PDP11

MAIN MEMORY CRAM TEST

MD-11-DZKCD-A

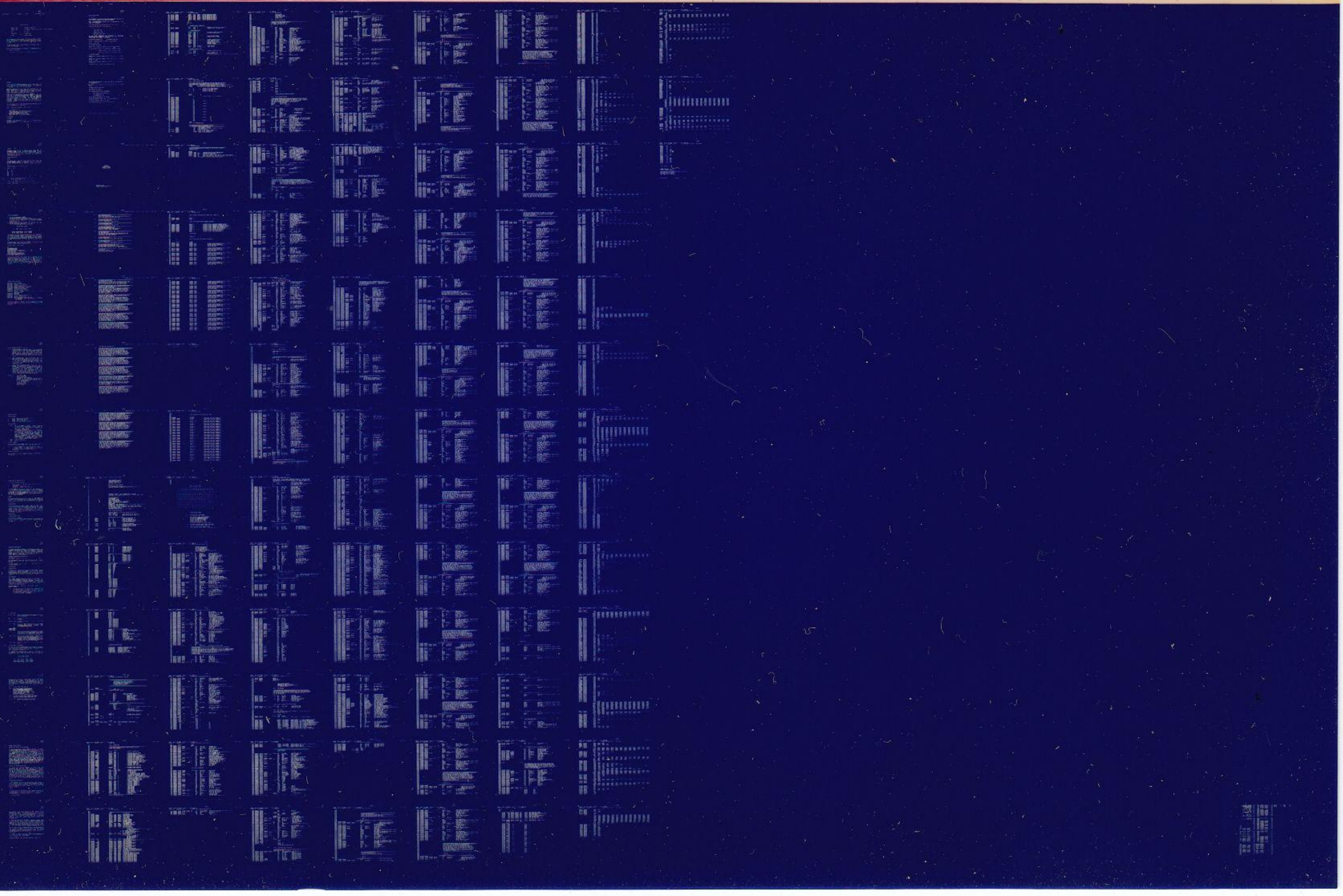
EP-DZKCD-A-DL-A

FICHE 1 OF 1

AUG 1977

digital

MADE IN USA



IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZKCD-A-D
PRODUCT NAME: MAIN MEMORY, JUMP AND CRAM TESTS ON MICRO-PROCESSOR
DATE: MAY 1977
MAINTAINER: DIAGNOSTICS
AUTHOR: DINESH GORADIA

1. ABSTRACT

The function of the KMC11 diagnostics is to verify that the option operates according to specifications. The diagnostics verfiy that there are no malfunctions and the all operations of the KMC11 are correct in its environment.

