050122  041503  000200
10839  065110  040515  051523  041040  MSG15:  .ASCIZ  /MASS BUS TESTER /
10840  065116  051525  052040  051505
10841  065124  042524  020122     000
10842  065131     040  051503  020061  MSG16:  .ASCIZ  / CS1    WRDCNT  BUSADR  BADREX    MR2     CS2      ST     ER      CS3/<
10843  065136  020040  053440  042122
10844  065144  047103  020124  041040
10845  065152  051525  042101  020122
10846  065160  041040  042101  042522
10847  065166  020130  020040  046440
10848  065174  031122  020040  020040
10849  065202  041440  031123  020040
10850  065210  020040  020040  052123
10851  065216  020040  020040  042440
10852  065224  020122  020040  020040
10853  065232  041440  031523  000200
10854  065240  020040  041503  020040  MSG17:  .ASCIZ  /  CC    BUSADR    CR2     CR1  PHYS BUSADR/<CRLF>
```

```
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  MSG20:  .ASCIZ  /THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789/<15><12>
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  MSG21:  .ASCIZ  /ILLEGAL DEVICE/<CRLF>
10874  065422  020114  042504  044526
10875  065430  042503  000200
10876  065434  020040  020040  020040  MSG22:  .ASCII  /        /
10877  065442  020040
10878  065444  020040  020040  020040  MSG23:  .ASCIZ  /        /
10879  065452     000
10880  065453     125  044516  052502  MSG24:  .ASCIZ  /UNIBUS EXERCISER /
10881  065460  020123  054105  051105
10882  065466  044503  042523  020122
10883  065474     000
10884  065475     117  052120  041456  MSG25:  .ASCIZ  /OPT.CP=/
10885  065502  036520     000
10886
10887                                  ;MSG30 THROUGH MSG38 ARE AT END OF LISTING
10888
10889  065505     125  042516  050130  EM1:    .ASCIZ  /UNEXPECTED TRAP TO 4/
10890  065512  041505  042524  020104
10891  065520  051124  050101  052040
10892  065526  020117  000064
10893  065532  041520  043117  050124  DH1:    .ASCIZ  /PCOFTP  PHYSPC    PSW   CPUERR/
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  DF1:    .BYTE   0,1,0,0,0
10900  065574     000     000
10901                                  .EVEN
10902  065576  001522  001522  001302  DT1:    .WORD   VADR,VADR,$TMP0,$TMP2,0
10903  065604  001306  000000
10904  065610  047125  054105  042520  EM2:    .ASCIZ  /UNEXPECTED TRAP TO 10/
10905  065616  052103  042105  052040
10906  065624  040522  020120  047524
10907  065632  030440  000060
10908  065636  041520  043117  050124  DH2:    .ASCIZ  /PCOFTP  PHYSPC    PSW/
10909  065644  020040  044120  051531
10910  065652  041520  020040  020040
```

```
10911  065660  051520  000127
10912                                          .EVEN
10913  065664  001522  001522  001302  DT2:    .WORD   VADR,VADR,$TMP0,0
10914  065672  000000
10915  065674  047125  054105  042520  EM3:    .ASCIZ  /UNEXPECTED TRAP TO 250(MGMT)/
10916  065702  052103  042105  052040
10917  065710  040522  020120  047524
10918  065716  031040  030065  046450
10919  065724  046507  024524     000
10920  065731     120  047503  052106  DH3:    .ASCIZ  /PCOFTP  PHYSPC    PSW    MMR0    MMR2/
10921  065736  020120  050040  054510
10922  065744  050123  020103  020040
10923  065752  050040  053523  020040
10924  065760  020040  046515  030122
10925  065766  020040  020040  046515
10926  065774  031122     000
10927          066000                          .EVEN
10928  066000  001522  001522  001302  DT3:    .WORD   VADR,VADR,$TMP0,$TMP2,$TMP3,0
10929  066006  001306  001310  000000
10930  066014  047125  054105  042520  EM4:    .ASCIZ  /UNEXPECTED TRAP TO 114/
10931  066022  052103  042105  052040
10932  066030  040522  020120  047524
10933  066036  030440  032061     000
10934  066043     120  047503  052106  DH4:    .ASCIZ  /PCOFTP  PHYSC PC    PSW   ERRREG  ERR ADR REG/
10935  066050  020120  050040  054510
10936  066056  041523  050040  020103
10937  066064  020040  050040  053523
10938  066072  020040  042440  051122
10939  066100  042522  020107  042440
10940  066106  051122  040440  051104
10941  066114  051040  043505     000
10942  066121     000     001     000  DF4:    .BYTE   0,1,0,0,2
10943  066124     000     002
10944  066126  040520  044522  054524  EM5:    .ASCIZ  /PARITY ERROR DURING DATA CHECK/
10945  066134  042440  051122  051117
10946  066142  042040  051125  047111
10947  066150  020107  040504  040524
10948  066156  041440  042510  045503
10949  066164     000
10950  066165     123  041522  042101  DH5:    .ASCIZ  /SRCADR  DSTADR  EADRREG  MEM ERR REG/
10951  066172  020122  042040  052123
10952  066200  042101  020122  042440
10953  066206  042101  051122  043505
10954  066214  020040  042515  020115
10955  066222  051105  020122  042522
10956  066230  000107
10957  066232     000     001     002  DF5:    .BYTE   0,1,2,0
10958  066235     000
10959                                          .EVEN
10960  066236  001302  001522  001310  DT5:    .WORD   $TMP0,VADR,$TMP3,$TMP2,0
10961  066244  001306  000000
10962  066250  051105  047522  020122  EM6:    .ASCIZ  /ERROR DURING DATA CHECK-RELOC WAS BY CP/
10963  066256  052504  044522  043516
10964  066264  042040  052101  020101
10965  066272  044103  041505  026513
10966  066300  042522  047514  020103
```

```
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                  .EVEN
10973  066340  001302  001522  000000  DT6:    .WORD   $TMP0,VADR,0
10974  066346  051105  047522  020122  EM10:   .ASCIZ  ?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                                  .EVEN
10991  066472  001302  001522  001306  DT10:   .WORD   $TMP0,VADR,$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                                  .EVEN
11005  066574  001302  001304  000000  DT11:   .WORD   $TMP0,$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                  .EVEN
11015  066656  001226  000000  DT12:   .WORD   $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                                  .EVEN
11032  067002  001312  001262  001316  DT14:   .WORD   $TMP4,$REG2,$TMP6,$REG3,0
11033  067010  001264  000000
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                  .EVEN
11045  067066  001432  001262  001442  DT16:   .WORD   FLTMP0,$REG2,FLTMP1,$REG3,0
11046  067074  001264  000000
11047  067100  030122  043040  044501  EM17:   .ASCIZ  /R0 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                  ENDTAG: .WORD   0
11058                                  ;;*****************************************************************
11059                                  ;THE FOLLOWING ASCII GETS OVERLAYED WHEN THE PROGRAM RUNS.
11060  067176  050117  051105  052101  SWITCH: .ASCII  /OPERATIONAL SWITCH SETTINGS/<CRLF>
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
```

```
11079  067340  020122  054524  042520
11080  067346  052517  051524     200
11081  067353     040  030440  004462          .ASCII  /  12          INHIBIT UBE/<CRLF>
11082  067360  044411  044116  041111
11083  067366  052111  052440  042502
11084  067374     200
11085  067375     040  030440  004461          .ASCII  /  11          INHIBIT ITTERATIONS/<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          .ASCII  /  2          THESE THREE SWITCHES/<CRLF>
11139  070032  052011  042510  042523
11140  070040  052040  051110  042505
11141  070046  051440  044527  041524
11142  070054  042510  100123
11143  070060  020040  030440  004411          .ASCII  /  1          ARE ENCODED TO SELECT RELOCATION/<CRLF>
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          .ASCII  /  0          ON THE FOLLOWING DEVICES:/<CRLF>
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          .ASCII  ?       0...RP11/RP03?<CRLF>
11157  070174  050122  030461  051057
11158  070202  030120  100063
11159  070206  030411  027056  051056          .ASCII  ?       1...RK11/RK05?<CRLF>
11160  070214  030513  027461  045522
11161  070222  032460     200
11162  070225     011  027062  027056          .ASCII  ?       2...NOT USED?<CRLF>
11163  070232  047516  020124  051525
11164  070240  042105     200
11165  070243     011  027063  027056          .ASCII  ?       3...NOT USED?<CRLF>
11166  070250  047516  020124  051525
11167  070256  042105     200
11168  070261     011  027064  027056          .ASCII  ?       4...RH70/RP04?<CRLF>
11169  070266  044122  030067  051057
11170  070274  030120  100064
11171  070300  032411  027056  051056          .ASCII  ?       5...RH70/RS04 OR RS03?<CRLF>
11172  070306  033510  027460  051522
11173  070314  032060  047440  020122
11174  070322  051522  031460     200
11175  070327     011  027066  027056          .ASCII  ?       6...NOT USED?<CRLF>
11176  070334  047516  020124  051525
11177  070342  042105     200
11178  070345     011  027067  027056          .ASCII  ?       7...NOT USED?<CRLF>
11179  070352  047516  020124  051525
11180  070360  042105  000200
11181  070364  052200  042510  043040  MSG4:   .ASCII  <CRLF>/THE FOLLOWING DEVICES AND DRIVES WILL BE USED FOR RELOCATION IF B
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
```

```
11191  070460  020116  043111  041040
11192  070466  052111  034040  051440
11193  070474  052105  100072
11194  070500  042504  044526  042503          .ASCIZ  /DEVICE DRIVES/<CRLF>
11195  070506  042011  044522  042526
11196  070514  100123    000
11197  070517     015  025012  047052  MSG30:  .ASCII  <15><12>/**NOTE** SWITCH REG BIT 8 HAS BEEN REVERSED IN REV D/<CRLF>
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          .ASCII  'NOTE THAT SWR BIT 8 SET NOW ALLOWS I/O RELOCATION'<CRLF><CRLF>
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          .ASCII  ''THIS PROGRAM SUPPORTS I/O RELOCATION ONLY WITH THE FOLLOWING DEVICES:''
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            .ASCIZ <CRLF>'RP03,RK05,RP04/5/6,RS03/4''
11229  071004  045522  032460  051054
11230  071012  030120  027464  027465
11231  071020  026066  051522  031460
11232  071026  032057    000
11233  071031     113  030502  026461  MSG31:  .ASCIZ  'KB11-EM''<15><12>
11234  071036  046505  005015     000
11235  071043     061  027461  032067  MSG32:  .ASCIZ  ''11/74          (KB11CM)''<15><12>
11236  071050  020040  020040  020040
11237  071056  020040  020040  045450
11238  071064  030502  041461  024515
11239  071072  005015     000
11240  071075     015  041412  052520  MSG34:  .ASCIZ  <15><12>''CPU UNDER TEST FOUND TO BE A ''
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  MSG35:  .ASCIZ  'KB11-B/C''<15><12>
```

```
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/<15><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#
ADC0       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
ADDPI  =   076170           587#
ADD0       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#
ARBFIN     043020          7425    7447    7468    7470    7481#
ASHCL0     030402          5582#
ASHCR0     030460          5603#
ASHL0      030172          5532#
ASHL1      031004          5716#
ASHN   =   076056           553#   7074
ASHNI  =   076156           585#   7039    7043    7051    7105
ASHP   =   076076           568#   7172    7183
ASHPI  =   076176           593#   7155
ASHR0      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#
ASL0       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#
ASR0       011662          2076    2077    2078    2080#
```

ASR1	012470	2269	2270	2271	2273#								
ASR2	013466	2559	2560	2562#									
ASR3	014210	2746	2748#										
ASR6	014662	2906	2907	2909#									
ASR7	015714	3157	3158	3160#									
A.DSC	041206	7022	7033	7034	7040	7061	7100	7106	7116	7137	7138	7319#	
A1	037762	7033#	7364										
A2	037764	7034#	7364										
A3	040000	7040#	7364										
A4	040050	7061#	7364										
A5	040170	7100#	7364										
A6	040204	7106#	7364										
A7	040364	7137#	7364										
A8	040366	7138#	7364										
B	041214	7126*	7322#	7391									
BICB1	017440	3601	3602	3604#									
BICB1A	017462	3612	3615#	3674	3675								
BIC0	016606	3378	3379	3380	3382#								
BIC1	017130	3491	3492	3494#									
BIC2	017776	3713	3714	3715	3717#								
BIC3	020512	3883	3885#										
BIC7	022452	4281#	4283										
BINB	020230	3778	3780	3783	3785	3788	3790	3793	3796#				
BINB7	022220	4216	4224#										
BIN1	017620	3644	3647	3650	3653	3656	3659	3662	3667#				
BISB1	017426	3596	3598#										
BIS0	016564	3369	3370	3372#									
BIS0A	016642	3396	3398#										
BIS1	017116	3485	3486	3488#									
BIS2	017734	3697	3699#										
BIS2A	020036	3731	3732	3734#									
BIS7	022412	4271#	4273										
BITB1	017416	3590	3591	3593#									
BITB2	020314	3818	3819	3821#									
BITB3	020676	3938	3941#										
BITB6	021312	4031#											
BITT1	017044	3462	3463	3465#									
BITT2	020022	3724	3725	3727#									
BIT0 =	000001	179#	1484	1485	1492	1734	1742	1744	8364	8402	8443	10223	10455
BIT00 =	000001	169#	179										
BIT01 =	000002	168#	178										
BIT02 =	000004	167#	177										
BIT03 =	000010	166#	176										
BIT04 =	000020	165#	175										
BIT05 =	000040	164#	174										
BIT06 =	000100	163#	173										
BIT07 =	000200	162#	172	1257									
BIT08 =	000400	161#	171										
BIT09 =	001000	160#	170	8972	9039								
BIT1 =	000002	178#											
BIT10 =	002000	159#	9022										
BIT11 =	004000	158#	514	1487	1488	1490	8872	8979					
BIT12 =	010000	157#	513	1683	1718	8277							
BIT13 =	020000	156#	8257	9029									
BIT14 =	040000	155#	512	1163	1479	1480	1482	8530	8602	8625	8640	8651	8670	8701
		8719	8734	8746	8765	8865	9962						

```
BIT15 = 100000    154#  1493  1494  1496  1498  1499  1501  1543  1552  7885  8374  8381  8493
                 8500  8620  8627  8714  8721  9913 10226 10453
BIT2  = 000004    177#
BIT3  = 000010    176#  8409
BIT4  = 000020    175#  1778  4827  5442  6087  7966  8181  8185
BIT5  = 000040    174#  1704  1726  6258  7504  7893  7895  8115  8196  8262  8414  8450  8535
                 8571  8606  8645  8675  8705  8739  8770 10307
BIT6  = 000100    173#  8208  8937
BIT7  = 000200    172#  1629
BIT8  = 000400    171#  1514  1635  1672  1702  1724  7899  8074  8151  8224  8291  8358  8393
                 8436  8476  8513  8555  8600  8611  8638  8668  8699  8732  8763
BIT9  = 001000    170#  1686  7909
BPTVEC= 000014    186#  4739  4742* 4743* 4816* 4817*
BUFF    041706   7314  7316  7318  7320  7322  7324  7326  7328  7330  7340  7404#
BUFFAD  041264   6822  6823  7002  7003  7128  7213  7215  7340# 7391
B.DSC   041212   7044  7066  7143  7144  7152  7157  7175  7321#
B1      040010   7044#  7367
B2      040062   7066#  7367
B3      040376   7143#  7367
B4      040400   7144#  7367
B5      040444   7162#  7367
B6      040474   7175#  7367
B7      040432   7157#  7367
C       041220   7127*  7324#  7391
CACHVE= 000114    193#  1791* 1793* 7601  7602* 7603* 7610* 7611* 7984* 7985* 10378* 10394* 10609*
                10636*
CALLHA= 000010    515#  7793  7901
CBIT    026122   5041#
CC0     011312   1939  1940  1941  1942  1943  1944  1945  1947#
CC1     011326   1951  1952  1953  1955#
CC2     011342   1959  1960  1961  1963#
CC3     011354   1967  1968  1970#
CC4     011370   1974  1975  1976  1978#
CHAR    041254   6846  6893  6898  6914  6995  7336#
CHAR1   037054   6846#  7362
CHAR2   037242   6893#  7362
CHAR3   037254   6898#  7362
CHAR4   037324   6914#  7362
CHAR5   037620   6995#  7362
CHKDAT  062534   7660  7840  7932 10377#
CHKSP   025706   4974#
CISER   040642   6839  6853  6875  6901  6917  6936  6955  6976  6998  7189  7222#
CISER1  040660   7096  7204  7225#
CISER2  040676   7081  7113  7228#
CISER3  040714   7036  7231#
CISOP   005712   1537* 1539  1540* 1542  1544* 1546  1555#
CISOPT= 010000    506#  1533  1551
CISP    001505    724#  1536* 1548  1550* 1559* 1590  4839  6784  6796
CISTRP  005730   1557  1560#
CISTST= 006600    527#  1545
CLRTBI  063162   4733  4927  4967  7547  7987 10498#
CLR0    011430   1989  1990  1991  1992  1994#
CMPB1   017372   3580  3582#
CMPB2   020300   3811  3812  3814#
CMPB3   020710   3944  3947#
CMPC  = 076044    545#  6899  6915  6934  6996
```

```
CMPCI = 076144          577#   6834   6848   6870   6950   6971
CMPN  = 076052          549#   7079
CMPNI = 076152          581#   7032   7109
CMPNUM  026156         5055#
CMPP  = 076072          564#
CMPPAK  040362         7136#
CMPPI = 076172          589#   7136   7185
CMPO    016466         3332   3333   3334   3336#
CMPOA   016730         3422   3425#
CMP1    017026         3453   3454   3455   3457#
CMP1A   017146         3498   3499   3500   3502#
CMP2    017724         3691   3692   3694#
CMP7    021520         4085   4086   4088#
CNTRLC= 000003        10528#  10533
CNTRLO= 000017        10529#  10548
CNVADR  062432         7664   9091   9171  10341#
COMB1   013062         2411   2412   2414#
COMB1A  013242         2477   2478   2480#
COMB2   013640         2615   2616   2618#
COMB5   014424         2823   2825#
COMB6   015266         3027   3028   3030#
COMB7   016206         3246   3247   3249#
COMP    037016         6833#
COMO    011474         2009   2010   2011   2012   2014#
COM1    012614         2319   2320   2322#
COM3    014266         2773   2775#
COM4    013354         2516   2518#
COM6    014646         2901   2902   2904#
COM7    016004         3185   3186   3188#
CONNUM  040124         7080   7086#
CONPAK  040534         7188   7194#
CONTRL= 177746          203#   1476   1506*  1507
CPCHK   004672         1370#
CPUERR= 177766          216#   4828*  5974*  6088*  10733  10738*
CR    = 000015           91#   9393   9403
CRLF  = 000200           92#   1142   1146   1188   1573   1763   7739   7748   9226   9364   9403  10642  10785
                       10809  10822  10834  10842  10854  10873  11060  11065  11068  11072  11076  11081  11085
                       11090  11094  11098  11107  11116  11121  11128  11134  11138  11143  11150  11156  11159
                       11162  11165  11168  11171  11175  11178  11181  11194  11197  11207  11228
CVTLN = 076057          554#
CVTLNI= 076157          586#   7086
CVTLP = 076077          569#
CVTLPI= 076177          594#   7194
CVTNL = 076053          550#
CVTNLI= 076153          582#   7089
CVTNP = 076055          552#
CVTNPI= 076155          584#   7099
CVTPL = 076073          565#
CVTPLI= 076173          590#   7197
CVTPN = 076054          551#
CVTPNI= 076154          583#   7102
C.DSC   041216         7052   7062   7147   7148   7151   7156   7176   7323#
C1      040030         7052#  7370
C2      040052         7062#  7370
C3      040406         7147#  7370
C4      040410         7148#  7370
```