Parameters must be set up to alert the diagnostics to the KMC11 configuration. These parameters are contained in the STATUS TABLE and are generated in two ways: 1) Manual Input - the operator answers questions. 2) Autosizing - the program determines the parameters automatically.

DZKCE tests the KMC11-AR micro-processor (M8204-YA) with low speed cram, or the KMC11 micro-processor (M8204). It performs jump tests on the micro-processor, and tests the CRAM and other unique functions of the M8204. If a KMC11-AR (M8200-YA) and line unit (M8201) are present, free-running tests are performed. These tests are skipped if a KMC (M8204) or no line-unit is present. The best test is with a line-unit installed. DZKCE can be used as a Heat Test Diagnostic by Manufacturing.

Currently there are four off line diagnostics that are to be run in sequence to insure that if an error should occur it will be detected at an early stage.

NOTE: Additional diagnostics may be added in the future.

The four diagnostics are:

1. DZKCC [REV] Basic W/R and Micro-processor tests
2. DZKCD [REV] jump and main memory tests
3. DZKCE [REV] DDCMP Line unit tests
4. DZKCF [REV] BITSTUFF Line unit tests
5. DZKCA [REV] KMC11 CPU MICRO-DIAGNOSTICS.

2. REQUIREMENTS

2.1 EQUIPMENT

Any PDP11 family CPU (except an LSI-11) with minimum 8k memory
ASR 33 (or equilivalent)
KMC11-AR (M8200-YA) or an KMC11-A (M8204) with a KMC11-DA or a KMC11-FA

2.2 STORAGE

Program will use all 8K of memory except where ABL and BOOTSTRAP LOADER reside. Locations 2100 thru 2300; contain the "STATUS TABLE" information which is generated at start of diagnostics by manual input (questions) or automatically (auto-sizing). This area is an overlay area and should not be altered by the operator.

3. LOADING PROCEEDURE

3.1 METHOD

All programs are in absolute format and are loaded using the ABSOLUTE LOADER. NOTE: if the diagnostics are on a media such as DISK ,MAGTAPE,DECTAPE, or CASSETTE; follow instructions for the monitor which has been provided on that specific media.

ABSOLUTE LOADER starting address *500

MEMORY	* SIZE
4k	17
8k	37
12k	57
16k	77
20k	117
24k	137
28k	157

3.1.1 Place address of ABS loader into switch register.
(also place 'HALT' SW up)

3.1.2 Depress 'LOAD ADDRESS' key on console and release.

3.1.3 Depress 'START KEY' on console and release (program should now be loading into CPU)

4. STARTING PROCEEDURE

a. Set switch register to 000200
b Depress 'LOAD ADDRESS' key and release
c. Set SWR to zero for 'AUTO SIZING' or SWR bit0=1 for manual input (questions) or SWR bit7=1 to use existing parameters set up by a previous start or a previously run KMC11 diagnostic.
d. Depress 'START KEY' and release. The program will type Maindec Name and program name (if this was the first start up of the program) and also the following:

MAP OF KMC11 STATUS

PC	CSR	STAT1	STAT2	STAT3
002100	160010	045310	177777	000000
002110	160020	045320	177777	000000

The program will type 'R' and proceed to run the diagnostic. The above is only an example. This would indicate the status table starting at add. 2100 in the program. In this example the table contains the information and status of two KMC11'S. THE STATUS TABLE MUST BE VERIFIED BY THE USER IF AUTO SIZING IS DONE. For information of status table see section 8.4 for help.

If the diagnostic was started with SW00=1 indicating manual parameter input then the following shows an example of the questions asked and some example answers:

```
HOW MANY KMC11'S TO BE TESTED?1

01
CSR ADDRESS?160010
VECTOR ADDRESS?310
BR PRIORITY LEVEL?  (4,5,6,7)?5
WHICH LINE UNIT?  IF NONE TYPE "N",  IF  M8201  TYPE  "1",  IF
M8202 TYPE "2"?1
IS THE LOOP BACK CONNECTOR ON?Y
SWITCH PAC#1 (DDCMP LINE#)?377
SWITCH PAC#2 (BM873 BOOT ADD)?377
```

Following the questions the status map is printed out as described above, the information in the map reflects the answers to the questions. If the diagnostic was started with SW00=0 and SW07=0 (AUTO-SIZING) then no questions are asked and only the status-map is printed out. If AUTO-SIZING is used the status information must be verified to be correct (match the hardware). if it does not match the hardware the diagnostic must be restarted with SW00=1 and the questions answered.

4.1 CONTROL SWITCH SETTINGS

SW 15 Set: Halt on error
SW 14 Set: Loop on current test
SW 13 Set: Inhibit error print out
SW 12 Set: Inhibit type out abell on error.
SW 11 Set: Inhibit iterations. (quick pass)
SW 10 Set: Escape to next test on error
SW 09 Set: Loop with current data
SW 08 Set: Catch error and loop on it
SW 07 Set: Use previous status table.
SW 06 Set: Halt in ROMCLK routine before clocking micro-processor
SW 05 Set: Reserved
SW 04 Set: Reserved
SW 03 Set: Reselect KMC11's desired active
SW 02 Set: Lock on selected test
SW 01 Set: Restart program at selected test
SW 00 Set: Build new status table from questions. (If SW07=0 and SW00=0 a new status table is built by auto-sizing)

Switch 06 and 08-15 are dynamic and can be changed as needed while the diagnostic is running. Switches 00-03 and switch 07 are static, and are used only on starting or restarting the diagnostic.

4.1.2 SWITCH REGISTER OPTIONS (at start up)

SW 01 RESTART PROGRAM AT SELECTED TEST. It is strongly suggested that at least one pass has been made before trying to select a test, the reason being is that the program has to clear areas and set up parameters. When this switch is used the diagnostic will ask TEST NO.? Answer by typing the number of the test desired and carrige return to begin execution at the selected test.

SW 02 LOCK ON SELECTED TEST. This switch when used with SW01 will cause the program to constantly loop on the selected test. Hitting any key on the console will let it advance to the next test and loop until a key is hit again. If SW02=0 when SW01 is used. The program will begin at the selected test and continue normal operations.

SW 03 RESELECT KMC11'S DESIRED ACTIVE. Please note that a message is typed out for setting the switch register equal to KMC11's active. this means if the system has four KMC11s; bits 00,01,02,03 will be set in loc 'KMACTV' from the switch register. Using this switch(SW00) alters that location;therefore if four KMC11s are in the system ***DO NOT*** set switchs greater than SW 03 in the up position. this would be a fatal error. do not select more active KMC11s than there is information on in the status table.

METHOD: A: Load address 200
B: Start with SW 00=1
C: Program will type message
D: Set a switch for each KMC desired active. EXAMPLE: If you have 4 KMC's but only want to run the first and the last set SWR bits 0 and 3 = 1. PRESS CONTINUE
E: Number (IF VALID) will be in data lights (excluding 11/05)
F: Set with any other switch settings desired. PRESS CONTINUE.

4.1.3 DYNAMIC SWITCHES

ERROR SWITCHES

1. SW 12 Delete print out/bell on error.
2. SW 13 Delete error printout.
3. SW 15 Halt on the error.
4. SW 08 Goto beginning of the test(on error).
5. SW 10 Goto next test(on error).

SCOPE SWITCHES

1. SW06 Halt in ROMCLK routine before clocking micro-processor instruction. This allows the operator to scope a micro-processor instruction in the static state before it is clocked. Hit continue to resume running.
2. SW09 (if enabled by 'SCOPI') on an error; If an '*' is printed in front of the test no. (ex. *TEST NO. 10) SW09 is incorporated in that test and therefore SW09 is usually the best switch for the scope loop (SW14=0, SW10=0, SW09=1, SW08=0). If SW09 is not enabeled; and there is a HARD error (constant); SW08 is best. (SW14=1,0, SW10=0, SW09=0, SW08=1). for intermittemt errors; SW14=1 will loop on test reguardless of error or not error. (SW14=1, SW10=0, SW09=0, SW08=1,0)
3. SW11 Inhibit interations.
4. SW14 Loop on current test.