```
C5        040442          7161#   7370
C6        040476          7176#   7370
C7        040430          7156#   7370
D         041234          7330#   7391
DBINB7    022214          4221#   4253*   4254*   4258*   4259*   4260    4264*   4271    4272*   4279*   4281    4282*
DBIN7     021420          4060#   4075*   4079*   4084    4090*   4098*   4105*   4109*   4113
DDATA     020734          3956#   3960*   3962*   3966    3970    3971    3974*   3975    3980*   3987*   3994*   3995    3998*
                          4001    4005
DDATAB    021376          4027*   4028*   4033*   4036    4040*   4041    4045    4050#
DECB1     013032          2397    2398    2400#
DECB1A    013162          2449    2450    2452#
DECB2     013774          2668    2669    2671#
DECB5     014564          2879    2881#
DECB6A    015366          3058    3059    3060    3062#
DECB7     016254          3265    3266    3268#
DECDAT    037634          6997    7002#
DECO      011554          2035    2036    2037    2038    2040#
DEC1      012406          2242    2244#
DEC1A     012656          2340    2342#
DEC2      013522          2576    2577    2579#
DEC5      014242          2759    2760    2761    2763#
DEC6      015014          2950    2951    2953#
DEC7      015676          3150    3151    3153#
DEVICE    001576           752#
DEVIND    001600           753#   7696*   7697*   7698*   7702    7704*   7705*   7706
DF1       065571          1009    1014    1019    10899#
DF10      066466          1044    10988#
DF11      066572          1049    1079    1085    11003#
DF12      066654          1054    1059    11013#
DF14      067014          1064    11034#
DF16      067061          1074    11042#
DF4       066121          1024    1034    10942#
DF5       066232          1029    10957#
DH1       065532          1007    10893#
DH10      066417          1042    10981#
DH11      066551          1047    1077    11000#
DH12      066640          1052    11011#
DH13      066720          1057    11021#
DH14      066762          1062    11028#
DH16      067034          1072    11038#
DH2       065636          1012    10908#
DH3       065731          1017    10920#
DH4       066043          1022    10934#
DH5       066165          1027    10950#
DH6       066320          1032    1083    10969#
DISPLA=   177570            86#   1803*   1921*   1984*   2101*   2224*   2348*   2499*   2602*   2720*   2805*   2887*   2972*
                          3096*   3217*   3307*   3431*   3551*   3673*   3761*   3832*   3896*   3953*   4015*   4058*   4122*
                          4162*   4215*   4344*   4436*   4541*   4608*   4664*   4714*   4834*   4922*   4973*   5040*   5090*
                          5263*   5353*   5404*   5449*   5516*   5634*   5686*   5770*   5794*   5838*   5891*   5914*   5959*
                          5980*   6003*   6097*   6137*   6197*   6257*   6326*   6514*   6767*   6805*   7419*   7490*   9018*
DIVP    =  076075           567#
DIVPI   =  076175           592#   7168
DIVO      030720          5687#
DIV1      031174          5771#   5784
DOCIS     036650          6797    6799#
DOMFPT    036464          6751    6753#
```

DONE	040576	7203	7206#											
DONE7	036636	6792	6796#											
DST.1A	041200	6829	6967	7215*	7216*	7223	7316#	7390						
DST.1D	041176	6836	6845	6850	6868	6872	6882	6892	6897	6923	6926	6927	6932	6942
		6946	6947	6952	6968	6973	6990	6994	7214*	7315#				
DST1	037026	6836#	7352											
DST10	037350	6923#	7352											
DST11	037356	6926#	7356											
DST12	037360	6927#	7356											
DST13	037372	6932#	7356											
DST14	037420	6942#	7356											
DST15	037430	6946#	7356											
DST16	037432	6947#	7356											
DST17	037444	6952#	7356											
DST18	037534	6973#	7356											
DST19	037606	6990#	7356											
DST2	037052	6845#	7352											
DST20	037616	6994#	7356											
DST3	037064	6850#	7352											
DST4	037144	6868#	7352											
DST5	037154	6872#	7352											
DST6	037206	6882#	7352											
DST7	037240	6892#	7352											
DST8	037252	6897#	7352											
DST9	037322	6913#	7352											
DT1	065576	1008	10902#											
DT10	066472	1043	10991#											
DT11	066574	1048	1078	1084	11005#									
DT12	066656	1053	1058	11015#										
DT14	067002	1063	11032#											
DT16	067066	1073	11045#											
DT2	065664	1013	10913#											
DT3	066000	1018	1023	10928#										
DT5	066236	1028	10960#											
DT6	066340	1033	10973#											
DUMMY	042640	7433#												
D.DSC	041232	7024	7041	7048	7053	7056	7068	7071	7118	7145	7162	7165	7171	7177
		7180	7329#											
D1	040002	7041#	7373											
D10	040506	7180#	7373											
D2	040020	7048#	7373											
D3	040032	7053#	7373											
D4	040042	7057#	7373											
D5	040066	7068#	7373											
D6	040074	7071#	7373											
D7	040466	7171#	7373											
D8	040452	7165#	7373											
D9	040500	7177#	7373											
E	041224	7228	7326#	7391										
ECHO	063402	10569#	10587											
EISOPT=	040000	504#												
EMTVEC=	000030	189#	1107*	1347*	1348*	4615*	4617*	4654*	4655*					
EMT1	024050	4614	4622#											
EMT1B	024130	4622	4626	4627	4630	4631	4635	4636	4639	4640	4641	4646#		
EMT1C	024134	4620	4648#											
EMT1D	024146	4649	4650	4653#										

```
EM1     065505  1006  10889#
EM10    066346  1041  10974#
EM11    066504  1046  10993#
EM12    066602  1051  11006#
EM13    066662  1056  11016#
EM14    066735  1061  1071  11024#
EM15    067020  1066  11036#
EM17    067100  1076  11047#
EM2     065610  1011  10904#
EM20    067144  1082  11053#
EM3     065674  1016  10915#
EM4     066014  1021  10930#
EM5     066126  1026  10944#
EM6     066250  1031  10962#
END     046162  7982#
ENDCIS  042544  6752  7413#
ENDCP   032312  5916  5989#
ENDKB   005560  1520  1525#
ENDM    046122  7537  7965#
ENDMEM  046120  7600  7603  7964#
ENDTAG  067174  1251  11057#
END1    046162  7983#
ENTER2  044240  7655  7688#  7740  7882
ERPRT   064270  1453  1776  4337  4824  5971  6084  7609  7983  10729#
ERRBA   001732  778#  8810*  8811*  8812*  9285*  9288*  9289*  9290*  9302*  9303*  9304*  9305*  9306*
                10304  10305
ERRRTN  001342  704#
ERRVEC= 000004  182#  1161*  1162*  1370*  1371*  1453*  1623*  1648*  1651*  1657*  1677*  1712*  1774
                4295*  4296*  4297  4337*  4737*  4738*  4781*  4824*  4825*  4826*  4827*  5962*  5971*
                6044*  6045*  6084*  6085*  6086*  6087*  7599*  7600*  7609*  7983*  8959  8960*  8962*
                8965*  10165  10166  10170*  10175*  10187*  10192*  10202*  10214*  10215*
ERR4    037452  6955#
EXIT    046000  7844  7933  7939#
EXITFL  001564  747#
EXITRE  044152  7628  7630  7661  7667#  7941
EXPEXT  060674  6681  6692  6737  9924  9946#
EXTINS  026344  5106#
E.DSC   041222  7045  7049  7058  7077  7087  7090  7101  7103  7107  7110  7149  7161  7163
                7166  7170  7186  7195  7198  7325#
E1      040012  7045#  7377
E10     040402  7145#  7377
E11     040420  7152#  7381
E12     040422  7153#  7381
E13     040464  7170#  7381
E14     040454  7166#  7381
E15     040520  7186#  7381
E16     040536  7195#  7381
E17     040544  7198#  7381
E18     040134  7090#  7381
E2      040022  7049#  7377
E3      040044  7058#  7377
E4      040106  7077#  7377
E5      040126  7087#  7377
E6      040172  7101#  7377
E7      040176  7103#  7377
E8      040206  7107#  7377
```

```
E9          040214          7110#   7377
F           041230          7229    7328#   7391
FACTOR      001534           735#   1812*   1930*   2974    3105*   3219    3676    3677    3958    4019    4134    4135    4139
                            4140    4171*   4242    4296    4422    4437    4445    4461    4486    4523    4723*   4842    4848
                            4859    5099*   5205    5525*   5849    5868    6012*   6336*   6776*   6813    6817    6819    7007
                            7023    7025    7117    7119    7421*   7638    7646*   10360
FLD20  =    104426          9210   10269#
FLTADD      036276          6352    6392    6395    6425    6522    6562    6565    6595    6703#
FLTDBL      060534          6516    9902#
FLTDIV      036250          6485    6655    6689#
FLTMPY      036226          6355    6365    6374    6384    6387    6437    6445    6453    6486    6525    6535    6544    6554
                            6557    6607    6615    6623    6656    6678#
FLTMP0      001432           715#   6526*   6578    6608*   6639    6657*   11045
FLTMP1      001442           716#   6566*   6629*   6660*   11045
FLTSGL      060542          6346    9903#
FLTSUB      036272          6410    6433    6458    6471    6493    6580    6603    6628    6641    6663    6702#
FL20   =    104424          9200    9827   10268#
FNDLOC      037502          6962    6965#
FNDSCN      037130          6861    6864#
FNDSKP      037572          6983    6986#
FNDSPN      037224          6885    6888#
FPOPT  =    020000           505#   1379    6343
FRSTAD      001540           737#   1809*   1927*   3102*   3754    4168*   4720*   5096*   5522*   6009*   6333*   6773*   7631
FRSTME      001542           738#   1251*   7641
F.DSC       041226          7050    7057    7063    7067    7072    7078    7104    7111    7153    7169    7181    7187    7327#
F1          040024          7050#   7384
F10         040416          7151#   7384
F11         040446          7163#   7388
F12         040462          7169#   7388
F13         040510          7181#   7388
F14         040522          7187#   7388
F2          040040          7056#   7384
F3          040054          7063#   7384
F4          040064          7067#   7384
F5          040076          7072#   7384
F6          040110          7078#   7384
F7          040200          7104#   7384
F8          040216          7111#   7384
F9          040412          7149#   7384
GETMAP      062676          8119    8200    8419    8540    8794   10290   10430#
GIVEMA      063142          8417    8453    8538    8574    8788    8823   10489#
GNS    =    ******  U        604    1141    1145    1187    1202    1206    1211    1266    1270    1274    1283    1291    1572
                            1598    1604    1610    1690    1696    1762    7738    7747    8043    8050    9225   10258   10259
                           10260   10261   10262   10263   10264   10265   10266   10267   10268   10269   10641
GSTST       012170          2183#
HALT1       032166          5960#
HIADRS=     177742           201#   10400   10616
HIMASK      041162          7125    7126    7127    7304#
HITMIS=     177752           205#
HT     =    000011            89#   9362    9403
INCB1       012746          2366    2367    2369#
INCB2       014070          2703    2704    2705    2707#
INCB3       014456          2837    2839#
INCB6       015200          3001    3003#
INCB6A      015350          3052    3053    3055#
INCB7       016240          3259    3260    3262#
```

```
INC0    011574          2044    2045    2046    2047    2049#
INC1    012522          2284    2285    2286    2288#
INC3    014276          2778    2780#
INC4    013404          2529    2530    2532#
INC6    014756          2935    2936    2937    2939#
INC7    016044          3202    3204#
INSTBL  003310          1168    1175    1195#
IOMON   044160          7612    7677#   7883
IOTTST  023546          4542#
IOTVEC= 000020           187#   1345*   1346*   4542    4555*   4610*   4611*   4656*   4657*   5933*   5952*
IOWC    001574           751#   7654*   7687*   7777
JMP1    022746          4353    4354    4355    4357#
JMP3    023016          4373    4377#
JMP4    023054          4389    4393#
JMP5    023120          4407    4411#
JMP6    023140          4419#
JMP7    023176          4430#
JSR1    023216          4437#
JSR3    023302          4447    4461#
JSR4    023370          4473    4486#
JSR6    023446          4495    4509#
JSR7    023500          4513    4523#
KB11CM  001504           723#   1468*   1516*   1579
KB11E   001502           721#   1109    1120    1127    1469*   1473*   1512    1514*   1517    1577    1585    4835    4935
                        5896    6750
KB11EM  001503           722#
KDPAR0= 172360           359#   6184
KDPAR1= 172362           360#
KDPAR2= 172364           361#
KDPAR3= 172366           362#
KDPAR4= 172370           363#
KDPAR5= 172372           364#
KDPAR6= 172374           365#
KDPAR7= 172376           366#
KDPDR0= 172320           337#   6176
KDPDR1= 172322           338#
KDPDR2= 172324           339#
KDPDR3= 172326           340#
KDPDR4= 172330           341#
KDPDR5= 172332           342#
KDPDR6= 172334           343#
KDPDR7= 172336           344#
KERSTK= 001200            76#   1218    1312    4815    7954    7988
KIPAR0= 172340           348#   6181    7560*   7943*   7951    10145   10179   10350   10660   10689
KIPAR1= 172342           349#   7561*   7944*
KIPAR2= 172344           350#   7562*   7945*
KIPAR3= 172346           351#   7563*   7946*
KIPAR4= 172350           352#   6200*   7564*   7565    7838*   7923    7925*   7927    7937*   7943    7944    7945    7946
                        10391*
KIPAR5= 172352           353#   7565*   7566*   7838    7923    7927
KIPAR6= 172354           354#  10618   10621*  10633*
KIPAR7= 172356           355#   7567*   7581    7595    10196
KIPDR0= 172300           326#   1493*   1494    1496*   6173    7551*
KIPDR1= 172302           327#   7552*
KIPDR2= 172304           328#   7553*
KIPDR3= 172306           329#   7554*
```

```
KIPDR4= 172310          330#    6201*   7555*
KIPDR5= 172312          331#    7556*
KIPDR6= 172314          332#    7557*   10622*
KIPDR7= 172316          333#    7558*
KM    = 000000          516#
KTABRT  064122         1794     5951    10700#
KTABT   033222         6198#
KTEX    033462         6199     6250#
KTOPT = 100000          503#
KTPAR   033042         6138#
KTPDR   032716         6099#
LDKT    063752         8783     8862    8925    9411    10559   10655#
LDRP3   047174         8133     8139#   8388
LDRP4   050116         8269     8276#   8607    8633    8646    8661    8676
LDRS    050422         8324     8333#   8706    8727    8740    8756    8771
LD2PNT  001632          767#    1538    1541
LD2PT1  001634          768#
LD3PNT  001636          769#
LF    = 000012           90#    9397    9403
LKOPT = 001000          508#    1383    7446    7492
LKS   = 177546          472#    1381    7456*   7458    7496*   8937*
LKSRV   054300         7477     7494    8924#
LKVEC = 000100          473#    7449*   7450*   7477*   7494*   7495*
LOADRS= 177740          200#   10399   10614
LOCATE  037460         6954     6958#
LOCC  = 076040          541#
LOCCHR  037470         6961#    6963*
LOCCI = 076140          573#    6959
LOMASK  041164         7129     7131    7133    7306#
LONG.1  041242         7088     7092    7094    7196    7200    7202    7225    7333#   7338    7394
LONG.2  041246         7091     7092    7094    7199    7200    7202    7226    7334#   7339    7394
LONG1   040130         7088#    7394
LONG1P  041260         7338#
LONG2   040540         7196#    7394
LONG2P  041262         7339#
LONG3   040546         7199#    7394
LONG4   040136         7091#    7394
LOOP    010000         1768     1772#   7957    8007    8065
LOOP1   007346         1746     1752    1754    1756    1768#
LSTMEM  001544          739#    1249*   1250*   7643
LTICKS  001630          765#    7802    7819    8926*   8927    8929*   8930*   8931    8933*   9412
L2D0  = 076020          524#    1537
L2D1  = 076021          531#
L2D2  = 076022          532#
L2D3  = 076023          533#
L2D4  = 076024          534#
L2D5  = 076025          535#
L2D6  = 076026          536#
L2D7  = 076027          537#    6905    7076
L3D0  = 076060          555#
L3D1  = 076061          525#    1540
L3D2  = 076062          556#
L3D3  = 076063          557#
L3D4  = 076064          558#
L3D5  = 076065          559#
L3D6  = 076066          560#
```

```
L3D7  = 076067          561#   6843   6880   6890   6895   6911   6925   6930   6992   7060   7065   7070   7164
                       7174   7179
MAINT = 177750          204#   1474*  1477   7997*  8009*
MAPEND  062426        10311  10321#
MAPH0 = 170202          443#   6305*
MAPH00= 170202          379#    443   1478
MAPH01= 170206          381#    445
MAPH02= 170212          383#    447
MAPH03= 170216          385#    449
MAPH04= 170222          387#    451
MAPH05= 170226          389#    453
MAPH06= 170232          391#    455
MAPH07= 170236          393#    457
MAPH1 = 170206          445#   6303*
MAPH10= 170242          395#
MAPH11= 170246          397#
MAPH12= 170252          399#
MAPH13= 170256          401#
MAPH14= 170262          403#
MAPH15= 170266          405#
MAPH16= 170272          407#
MAPH17= 170276          409#
MAPH2 = 170212          447#
MAPH20= 170302          411#
MAPH21= 170306          413#
MAPH22= 170312          415#
MAPH23= 170316          417#
MAPH24= 170320          419#
MAPH25= 170326          421#
MAPH26= 170332          423#
MAPH27= 170336          425#
MAPH3 = 170216          449#
MAPH30= 170342          427#
MAPH31= 170346          429#
MAPH32= 170352          431#
MAPH33= 170356          433#
MAPH34= 170362          435#
MAPH35= 170366          437#
MAPH36= 170372          439#
MAPH37= 170376          441#
MAPH4 = 170222          451#
MAPH5 = 170226          453#
MAPH6 = 170232          455#
MAPH7 = 170236          457#
MAPL0 = 170200          442#   6260   6293   6304*  10315  10464
MAPL00= 170200          378#    442
MAPL01= 170204          380#    444
MAPL02= 170210          382#    446
MAPL03= 170214          384#    448
MAPL04= 170220          386#    450
MAPL05= 170224          388#    452
MAPL06= 170230          390#    454
MAPL07= 170234          392#    456
MAPL1 = 170204          444#   6302*
MAPL10= 170240          394#
MAPL11= 170244          396#
```

```
MAPL12= 170250            398#
MAPL13= 170254            400#
MAPL14= 170260            402#
MAPL15= 170264            404#
MAPL16= 170270            406#
MAPL17= 170274            408#
MAPL2 = 170210            446#
MAPL20= 170300            410#
MAPL21= 170304            412#
MAPL22= 170310            414#
MAPL23= 170314            416#
MAPL24= 170320            418#
MAPL25= 170324            420#
MAPL26= 170330            422#
MAPL27= 170334            424#
MAPL3 = 170214            448#
MAPL30= 170340            426#
MAPL31= 170344            428#
MAPL32= 170350            430#
MAPL33= 170354            432#
MAPL34= 170360            434#
MAPL35= 170364            436#
MAPL36= 170370            438#
MAPL37= 170374            440#
MAPL4 = 170220            450#
MAPL5 = 170224            452#
MAPL6 = 170230            454#
MAPL7 = 170234            456#
MAPTBL  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    031664           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#
```

```
MBTST = 160112          491#    978
MBTTBL  002306          972#   1424    1444    1448*   1449*   1450    7519*   7520*   7521*   7522*   7523*   7524*   8864
                       8869*   8901    8915*  10773
MBTVEC= 000774          499#    981
MBTWC = 160102          487#    973
MBT1    043206         7519#
MED6X = 076600          595#
MED74C= 076601          526#   1544
MEMERR= 177744          202#  10223   10226   10229*  10398   10404*  10613   10627*  10628   10634   10635*
MEMHOL  054134         8880    8882    8889#
MEMSIZ  003656         1245    1247    1249#
MFPT  = 000007          528#   1471    6790
MFPTTR  005552         1470    1522#
MHOLE   053636         8818    8821    8831#
MMON    001531          732#   1314*   4923    5923    6104    6143    6198    7517    7598*   7629    7749    7794    7826
                       7853    7880    7896    7939    7975*  10343   10387   10619   10655   10681
MMR0  = 177572          227#    231    10703   10709*
MMR1  = 177574          228#    232
MMR2  = 177576          229#    233    10704
MMR3  = 172516          230#    234     6258    7504    7966*   8115    8196    8414    8450    8535    8571   10307
MMVEC = 000250          195#   1794*   1795*   5919*   5951*   6206    6207    6210*   6211*   6247*   6248*
MOVB1   017346         3565    3566    3567    3568    3571#
MOVC  = 076030          538#   6847    6894    6970
MOVCI = 076130          570#   6866    6988
MOVE    037042         6838    6842#
MOVER   037340         6916    6920#
MOVRC = 076031          539#   6929
MOVRCI= 076131          571#   6921
MOVT    037410         6935    6939#
MOVTC = 076032          540#
MOVTCI= 076132          572#   6940    6945
MOV0    016416         3310    3311    3313#
MOV0A   016446         3325    3327#
MOV1    017216         3523    3525#
MOV7    021470         4076    4078#
MPI     031754         5915#
MRGTAB  064400         8008    8009   10757#
MRKTST  027320         5354#
MSGINX  064440         7732    9191    9219    9481   10775#
MSG1    064464        10785#
MSG10   064613        10779   10805#
MSG11   064621        10780   10807#
MSG12   064627         9230   10809#
MSG13   064742         9237   10822#
MSG14   065052         9320   10834#
MSG15   065110        10783   10839#
MSG16   065131         9255   10842#
MSG17   065240         9261   10854#
MSG2    064476        10787#
MSG20   065314         8014   10862#
MSG21   065414        10777   10778   10781   10782   10873#
MSG22   065434         9276   10876#
MSG23   065444         9281   10878#
MSG24   065453        10784   10880#
MSG25   065475         1747   10884#
MSG3    064510         9079   10789#
```

```
MSG30   070517          1767   9471   11197#
MSG31   071031          1589   11233#
MSG32   071043          1583   11235#
MSG34   071075          1576   11240#
MSG35   071135          1581   11246#
MSG36   071150          1587   11248#
MSG37   071161          1592   11250#
MSG38   071211          1594   11255#
MSG4    070364          9472   11181#
MSG5    064577         10775   10801#
MSG6    064605         10776   10803#
MSKTAB  041132          7006    7292#
MTICKS  001626           764#   1322*  8934*  9422
MULP  = 076074           566#
MULPI = 076174           591#   7142   7146
MULO    030562          5635#
MXMMHI  001606           756#   1309*  8816   8881
MXMMLO  001610           757#   1310*  8819   8879
NEGB1   013016          2390    2391   2393#
NEGB4   014020          2680    2681   2683#
NEGB6   015304          3034    3035   3037#
NEGB7   016322          3284    3285   3287#
NEGO    011614          2054    2055   2056   2058#
NEG1    012646          2334    2335   2337#
NEG2    013426          2541    2543#
NEG5    014254          2766    2767   2769#
NEG6    014702          2913    2914   2915   2917#
NEG7    015752          3172    3173   3175#
NEXEC   001530           731#   1816   1934   3109   4175   4727   5103   5529   6016   6340   6780   7958
NEXPAR  001546           740#   1315*  1318*  7564   7574   7588   7679   7937   7942*  7971*  7974*
NINTAB  041142          7296#
NOMEM   044154          7644    7666   7668#
NOTRAP  005724          1556    1559#
NOTYPE  001476           719#   1366*  9388   10551*  10569
NOZERO  040440          7154    7160#
NULLS   001554           743#   7428   10590
NUMPAC  040166          7095    7099#
NUMRIC  037776          7035    7039#
NWBASH  001572           750#   7651*  7683*  7791   7800*
NWBASL  001570           749#   7650*  7682*  7752   7790   7799*  7832
NXLOC   037464          6959#   6964
NXSCAN  037112          6858#   6863
NXSKIP  037554          6980#   6985
NXSPAN  037202          6880#   6887
OAERR   022476          4288#
OBJ.1A  041204          7318#   7390
OBJ.1D  041202          6907    6913   7317#
OBJ1    037300          6907#   7360
OFFTAB  041266          6812    7344#
OKSIZ   004254          1262    1301#
OLDBAS  001566           748#   7649*  7678*  7751   7789   7798*  7828
OLDPSW  043304          7545#   7956
ONE     041252          7073    7167   7182   7335#
ONE1    040100          7073#   7396
ONE2    040512          7182#   7396
ONE3    040456          7167#   7396
```