4.2 STARTING ADDRESS

Starting address is at 000200 there are no other starting addresses for the KMC11 diagnostics. (See Section 4.0)

NOTE: If address 000042 is non-zero the program assumes it is under ACT11 or XXDP control and will act accordingly after all available KMC11's are tested the program will return to 'XXDP' or 'ACT-11'.

5. OPERATING PROCEDURE

When program is initially started messages as described in section 4.0 will be printed, and program will begin running the diagnostic

5.2 PROGRAM AND/OR OPERATOR ACTION

The typical approach should be

1. Halt on error (via SW 15=1) when ever an error occurs.
2. Clear SW 15.
3. Set SW 14: (loop on this test)
4. Set SW 13: (inhibit error print out)

The TEST NUMBER and PC will be typed out and possibily an error message (this depends on the test) to give the operator an idea as to the source of the problem. If it is necessary to know more information concerning the error report; LOOK IN THE LISTING for that TEST NUMBER which was typed out and then NOTE THE PC of thE ERROR REPORT this way the EXACT FUNCTION of the test CAN BE DETERMINED.

6. ERRORS

As described previously there will always be a TEST NUMBER and PC typed out at the time of an error (providing SW 13=0 and SW 12=0). in most cases additional information will be supplied in the the error message to give the operator an indication of the error.

6.2 ERROR RECOVERY

If for some reason the KMC11 should 'HANG THE BUS' (gain control of bus so that console manual functions are inhibited) an init or power down/up is necessary for operator to regain control of cpu. If this should happen; look in location '$TSTNM' (address 1202)for the number of the test that was running at the time of the catastrophic error. In this way the operator will have an idea as to what the KMC11 was doing at the time of the error.

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

See section 4. (PLEASE)
Status table should be verified reguardless of how program was started. Also it is important to use this listing along with the information printed on the TTY to completly isolate problems.

7.2 OPERATING RESTRICTIONS

The first time a KMC11 diagnostic is loaded into core and run the STATUS TABLE must be set up. This is done by manual input (SW00=1) or by autosizing (SW00=0 and SW07=0). Thereafter however the status table need not be setup by subsequent restarts or even loading the next KMC diagnostic because the STATUS TABLE is overlayed. The current parameters in the STATUS TABLE are used when SW07=1 on start up.

7.3 HARDWARE CONFIGURATION RESTRICTIONS

KMC11(M8204)- Jumper W1 must be in,

LINE UNIT(M8201)- Jumpers W1, W2, and W4 must be IN. Jumpers W3, and W5 must be OUT. SW8 of E26 must be in the ON POSITION.

LINE UNIT (M8202)- Jumper W1 must be in. SW8 of E26 must be in the OFF position.

8. MISCELLANEOUS

8.1 EXECUTION TIME

All KMC11 device diagnostics will give an 'END PASS' message (providing no errors and sw12=0) within 4 mins. This is assuming SW11=1 (DELETE ITERATIONS) is set to give the fastest possible execution. The actual execution time depends greatly on the PDP11 CPU configuration and the amount of memory in the system.

8.2 PASS COMPLETE

NOTE: EVERY time the program is started; the tests will run as if SW11 (delete iterations) was up (=1). This is to 'VERIFY NO HARD ERRORS' as soon as possible. Therefore the first pass -EACH TIME PROGRAM IS STARTED- will be a 'QUICK PASS' until all KMC11's in system are tested. When the diagnostic has completed a pass the following is an example of the print out to be expected.

END PASS DZKCD CSR: 175000 VEC: 0300 PASSES: 000001
ERRORS: 000000

NOTE: The pass count and error counts are cummulitive for each KMC11 that is running, and are set to zero only when the diagnostic is started. Therefore after an overnight run for example, the total passes and errors for each KMC11 since the diagnostic was started are reflected in PASSES: and ERRORS:.

8.4 KEY LOCATIONS

$Ipadr (1206) Contains the address where program will return when iteration count is reached or if loop on test is asserted.

NEXT (1442) Contains the address of the next test to be peformed.

STSTNM (1202) Contains the number of the test now being peformed.

RUN (1500) The bit in 'RUN' always points to the KMC11 currently being tested. EXAMPLE: (RUN) 1500/0000000001000000 Means that KMC11 no.06 is the KMC11 now running.