```
OPT.CP  001500   720#  1565*  1748  4843  6343  6787  7423  7446  7492  7500  7511
OVFLW   024514  4731#
PACDAT  040230  7112  7116#
PACKED  040374  7139  7142#  7159
PARTBL  033166  6138  6181#
PA1500  001524   729#  8814*  8819  8839  8875*  8877*  8879  8897  9094*  9097  9172  9184*  9292
               10309*  10316*  10319*  10357*
PA2116  001526   730#  8815*  8816  8840  8876*  8878*  8881  8898  9095*  9185*  10310*  10317*  10320*
               10356*
PDRTBL  033150  6099  6173#
PERET   063600  10628#
PHYMAP  062324  8813  9291  10303#
PIRQ  = 177772    84#  5843  5862  5863*
PIRQVE= 000240   194#  5845*  5846*  5848*  5849*  5853*  5881*  5882*
PIRQ0   031436  5839#
PKM   = 000000   518#
PLKCSB= 172542   470#
PLKCSR= 172540   469#
PLKVEC= 000104   471#
PROT    003574  1229#  1248*  1250  1316  7972
PR0   = 000000   116#
PR1   = 000040   117#
PR2   = 000100   118#  5467  5475
PR3   = 000140   119#
PR4   = 000200   120#  1363  1365  4544  4562  5492  5498  7546
PR5   = 000240   121#  4551  5465  7523  7786
PR6   = 000300   122#
PR7   = 000340   123#  1777  1783  1788  1790  1792  1796  4550  4600  4611  4655  4665  4666
               4696  4698  4734  4826  5414  5416  5422  5442  5481  5486  5493  5798  5846
               5847  5884  6086  7450  7495  10285  10434
PS    = 177776    81#  82  1339*  10170
PSM   = 010000   513#  4577
PSW   = 177776    82#  1163*  1184*  1764*  1775  1796*  2183  2197*  2202*  4601*  4617  4665*  4671
               4699*  4731  4732*  4734*  4762*  4767*  4825  4925  4926*  4929  4965  4966*  5030*
               5076*  5405  5414*  5422*  5465*  5471  5481*  5486  5492*  5498  5502*  5539  5543
               5547*  5549  5565  5571  5723  5727  5747  5751  5845  5850*  5883*  5915  5950*
               5973*  5982  5985  5988*  6045  6052  6085  6211  6221  7439  7443*  7462*  7544
               7546*  7955*  7986*  8368  8406  8447  8487  8527  8568  8609  8648  8678  8708
               8742  8773  10432  10477*  10498  10509*  10658  10659*  10694*
PSWCHK  025464  4923#
PSWTAB  064364  8006  10749#
PUM   = 030000   519#  4576  4582  5465  5481  5486  5492  5498
PWRVEC= 000024   188#  1351*  1352*  9991*  9992*  10000*  10015*  10016*
QV      001532   733#  1224*  1232*  1244  1751  6300  7741  8010  10447
RDCHR = 104412  10057  10263#
RDDEC = 104416   1208  10265#
RDLIN = 104414  10097  10264#
REG   = 004000   514#  5454  5457  5465  5486  5498
REGINX  064414  9242  9257  9310  10765#
RELE0   011152  1818  1908#
RELE1   015436  1936  3083#
RELE2   021744  3111  4149#
RELE3   024376  4177  4701#
RELE4   026226  4729  5077#
RELE5   030054  5105  5503#
RELE6   032312  5531  5990#
```

```
RELE7   033666          6018    6301    6306#
RELE8   036434          6342    6668    6670    6741#
RELE9   042526          6782    6798    7206    7406#
RELNIO  044116          7648    7657#   7659    7756
RELOC   043764          1911    3086    4152    4704    5080    5506    5993    6309    6744    7409    7627#
RELOCP  045672          7686    7759    7920#   7928
REL0    010156          1807#
REL00   011166          1912#
REL1    011214          1925#
REL11   015452          3087#
REL2    015500          3100#
REL22   021760          4153#
REL3    022006          4166#
REL33   024412          4705#
REL4    024440          4718#
REL44   026242          5081#
REL5    026270          5094#
REL55   030070          5507#
REL6    030116          5520#
REL66   032326          5994#
REL7    032354          6007#
REL77   033702          6310#
REL8    033736          6331#
REL88   036450          6345    6745#
REL9    036510          6771#
REL99   042542          7410#
RESERR  064216          1564    1780    4338    4916    5972    10716#
RESET1  032264          5981#
RESKT   064050          8798    8826    8853    8867    8913    8935    9455    10567   10681#
RESPSW  063176          10505#  10511
RESREG= 104422          1519    1547    1562    8076    8129    8153    8206    8226    8268    8293    8323    8799    8827
                        8854    8868    8914    8936    9145    9456    9673    9799    9852    10267#  10321   10358   10476
                        10568
RESTOR  005732          1518    1562#
RESTPS  063206          10509#
RESTRP  025146          4835#
RESVEC= 000010          183#    1148*   1149*   1372*   1373*   1470*   1535*   1564*   4297*   4338*   4847*   4848*   4916*
                        5963*   5972*   7786*   7787*   7900*
RETPC   044156          7640*   7669#
RETPSW  063170          10500#  10510
RETRY   043672          7599#   7938
RKBA    002172          918#    8211*   8508*   8550*
RKCS    002166          916#    1628*   1637*   7889*   8215*   8478    8482*   8484    8498    8509*   8515    8522*   8524
                        8530    8551*   8557    8563*   8565    9277
RKDA    002174          919#    1627*   8207*   8506*   8548*
RKDRV   047226          854     8149#
RKDS    002162          914#    1631    8208    10766
RKER    002164          915#    1629
RKERR   051436          8481    8490#   8520    8561
RKFUN   002122          877#    8214*   8462*   8463    8468*   8470    8562*
RKHANA  002070          854#
RKHDA   002106          866#    8194*   8207    8506    8548
RKHSTA  001740          785#    8150*   8151    8156*   8476    8490*   8497*   8500*   8513    8555    8576*
RKHWC   001760          798#    8210    8507    8549
RKLOOP  051464          8477    8497#   8514    8556
RKNEWH  002030          828#    8542    8544*   8547
```

```
RKNEWL  002026            827#  8539  8550
RKOLD   001776            810#  8199  8202  8205*  8211  8505  8508
RKPSW   002200            921#  8213*
RKREAD  052014           8473   8555#
RKRPT   051300           8154   8461#
RKSRV   051330           8212   8468#
RKTRY   002133            889#  8195*  8480  8486*  8503*  8519  8526*  8534*  8559  8567*
RKUNIT  002054            843#  8191
RKVEC   002176            920#  8212*
RKWC    002170            917#  8210*  8507*  8549*
RKWRCK  051564           8471   8513#
RKWTRY  047542           8207#  8466  8488
RK1     051510           8464   8479  8503#
RK10    001616            760#  8201*  8204*  8215  8504*  8505*  8509  8546*  8547*  8551
RK11    001620            761#  8155*  8173  8492  8517
RK2     051532           8506#  8528
RK3     051762           8467   8548#  8569
RNTBIN  001604            755#  7689*  7782*  7783  7796  8078  8155  8228  8295
ROLB1   012760           2372   2373   2375#
ROLB2   013712           2638   2639   2640   2642#
ROLB3   014532           2862   2863   2865#
ROLB6   015320           3040   3041   3043#
ROLB6A  015414           3070   3071   3073#
ROLB7   016340           3291   3292   3294#
ROL0    011646           2070   2071   2073#
ROL1    012442           2256   2257   2258   2260#
ROL1A   012454           2263   2264   2266#
ROL3    014324           2790   2791   2793#
ROL4    013502           2566   2567   2569#
ROL6    014632           2894   2895   2896   2898#
ROL7    016062           3208   3209   3211#
RORB1   013046           2404   2405   2407#
RORB1A  013136           2437   2438   2440#
RORB4   013674           2630   2631   2632   2634#
RORB5   014474           2843   2844   2845   2847#
RORB6   015252           3021   3022   3024#
RORB7   016224           3253   3254   3256#
RORO    011532           2026   2027   2028   2029   2031#
ROR1    012372           2234   2235   2237#
ROR1A   012504           2276   2277   2278   2280#
ROR2    013370           2522   2523   2525#
ROR5    014176           2739   2740   2742#
ROR6    014722           2921   2922   2923   2925#
ROR7    015734           3165   3166   3168#
RPCC    002232            936#
RP3BA   002150            907#  8142*  8431*
RP3CS   002144            905#  1661*  1662   8132*  8137*  8360  8364*  8365  8379  8389*  8395  8402*  8403
                         8432*  8438   8443*  8444   9280
RP3DA   002152            908#  8113*  8139*  8428*
RP3DC   002154            909#  8140*  8429*
RP3DRV  046610            853   8072#
RP3DS   002140            903#  1668   8130   10765
RP3ER   002142            904#  8409
RP3ERR  050612           8363   8371#  8400   8441
RP3FUN  002120            876#  8136*  8345*  8346   8351*  8352  8401*  8442*
RP3HAN  002066            853#  7787   7900
```

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# 8280
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 925# 8280*

```
RP4WCK  052426  8596  8638#
RP4WTR  050064  8269#  8593  8610
RP41  052376  8591  8603  8630#
RP411  001622  762#  8228*  8245  8619  8653
RP42  052404  8631#  8632  8649
RP43  052536  8592  8659#  8660  8679
RSBA  002246  944#  8337*  8758*
RSBAE  002250  945#  8336*  8757*
RSCS1  002242  942#  1717*  8330*  8728*  8759*  10770
RSCS2  002254  947#  1716*  1718  1726*  7893*  8333*  8705*  8739*  8746  8770*
RSCS3  002256  948#
RSDA  002252  946#  8334*
RSDRV  050172  858  8289#
RSDS  002260  949#  1720  8328  8701  8719  8725  8734  8754  8765
RSER  002262  950#
RSERR  053006  8704  8711#  8737  8768
RSFUN  002130  880#  8327*  8688*  8689  8693*  8694  8738*  8769*
RSHANA  002100  858#
RSHDA  002116  870#  8321*  8334
RSHSTA  001750  789#  8290*  8291  8699  8711*  8718*  8721*  8732  8763  8776*
RSHWC  001770  802#  8335
RSLOOP  053034  8700  8718#  8733  8764
RSNEWH  002050  836#  8757
RSNEWL  002046  835#  8758
RSOLD  002016  818#  8336  8337
RSPSW  002266  952#  8326*
RSREAD  053260  8696  8763#
RSRPT  052666  8294  8687#
RSSRV  052712  8325  8693#
RSTRY  002136  892#  8322*  8703  8707*  8724*  8736  8741*  8753*  8767  8772*
RSUNIT  002064  847#  8333
RSVEC  002264  951#  8325*
RSWC  002244  943#  8335*
RSWCK  053112  8695  8732#
RSWTRY  050362  8324#  8692  8709
RS11  001624  763#  8295*  8310  8713  8748
RS41  053062  8690  8702  8724#
RS42  053070  8725#  8726  8743
RS43  053222  8691  8754#  8755  8774
RTI1  046374  8011  8016#
RTT1  027422  5405#
RTT1EX  027624  5443#
RTT2A  027646  5450  5452#
RTT2EX  030046  5408  5461  5476  5478  5487  5489  5499  5502#
RT1  063744  10609  10647#
RUNTBL  001662  772#  7784*  7860  8094  8097*  8174  8177*  8246  8249*  8312*
RUNTRA  001700  773#  7915  8111*  8188*  8264*  8318*  8374*  8493*  8620*  8714*
SAVPSW  027644  5405*  5406  5412  5437  5451#  5452  5502
SAVREG=  104420  1467  1534  8072  8149  8222  8289  8782  8861  8924  9076  9410  9655  9749
                 9824  10266#  10303  10341  10436  10557
SAVRNH  041170  6807*  6811  7309#
SAVRNL  041166  6806*  6810  7308#
SBCB1  012772  2378  2379  2381#
SBCB3  014552  2873  2875#
SBCB4  014006  2674  2675  2677#
SBCB6  015154  2991  2992  2994#
```

```
SBCB7   016152          3231    3232    3233    3235#
SBC0    011676          2084    2085    2087#
SBC1    012536          2291    2292    2293    2295#
SBC1A   012554          2299    2300    2301    2303#
SBC5    014336          2797    2799#
SBC6    014736          2928    2929    2931#
SBC7    015640          3135    3136    3138#
SBINB7  022204          4217#   4248*   4249    4253    4260    4264    4268*   4271*   4274    4278*   4281*   4284
SBIN7   021416          4059#   4074*   4075    4084    4090    4097*   4098    4105    4109    4113
SCAN    037100          6852    6856#
SCANC =  076042          543#
SCANCI=  076142          575#    6858
SDATA   020732          3955#   3961*   3962    3963*   3964*   3969*   3970*   3971    3975    3980    3986*   3987    3993*
                        3998    3999*   4000*   4001    4004*   4005    4008
SDATAB  021374          4025*   4026*   4027    4029    4033    4036    4040    4041    4049#
SDPAR0=  172260          315#    6186
SDPAR1=  172262          316#
SDPAR2=  172264          317#
SDPAR3=  172266          318#
SDPAR4=  172270          319#
SDPAR5=  172272          320#
SDPAR6=  172274          321#
SDPAR7=  172276          322#
SDPDR0=  172220          293#    6178
SDPDR1=  172222          294#
SDPDR2=  172224          295#
SDPDR3=  172226          296#
SDPDR4=  172230          297#
SDPDR5=  172232          298#
SDPDR6=  172234          299#
SDPDR7=  172236          300#
SECT2   034422          6416    6419#
SECT2D  035534          6586    6589#
SECT3   034766          6477    6481#
SECT3D  036100          6647    6651#
SETOP   005734          1553    1563#
SETUP   040602          6833    6842    6856    6878    6904    6920    6939    6958    6979    7212#
SET.1D  041236          6857*   6860    6862*   6879*   6883    6886*   7331#
SET1    037116          6860#   7360
SET2    037210          6883#   7360
SIGN    041163          7130    7132    7134    7305#
SIPAR0=  172240          304#    5927    6183    7588*   7589    7591    7593
SIPAR1=  172242          305#    7589*   7590*
SIPAR2=  172244          306#    7591*   7592*
SIPAR3=  172246          307#    7593*   7594*
SIPAR4=  172250          308#    6204*
SIPAR5=  172252          309#
SIPAR6=  172254          310#    5927*
SIPAR7=  172256          311#    7595*
SIPDR0=  172200          282#    6175    7583*
SIPDR1=  172202          283#    7584*
SIPDR2=  172204          284#    7585*
SIPDR3=  172206          285#    7586*
SIPDR4=  172210          286#    6205*
SIPDR5=  172212          287#
SIPDR6=  172214          288#    5928*
```

```
SIPDR7= 172216            289#  7587*
SIXTAB  041152           7300#
SIZE    006272           1623#  1766
SIZEHI= 177762            213#  1277
SIZELO= 177760            211#  1261  1285
SKIP    037550           6975   6979#
SKPC  = 076041            542#
SKPCHR  037560           6982#  6984*
SKPCI = 076141            574#  6980
SM    = 040000            512#  5457  5498
SOBDBL  060672           9902*  9903  9909  9921  9925*  9927#
SOB0    027104           5266   5277#
SOB1    027112           5279#  5283
SOB10   027066           5268#  5331
SOB2    027140           5279   5280  5281  5282  5284  5285  5286  5287  5290#
SOB3    027152           5294#  5302
SOB4    027204           5294   5295  5296  5297  5300  5303  5304  5307#
SOB5    027206           5309#
SOB5A   027226           5318#
SOB6    027252           5329#
SOB7    027260           5332#
SOB8    027262           5275   5334#
SOB9    027076           5273#  5344
SPAN    037170           6874   6878#
SPANC = 076043            544#  6884
SPANCI= 076143            576#
SPARE0  001742            786#
SPARE1  001744            787#
SPCHK   025736           4988#
SPLTST  031260           5795#
SRC.1A  041174           6865*  6889*  6909*  6966*  6987*  7213*  7222  7314#  7390
SRC.1D  041172           6835   6844   6849   6859   6864*  6867   6871  6881   6888*  6891  6896  6906  6910*
                         6912   6922   6931   6941   6951   6960   6965* 6972   6981   6986* 6989  6993  7212*
                         7313#
SRC1    037024           6835#  7344
SRC10   037276           6906#  7344
SRC11   037320           6912#  7348
SRC12   037346           6922#  7348
SRC13   037370           6931#  7348
SRC14   037416           6941#  7348
SRC15   037442           6951#  7348
SRC16   037466           6960#  7348
SRC17   037532           6972#  7348
SRC18   037556           6981#  7348
SRC19   037604           6989#  7348
SRC2    037050           6844#  7344
SRC20   037614           6993#  7348
SRC3    037062           6849#  7344
SRC4    037114           6859#  7344
SRC5    037142           6867#  7344
SRC6    037152           6871#  7344
SRC7    037204           6881#  7344
SRC8    037236           6891#  7344
SRC9    037250           6896#  7344
SR0   = 177572            231#  6226  6249*  7596*  7965*  10176  10191*  10199*
SR1   = 177574            232#  6232  6235
```

```
SR2    = 177576            233#   6229
SR3    = 172516            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
SUBNI  = 076151            580#   7055
SUBP   = 076071            563#
SUBPAS   001560            745#   1331*  7515    7991*  7992   7996*  8004   9115
SUBPI  = 076171            588#   7150   7160
SUB0     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#
SUPSTK= 000700              77#   1772   4336    4602   4768   4774   4809   4823   5636   5953   6058   6078   6082
                           6261   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#
```

```
SW3     = 000010            148#  7513
SW4     = 000020            147#  8079  8157  8230  8296
SW5     = 000040            146#  7692
SW6     = 000100            145#  7535  7627
SW7     = 000200            144#  1757
SW8     = 000400            143#  7647  7684
SW8MSG    007342           1758   1767#
SW9     = 001000            142#  7845
SXRA      026720           5215#  5224*
SXRB      026722           5216#  5242
SXT0      026366           5109   5110  5111  5112  5115#
SXT1      026452           5142#
SXT2      026736           5222#
SXT3      026752           5228#
SXT4      026520           5160   5162#
SXT5      026762           5233#
SXT6      026662           5197   5198  5199  5200  5202#
SXT6A     026712           5208   5211#
SXT7      027026           5250#
SYSSIZ    001642            771#  1640* 1674* 1709* 1729* 1734* 1742* 1744* 7709  7712  7722  7729* 7770*
                           9475   9486
SYSTID=   177764            215#
TBITVE=   000014            184#  4666* 4698* 4928  4941* 4944* 4959* 5411* 5413* 5440* 5441* 5442* 5847*
                           5884* 8003*
TEST1     012632           2326   2327  2328  2330#
TEST2     013344           2510   2511  2513#
TEST6     015430           3076   3077  3079#
TIMEBU    056734           1332   1337* 1338* 9416* 9421* 9426* 9431* 9438* 9445* 9452* 9453  9458#
TIMER     052140           8483   8523  8564  8578#
TKBFR     001454            717    718# 10563 10565
TKBFRP    001452            717#  10560 10566*
TKISR     063222           1362   10531#
TKVEC   = 000060            191#  1362* 1363*
TPISR     063434           1364   7481  10585#
TPVEC   = 000064            192#  1364* 1365* 7448* 7481*
TRANS     040732           6944   6949  7240#  7332  7337
TRANS1    037424           6944#  7360
TRANS2    037436           6949#  7360
TRANS3    041256           7337#  7360
TRANS4    041240           7332#  7391
TRAPVE=   000034            190#  1106* 1199* 1263* 1349* 1350* 4669* 4671* 4677* 4680* 4684* 4689* 4695
TRAP1     024310           4668   4683#
TRAP1C    024350           4682   4695#
TRPRTN    005716           1535   1557#
TRTVEC=   000014            185#
TSTB1     013210           2462   2463  2465#
TSTB2     014030           2686   2688#
TSTB2A    014040           2691   2693#
TSTB6     015126           2979   2980  2981  2983#
TST0      011452           1999   2000  2001  2002  2004#
TST1      010140           1801#
TST10     013564           2600#
TST11     014110           2718#
TST12     014340           2803#
TST13     014566           2885#
TST14     015054           2970#
```

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#	