KMCR00-KMCR17
KMST00-KMST17
(2100)-(2300)

These locations contain the information needed to test up to 16 (decimal) KMC11s sequentially, they contain the CSR,VECTOR and STATUS concerning the configuration of each KMC11.

KMACTV (1470) Each bit set in this location indicates that the associated KMC11 will be tested in turn. EXAMPLE: (KMACTV) 1470/0000000000011111 means that KMC11 no. 00,01,02,03,04 will be tested. EXAMPLE: (KMACTV) 1470/0000000000010001 Means that KMC11 no. 00,04 will be tested.

KMCSR (2066) Contains the CSR of the current KMC11 under test.

8.4A 'STATUS TABLE' (2100-2300)

The table is filled by AUTO SIZING or by the manual parameter input (questions) as described previously. Also if desired by user; the locations may be altered by hand (toggled in) to suit the specific configuration.

The example status map shown below contains information for two KMC11'S. the table can contain up to 16 KMC11'S. Following the map is a description of the bits for each map entry

MAP OF KMC11 STATUS

PC	CSR	STAT1	STAT2	STAT3
002100	160010	045310	177777	000000
002110	160020	016320	000000	000000

Each map entry contains 4 words which contain the status information for 1 KMC11. The PC shows where in core memory the first of the 4 words is. In the example above the first KMC'S status is in locations, 2100, 2102, 2104, and 2106. The second KMC status is located at 2110, 2112, 2114, and 2116. The information contained in each 4 word entry is defined as follows:

CSR: Contains KMC11 CSR address

STAT1: BITS 00-08 IS KMC11 VECTOR ADDRESS
BIT14=1 TURNAROUND CONNECTOR IS ON
BIT14=0 NO TURNAROUND CONNECTOR
BIT13=0 LINE UNIT IS AN M8201
BIT13=1 LINE UNIT IS AN M8202
BIT12=1 NO LINE UNIT
BITS 09-11 IS KMC11 BR PRIORITY LEVEL

STAT2: LOW BYTE IS SWITCH PAC#1 (DDCMP LINE NUMBER)
HIGH BYTE IS SWITCH PAC#2 (BM873 BOOT ADD)

STAT3: BIT0=1 PERFORM FREE RUNNING TESTS ON KMC
(must be set manually. SEE TEST 50

8.5 METHOD OF AUTO SIZING

8.5.1 FINDING THE CONTROL STATUS REGISTER.

The auto-sizing routine finds a KMC11 as follows: It starts at address 160000 and tests all address in increments of 10 up to and including address 167760. If the address does not time out, the following is done, the first CRAM address is written to a 125252 then it is read back. If it contains a -1 or 125252 KMC11 has been found, if not, the address is updated by 10 and the search continues. A -1 indicates a KMC11 with no CRAM, and a 125252 indicates a KMC11 with CRAM Further tests are performed at this point to determine which line unit, if any, is installed, if a loop-back connector is installed and various switch settings on the line unit. THIS IS WHY THE STATUS TABLE MUST BE VERIFIED BY THE USER AND IF ANY OF THE INFORMATION DOES NOT AGREE WITH THE HARDWARE THE DIAGNOSTIC MUST BE RESTARTED AND THE QUESTIONS MUST BE ANSWERED. All KMC11's in the system will be found by the auto-sizer. If it does not find a KMC11 the diagnostic must be restarted and the questions answered.

8.5.2 FINDING THE VECTOR AND BR LEVEL

The vector area (address 300-776) is filled with the instruction IOT and '.+2' (next address). The processor status is started at 7 and the KMC is programmed to interrupt. The PS is lowered by 1 until the KMC interrupts, a delay is made and if no interupt occures at PS level 3 (because of a bad KMC11) the program assumes vector address 300 at BR level 5 and the problem should be fixed in the diagnostic. Once the problem is fixed; the program should be re-setup again to get correct vector. If an interupt occured; the address to which the KMC11 interupted to is picked up and reported as the vector. NOTE: if the vector reported is not the vector set up by you; there is a problem and AUTO SIZING should not be done.

8.6 SOFTWARE SWITCH REGISTER

If the diagnostic is run on an 11/04 or other CPU without a switch register then a software switch register is used to allow user the same switch options as described previously. If the hardware switch register does not exist or if one does and it contains all ones (177777) this software switch register is used.