```
TST77   043032          7488#
TTOPT = 000400           509#   1387   7423
TTYCHK  042570          7421#
TYPDS = 104410          1607   8047   8054  10262#
TYPE  = 104400          1139   1143   1185   1200   1204   1209   1264   1268   1272   1276   1281   1289   1570
                        1576   1581   1583   1587   1589   1592   1594   1596   1602   1608   1612   1688   1694
                        1747   1750   1760   1767   7734   7736   7745   8041   8048   8055   9024   9032   9077
                        9079   9080   9084   9101   9103   9106   9110   9114   9118   9122   9135   9137   9140
                        9142   9146   9166   9176   9193   9203   9213   9221   9223   9230   9237   9255   9261
                        9266   9276   9281   9296   9298   9315   9317   9318   9320   9323   9367   9453   9454
                        9470   9471   9472   9483   9490   9493   9558   9634  10017  10061  10064  10068  10134
                       10136  10258#  10639
TYPOC = 104402          1213   1601   1693   1749   9083   9105   9109   9113   9117   9121   9163   9265   9314
                        9322  10259#
TYPON = 104406         10261#
TYPOS = 104404          1278   1286   1294  10260#
TYPSIZ  056746          1759   9470#
TYPTIM  056506          8030   9078   9410#
T.END   005470          1500   1503#
T1      005322          1474#  1523
T2      005364          1481   1484#
T3      005416          1486   1489   1492#
T4      005450          1495   1498#
UBEADR  001726           777#  8791*  8792*  8793   8795   8796   8848   8850  10277*  10278*  10289  10291  10292
UBEBA = 170004           478#   960
UBECC = 170002           477#   959
UBECLR= 170010           480#   963
UBECR1= 170006           479#   962   1392*  1393
UBECR2= 170016           482#   961
UBEDB = 170000           476#  1390   1398   1403   1408
UBEERR  053642          8809   8832#
UBEGO = 170014           481#
UBEINI  062214          7506   8825  10275#
UBEOPT= 000200           510#  1395
UBESAV  001722           776#  8789*  8790*  10275*  10276*
UBESET  043110          7493   7500#
UBESRV  053352          8782#  10284
UBETBL  002270           959#  1414   8784   8800*  8828*  8846   9301  10282  10774
UBEVEC= C00510           483#   964    965
UBE2    053470          8786   8804#
UBE3    053712          8833   8838   8844#
UBM6    015434          2973   2975*  2978   2986   2990*  2997*  2999*  3005*  3013*  3020*  3026*  3033*  3039*
                        3045*  3051*  3057*  3064*  3069*  3075   3082#
UBREAK= 177770           520#  1112*  1113   1116*  1117   1123*  1124   1130*  1131   1134   1138*  1153*  1155*
                        1157*  1159*  1172*  1178*  1183*  1543*  1552*  5892
UBRERR  002720          1114   1118   1125   1133#
UBRK2   002742          1132   1138#
UBRTBL  003276          1152   1174   1191#
UB7     021426          4048   4064#
UDPAR0= 177660           271#  6185
UDPAR1= 177662           272#
UDPAR2= 177664           273#
UDPAR3= 177666           274#
UDPAR4= 177670           275#
UDPAR5= 177672           276#
UDPAR6= 177674           277#
```

```
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#
UWM6    014606           2889#   2893*   2900*   2905*   2912*   2920*   2927*   2934*   2941*   2943*   2949*   2955*   2961*
UWM7    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    5244#
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#
XXDP    003570           1226#   1246
XXDPC   003571           1227#   1241*   1753
$AC0    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    9975
$AC1    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*   6609    6614*   6617    6622*
                         6624    6630    6636    6642*   6645*   6646    6653*   6658    6664*   6666*   6667    6678*   6689*
                         6703    6707    6720    6721*
$AC2    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*
```

```
                    6585    6597*   6606    6617*   6627    6633    6635*   6636*   6637*   6638*   6646
$AC3    001420       710#    6376*   6390    6402    6404    6454*   6463    6465    6546*   6560    6572    6574    6624*
                    6633    6635
$AC4    001422       711#
$AC5    001424       712#
$BDADR  001224       664#
$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#
$CORE   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#
$EOPCT  046426       1353*   8036#   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
$HINUM  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
$KTNEX  061774      10175    10201#
$KTOUT  061764      10192    10198#  10230
$KTTI   061624       1257*   10173#  10177*  10201*
$LF     001340        703#    9047    9403    10068   10077   10138
```

```
$LONUM  001550          741#  1220*  6806  6810*  6826  8086  8237  9874  9881  9887*  9916  10441
$LPADR  001210          657#  1357*  1815*  1933*  3108*  4174*  4726*  5102*  5528*  6015*  6339*  6779*  8974*
                       8988*  8995  8997
$LPERR  001212          658#  1135*  1358*  1813*  1814*  1815  1931*  1932*  1933  3106*  3107*  3108  4172*
                       4173*  4174  4724*  4725*  4726  5100*  5101*  5102  5526*  5527*  5528  6013*  6014*
                       6015  6337*  6338*  6339  6777*  6778*  6779  6808*  7677*  7822  7876*  8832  8833*
                       8844*  8890  8891*  8900*  8974  8989*  8997  9041  10629  10630*  10632*  10648  10705
                      10706*  10708*  10710  10719  10720*  10722*  10723  10734  10735*  10737*  10740
$LSTAD  062140         10216*  10231#
$LSTBK  062142          1259*  1260  1293  1303  10217*  10232#
$MAINT  001640           770#  7998*  8008*  9107
$MTMOU  062106         10193  10223#
$MXCNT  054620          8987  8997#
$NULL   001250           672#  9372  9403
$NWTST= 000001          1797#  1915#  1979#  2096#  2219#  2343#  2494#  2597#  2715#  2800#  2882#  2967#  3090#
                        3212#  3302#  3426#  3546#  3668#  3756#  3827#  3891#  3948#  4010#  4052#  4054  4116#
                        4118  4156#  4210#  4339#  4431#  4534#  4536  4603#  4659#  4708#  4829#  4917#  4968#
                        5035#  5084#  5256#  5258  5348#  5378#  5380  5444#  5510#  5626#  5628  5677#  5679
                        5762#  5764  5789#  5829#  5831  5885#  5887  5909#  5954#  5975#  5997#  6090#  6092
                        6131#  6133  6189#  6191  6251#  6253  6313#  6315  6502#  6504  6754#  6756  6800#
                        7414#  7484#  7486
$OCNT   057314          9530*  9559*  9572#
$OCTVL  057664          9657  9685#
$OMODE  057316          9525*  9529*  9534  9537*  9548*  9574#
$OVER   054576          8955  8975  8985  8992#
$PASS   001200           652#  1313*  1755  8033*  8034*  8045  8066  8981  8998  9119
$POWER  061216         10018  10025#
$PWRAD  061204         10020#
$PWRDN  061062          1351  9991#  10015
$PWRMG  061200         10018#
$PWRUP  061130         10000  10005#
$QUES   001336           701#  9047  10061  10077  10136  10138
$RAND   060436          6825  8084  8162  8235  8301  9870#  9906  10439
$RDCHR  061226         10039#  10263
$RDDEC  061400         10092#  10265
$RDLIN  061262         10053#  10264
$RDOCT= ****** U       10265
$RDSZ = 000010         10046#
$REGAD  001254           676#
$REG0   001256           678#  6350  6361  6383  6423  6430  6450  6483  6491  6492  6494  6520  6531
                        6553  6593  6600  6620  6653  6661  6662  6664  9955
$REG1   001260           679#  6351  6372  6382  6424  6432  6442  6484  6521  6542  6552  6594  6602
                        6612  6654
$REG10  001276           686#
$REG11  001300           687#
$REG2   001262           680#  6357*  6409  6439*  6470  6488*  6527*  6579  6609*  6640  6658*  11032  11045
$REG3   001264           681#  6397*  6411  6460*  6472  6492*  6567*  6581  6630*  6642  6662*  11032  11045
$REG4   001266           682#  10629*  10632  10705*  10708  10719*  10722  10734*  10737
$REG5   001270           683#  7428*  8004*  8014*  10535*  10543*  10553*  10585*  10586  10590*
$REG6   001272           684#
$REG7   001274           685#
$RESRE  057740          9720#  10267
$RTRN   001520           727#  1782  5440  8003
$SAVPA  001506           725#  10669  10684
$SAVPS  001516           726#  10658*  10694
$SAVRE  057702          9704#  10266
```

```
$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#
$SIZEX  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
$SWRMK= 000000          8950
$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#
```

```
$TRPAD  062164          10246  10257#
$TSTNM  001202            653#  1802*  1803  1920*  1921  1983*  1984  2100*  2101  2223*  2224  2347*  2348
                         2498*  2499  2601*  2602  2719*  2720  2804*  2805  2886*  2887  2971*  2972  3095*
                         3096  3216*  3217  3306*  3307  3430*  3431  3550*  3551  3672*  3673  3760*  3761
                         3831*  3832  3895*  3896  3952*  3953  4014*  4015  4057*  4058  4121*  4122  4161*
                         4162  4214*  4215  4343*  4344  4435*  4436  4540*  4541  4607*  4608  4663*  4664
                         4713*  4714  4833*  4834  4921*  4922  4972*  4973  5039*  5040  5089*  5090  5262*
                         5263  5352*  5353  5403*  5404  5448*  5449  5515*  5516  5633*  5634  5685*  5686
                         5769*  5770  5793*  5794  5837*  5838  5890*  5891  5913*  5914  5958*  5959  5979*
                         5980  6002*  6003  6096*  6097  6136*  6137  6196*  6197  6256*  6257  6325*  6326
                         6513*  6514  6766*  6767  6804*  6805  7418*  7419  7420  7489*  7490  7491  7531*
                         7953*  8031*  8946  8994*  8998  9015*  9018  9047  9111
$TTYIN  061370          10054  10055  10072  10076#
$TYPBN= ****** U        10263
$TYPDS  057320           9588#  10262
$TYPE   056264           9350#  10249  10258
$TYPEC  056430           9369  9376  9383  9388#  9391
$TYPEX  056504           9389  9396  9398  9401#
$TYPOC  057116           9528#  10259
$TYPON  057132           9527  9530#  10261
$TYPOS  057072           9523#  10260
$XTSTR  054400           8957#
$$GET4= 000000           8059#
$$TMP4  001426            713#
$$TMP6  001430            714#
$$TRP = 000002          10248#  10259  10260  10261  10262  10263  10264  10265  10266  10267  10268  10269  10270
$OFILL  057315           9524*  9528*  9538  9573#
.       = 071235          600#  604  606#  609#  635  636#  638#  640#  649#  655#  704  705#  715#
                          716#  718#  725#  771#  772#  773#  776#  777#  778#  893#  1194#  1203#  1212#
                         1267#  1271#  1284#  1292#  1343  1357  1358  1573#  1605#  1691#  1697#  1707  1769#
                         1811  1817  1929  1935  1946  1954  1962  1969  1977  1993  2003  2013  2021
                         2030  2039  2048  2057  2066  2072  2079  2086  2094  2120  2130  2140  2147
                         2154  2163  2179  2226  2236  2243  2252  2259  2265  2272  2279  2287  2294
                         2302  2307  2315  2321  2329  2336  2341  2349  2362  2368  2374  2380  2386
                         2392  2399  2406  2413  2421  2428  2432  2439  2444  2451  2458  2464  2468
                         2473  2479  2483  2489  2492  2500  2512  2517  2524  2531  2537  2542  2546
                         2554  2561  2568  2572  2578  2582  2588  2592  2595  2603  2617  2621  2625
                         2633  2641  2648  2656  2664  2670  2676  2682  2687  2692  2698  2706  2710
                         2713  2721  2734  2741  2747  2754  2762  2768  2774  2779  2785  2792  2798
                         2806  2824  2827  2833  2838  2846  2852  2857  2864  2868  2874  2880  2897
                         2903  2908  2916  2924  2930  2938  2946  2952  2958  2962  2965  2982  2987
                         2993  3002  3009  3016  3023  3029  3036  3042  3048  3054  3061  3066  3072
                         3078  3081  3104  3110  3130  3137  3145  3152  3159  3167  3174  3180  3187
                         3191  3198  3203  3210  3234  3242  3248  3255  3261  3267  3273  3280  3286
                         3293  3300  3312  3320  3326  3335  3343  3363  3371  3381  3389  3397  3400
                         3410  3415  3424  3432  3447  3456  3464  3473  3480  3487  3493  3501  3512
                         3519  3524  3531  3539  3544  3552  3570  3574  3581  3592  3597  3603  3608
                         3614  3617  3666  3686  3693  3698  3705  3716  3726  3733  3739  3747  3752
                         3762  3795  3803  3807  3813  3820  3825  3857  3865  3870  3873  3879  3884
                         3889  3924  3927  3933  3940  3946  3954  3967  3972  3976  3983  3989  3996
                         4002  4006  4030  4034  4037  4042  4046  4077  4080  4081  4087  4094  4102
                         4106  4110  4114  4147  4170  4176  4182  4188  4193  4197  4203  4208  4250
                         4255  4261  4265  4275  4285  4290  4326  4328*  4349  4351  4356  4361  4369
                         4371  4376  4392  4396  4410  4418  4429  4517  4548  4564  4589  4596  4614
                         4644  4645  4652  4668  4676  4679  4683  4688  4722  4728  4736  4811  4932
                         4949  4996  5003  5011  5017  5028  5049  5064  5073  5098  5104  5114  5127
```

```
                    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
                   10138   10575   10642#  10927#  10972#  11014#  11044#
.PARSR  063464      1791    7611    7984   10394   10609#  10636
```

```
COMMEN     1#    461#
ENDCOM     1#    461#
ERROR     79#    511   1136   4658   6417   6478   6500   6587   6648   6669   6795   6840   6854   6876   6902
          6918   6937   6956   6977   6999   7037   7082   7097   7114   7140   7190   7205   7665   7821   7879
          7884   7936   8837   8841   8895   8899  10403  10631  10707  10721  10736
ESCAPE     1#    461#
FLTST1  6313#   6315   6504
HLT      511#   1947   1955   1963   1970   1978   1994   2004   2014   2022   2031   2040   2049   2058   2067
          2073   2080   2087   2095   2121   2131   2141   2148   2155   2164   2180   2214   2237   2244   2253
          2260   2266   2273   2280   2288   2295   2303   2308   2316   2322   2330   2337   2342   2363   2369
          2375   2381   2387   2393   2400   2407   2414   2422   2429   2433   2440   2445   2452   2459   2465
          2469   2474   2480   2484   2490   2493   2513   2518   2525   2532   2538   2543   2547   2555   2562
          2569   2573   2579   2583   2589   2593   2596   2618   2622   2626   2634   2642   2649   2657   2665
          2671   2677   2683   2688   2693   2699   2707   2711   2714   2735   2742   2748   2755   2763   2769
          2775   2780   2786   2793   2799   2825   2828   2834   2839   2847   2853   2858   2865   2869   2875
          2881   2898   2904   2909   2917   2925   2931   2939   2947   2953   2959   2963   2966   2983   2988
          2994   3003   3010   3017   3024   3030   3037   3043   3049   3055   3062   3067   3073   3079   3131
          3138   3146   3153   3160   3168   3175   3181   3188   3192   3199   3204   3211   3235   3243   3249
          3256   3262   3268   3274   3281   3287   3294   3301   3313   3321   3327   3336   3344   3359   3364
          3372   3382   3390   3398   3401   3411   3416   3425   3448   3457   3465   3474   3481   3488   3494
          3502   3513   3520   3525   3532   3540   3545   3571   3582   3593   3598   3604   3609   3615   3667
          3687   3694   3699   3706   3717   3727   3734   3740   3748   3753   3796   3804   3808   3814   3821
          3826   3858   3866   3871   3874   3880   3885   3890   3925   3928   3934   3941   3947   3968   3973
          3978   3984   3990   3997   4003   4007   4031   4035   4038   4043   4047   4078   4082   4088   4095
          4103   4107   4111   4115   4183   4189   4194   4198   4204   4209   4251   4256   4262   4266   4276
          4286   4301   4303   4305   4307   4309   4311   4313   4315   4317   4319   4321   4323   4325   4327
          4329   4357   4377   4393   4411   4419   4430   4458   4483   4494   4506   4514   4518   4532   4556
          4597  4609#   4621   4646   4653  4658#   4678   4681   4685   4690   4775   4812   4856   4933   4950
          4955   4983   4997   5004   5012   5018   5029   5050   5065   5074   5115   5128   5142   5156   5170
          5187   5202   5211   5222   5228   5233   5244   5250   5255   5271   5290   5307   5311   5316   5322
          5326   5332   5341   5347   5368   5372   5376   5427   5432   5474   5477   5488   5500   5554   5577
          5597   5621   5657   5676   5708   5732   5756   5788   5808   5822   5857   5864   5906   5937   5949
          5965   5987   6030   6059   6079   6126   6166   6218   6243   6297   7469
MSGA     4052#   4054
MSGB     4116#   4118
MSGC     4533#   4536
MSGD     5256#   5258
MSGE     5378#   5380
MSGF     5626#   5628
MSGG     5677#   5679
MSGH     5762#   5764
MSGI     5829#   5831
MSGJ     5885#   5887
MSGK     6089#   6092
MSGL     6130#   6133
MSGM     6189#   6191
MSGN     6250#   6253
MSGO     7483#   7486
MSGP     6754#   6756
MULT       1#    461#
MYTAGS   642#    704
NEWTST     1#      2#   461#   1797   1915   1979   2096   2219   2343   2494   2597   2715   2800   2882   2967
          3090   3212   3302   3426   3546   3668   3756   3827   3891   3948   4010   4052   4116   4156   4210
          4339   4431   4533   4603   4659   4708   4829   4917   4968   5035   5084   5256   5348   5378   5443
          5510   5626   5677   5762   5789   5829   5885   5909   5954   5975   5997   6089   6130   6189   6250
          6313   6502   6754   6800   7414   7483
```

```
POP       1#   461#  9629  9725  9889 10009 10127
PUSH      1#   461#  9588  9705  9870  9993 10094
RELOCA 1086#  1804  1922  3097  4163  4715  5091  5517  6004  6328  6768
RELOCB 1086#  1908  3083  4149  4701  5077  5503  5989  6306  6741  7406
SCOPE    80#  1801  1908  1919  1982  2099  2222  2346  2497  2600  2718  2803  2885  2970  3083
         3094  3215  3305  3429  3549  3671  3759  3830  3894  3951  4013  4056  4120  4149  4160
         4213  4342  4434  4539  4606  4662  4701  4712  4832  4920  4971  5038  5077  5088  5261
         5351  5402  5447  5503  5514  5632  5684  5768  5792  5836  5889  5912  5957  5978  5990
         6001  6095  6135  6195  6255  6306  6324  6512  6741  6765  6803  7406
SETTRA 10249# 10259 10260 10261 10262 10263 10264 10265 10266 10267 10268 10269
SETUP     1#   461#  1339
SKIP      1#   461#  1746  1754  1756  1758  4557  6170  6499
SLASH     1#   461#
SPACE   461#
STARS     1#   461#   615   642   988  1090  1104  1216  1613  1622  1642  1654  1656  1676  1711
         1797  1799  1915  1917  1979  1981  2096  2098  2219  2221  2343  2345  2494  2496  2597
         2599  2715  2717  2800  2802  2882  2884  2967  2969  3090  3092  3212  3214  3302  3304
         3426  3428  3546  3548  3668  3670  3756  3758  3827  3829  3891  3893  3948  3950  4010
         4012  4052  4055  4116  4119  4156  4158  4210  4212  4339  4341  4431  4433  4534  4538
         4603  4605  4659  4661  4708  4710  4829  4831  4917  4919  4968  4970  5035  5037  5084
         5086  5256  5260  5348  5350  5378  5401  5444  5446  5510  5512  5626  5631  5677  5683
         5762  5767  5789  5791  5829  5835  5885  5888  5909  5911  5954  5956  5975  5977  5997
         5999  6090  6094  6131  6134  6189  6194  6251  6254  6313  6322  6418  6480  6502  6511
         6588  6650  6672  6677  6684  6688  6695  6701  6754  6763  6800  6802  7414  7416  7484
         7487  7498  7509  7525  7530  7613  7626  7670  7676  7976  7981  8019  8068  8071  8145
         8148  8218  8221  8285  8288  8340  8343  8457  8460  8584  8587  8683  8686  8778  8781
         8857  8860  8917  8923  8940  8998  9047  9074  9159  9168  9180  9187  9196  9206  9216
         9232  9240  9253  9268  9308  9328  9403  9409  9463  9469  9497  9575  9643  9686  9732
         9748  9805  9823  9858  9893  9901  9928  9945  9986 10028 10046 10077 10138 10233 10270
        10274 10295 10302 10324 10340 10365 10376 10407 10429 10484 10488 10495 10497 10502 10504
        10513 10526 10580 10584 10593 10608 10649 10654 10677 10680 10697 10699 10713 10715 10726
        10728 11058
TRMTRP 10249#
TYPBIN    1#   461#
TYPDEC    1#   461#  1606  8045  8052
TYPNAM    1#   461#  1566
TYPNUM    1#   461#
TYPOCS    1#   461#  1277  1285  1293
TYPOCT    1#   461#  1600  1692  9081  9107  9119
TYPTXT    1#   461#  1139  1143  1185  1200  1204  1209  1264  1268  1272  1281  1289  1595  1602
         1608  1688  1694  1760  7736  7745  8041  8048  9223 10639
$$CMRE   642#   678   679   680   681   682   683   684   685   686   687
$$CMTM   642#   688   689   690   691   692   693   694   695   696   697
$$ESCA    1#   461#
$$NEWT    1#     4#   461#  1797  1915  1979  2096  2219  2343  2494  2597  2715  2800  2882  2967
         3090  3212  3302  3426  3546  3668  3756  3827  3891  3948  4010  4052  4116  4156  4210
         4339  4431  4534  4603  4659  4708  4829  4917  4968  5035  5084  5256  5348  5378  5444
         5510  5626  5677  5762  5789  5829  5885  5909  5954  5975  5997  6090  6131  6189  6251
         6313  6502  6754  6800  7414  7484
$$SET  10249# 10259 10260 10261 10262 10263 10264 10265 10266 10267 10268 10269
$$SKIP    1#   461#  4557  6171  6499
.EQUAT    1#
.HEADE    1#    22
.KT11     1#
.SETUP    1#  1319
.SWRHI    1#    42
```

```
.SWRLO      1#     55#     58      61
.$ACT1      1#    615
.$CATC      1#    597
.$CMTA      1#      2#    642
.$DB2D      1#
.$DB2O      1#      6#   9643
.$DIV       1#
.$EOP       1#   8019
.$ERRO      1#   8998
.$ERRT      1#
.$MULT      1#
.$POWE      1#   9986
.$RAND      1#      7#   9858
.$RDDE      1#  10077
.$RDOC      1#
.$READ      1#  10028
.$SAVE      1#   9686
.$SB2D      1#
.$SB2O      1#
.$SCOP      1#      2#   8940
.$SIZE      1#  10138
.$SUPR      1#
.$TRAP      1#  10233
.$TYPB      1#
.$TYPD      1#   9575
.$TYPE      1#    642#   9328
.$TYPO      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)
```