Control:

To obtain control at any allowable time during execution of the diagnostic the operator types a CTRL G on the console terminal keyboard. As soon as the CTRL G is recognized, by the diagnostic, the following message will be displayed:

```
SWR=XXXXXX NEW?
```

Where XXXXXX is the current contents of the software switch register in octal. The software control routine will then await operator action. At which time the operator is required to type one or more of the legal characters: 1) 0 - 7, 2) line feed(<LF>), 3) carriage return(<CR>), or 4) control-U (CTRL U). No check is made for legality. If the input character is not a <LF>, <CR>, or CTRL U it is assumed to be an octal digit.

To change the contents of the SSR the operator simply types the new desired value in octal - leading zeros need not be typed. And terminates the input string with a <CR> or <LF> depending on the program action desired as described below. The input value will be truncated to the last 6 digits typed. At least one digit must be typed on any given input string prior to the terminator before a change to the SSR will occur.

When the input string is terminated with a <CR> the diagnostic will continue execution from the point at which it was interrupted. If a <CR> is the only thing typed the program will continue without changing the SSR. The <LF> differs from the <CR> by restarting the program as if it were restarted at address 200.

If a CTRL U is typed at any point in the input string prior to the terminator the input value will be disregarded and the prompt displayed (SWR = XXXXXX NEW?).

To set the SSR for the starting switches, first load the diagnostic, then hit CTRL G, then start the diagnostic.

APT/ACT/XXDP/SLIDE

++++++++++++

THIS DIAGNOSTIC IS APT/ACT/XXDP/SLIDE COMPATIBLE USER WOULD BE ABLE TO RUN IT UNDER APT/ACT/XXDP ENVIRONMENT.

NOTE: FOR MANUFACTURING PURPOSE ONLY ITS DESCRIBED HOW TO RUN UNDER APT ENVIRONMENT.

****

ETABLE SETTING FOR APT TO RUN UNDER APT

++++++++++++++++++++

FIRST PASS TIME:

LONGEST TEST TIME:

ADDITIONAL TEST TIME:

ALL THE ABOVE PARAMETERS ARE DEPENDENT ON PARTICULAR DIAGNOSTICS AND SHOULD BE LOADED AT THE TIME OF SETTING ETABLE.THERE IS NO DEFAULT TIME SET UP.

SOFTWARE ENVIRONMENT:001 ENVIRONMENT MODE:200

SWITCH 1:-SHOULD BE USED AS NORMAL SWITCH REGISTER.

SWITCH 2:-NOT USED.

CPU OPTIONS:-NOT USED.

MEMORY TYPE 1:-BITS<2:4>:=BITS <12:14> OF STAT1 OF DEV:0.

MAXIMUM ADDRESS:-BITS<17:19>:=BITS<12:14> OF STAT1 OF DEV:1

BITS<2:4>:=BITS <12:14> OF STAT1 OF DEV:2

BITS<10:12>:=BITS<12:14> OF STAT1 OF DEV:3

IN THE SAME MANNER

MEMORY TYPE 2 MAXIMUM ADDRESS:-GETS STAT1<12:14> OF DEVICE 4,5,6,7.

MEMORY TYPE 3 MAXIMUM ADDRESS:-GETS STAT1<12:14> OF DEVICE 8,9,10,11.

MEMORY TYPE 4 MAXIMUM ADDRESS:-GETS STAT1<12:14> OF DEVICE 12,13,14,15.

INTERRUPT VECTOR 1:FIRST DEVICE RECEIVE VECTOR.

REST OF THE DEVICE(KMC'S) VECTOR SHOULD BE SET UP SEQUENTIALLY
IN INCREMENTS OF 10.

BUS PRIORITY:KMC'S PRIORITY(SHOULD BE SAME FOR ALL KMC'S UNDER
TEST).

INTERRUPT VECTOR 2:NOT USED.

BUS PRIORITY:NOT USED.

BASE ADDRESS:FIRST DEVICE CSR ADDRESS.

REST SHOULD FOLLOW SEQUENTIALLY

IN INCREMENTS OF 10.

DEVICE MAP:AS DESCRIBED IN APT MANUAL.

CONTROLLER SPECIFIC CODE 1:-NO. OF DEVICES UNDER TEST.

CONTROLLER SPECIFIC CODE 2:-NOT USED.

DEVICE DESCRIPTOR WORD 0:STAT2 OF FIRST DEVICE.

. .

. .

TO

. .

. .

DEVICE DESCRIPTOR WORD 15:STAT2 OF 16TH DEVICE.(KMC)

DOCUMENT
****
MAINDEC-11-DZKCD
****

```
2191    ********** TEST 1 **********
        TEST OF BR RIGHT SHIFT
        VERIFY THAT A DEST OF BR RSH (011) OF A MICRO-INSTRUCTION
        SHIFTS THE RESULTING BR DATA RIGHT ONCE.

2233    ********** TEST 2 **********
        IOP CRAM WRITE/READ TEST
        FLOAT A 1 THROUGH EACH CRAM LOCATION

2267    ********** TEST 3 **********
        IOP CRAM WRITE/READ TEST
        FLOAT A 0 THROUGH EACH CRAM LOCATION

2304    ********** TEST 4 **********
        IOP CRAM DUAL ADDRESSING TEST
        WRITE EACH ADDRESS INTO ITSELF,READ EACH
        ADDRESS TO VERIFY CORRECT ADDRESSING

2350    ********** TEST 5 **********
        IOP CRAM READ TEST
        THIS TEST WRITES THE CRAM WITH THE CROM MICRO-CODE MAP
        THEN READS IT BACK AND COMPARES EACH ADDRESS WITH THE
        DUPLICATE OF THE CROM MICRO-CODE.

2387    ********** TEST 6 **********
        IOP MAIN MEMORY TEST
        FLOAT A 1 THROUGH ALL MAIN MEMORY LOCATIONS

2433    ********** TEST 7 **********
        IOP MAIN MEMORY TEST
        FLOAT A 0 THROUGH ALL MAIN MEMORY LOCATIONS

2481    ********** TEST 10 **********
        IOP MAIN MEMORY DUAL ADDRESSING TEST
        LOAD EACH MEMORY LOCATION WITH ITS OWN ADDRESS
        READ BACK EACH LOCATION TO VERIFY CORRECT ADDRESSING

2549    ********** TEST 11 **********
        IOP MAR TEST
        PERFORM DUAL ADDRESSING TEST
        USING MAR AUTO-INC FEATURE
```

```
2589    ************************** TEST 12 ****************************
        IOP (CRAM) ODT BITS TEST
        LOAD MAR WITH A 0 INC MAR UNTIL IT OVERFLOWS (2000 TIMES)
        VERIFY THAT IBUS* 10 BITS IS SET ONLY WHEN MAR BIT 8 IS A ONE
        AND THAT IBUS* 10 BIT6 IS SET ON MAR OVERFLOW(2000)

2650    ************************** TEST 13 ****************************
        CRAM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
        PERFORM THE JUMP INSTRUCTION
        VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
        THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
        THEN PORT4 CONTAINS A 37

2711    ************************** TEST 14 ****************************
        CRAM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
        PERFORM THE JUMP INSTRUCTION
        VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
        THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
        THEN PORT4 WILL CONTAIN A 37

2769    ************************** TEST 15 ****************************
        CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
        SET THE C BIT, PERFORM THE JUMP INSTRUCTION,
        VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
        THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
        THEN PORT4 WILL CONTAIN A 37

2830    ************************** TEST 16 ****************************
        CRAM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
        SET THE Z BIT, PERFORM THE JUMP INSTRUCTION,
        VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
        THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
        THEN PORT4 WILL CONTAIN A 37

2891    ************************** TEST 17 ****************************
        CRAM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
        SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION,
        VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
        THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
```

THEN PORT4 WILL CONTAIN A 37

2952 ********** TEST 20 **********
CRAM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3013 ********** TEST 21 **********
CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3074 ********** TEST 22 **********
CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3135 ********** TEST 23 **********
CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE

3140 BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
THEN PORT4 CONTAINS A 37

3196 ********** TEST 24 **********
CRAM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
THEN PORT4 CONTAINS A 37

```
3257    ********** TEST 25 **********
        CRAM TEST OF JUMP(I) ON BR0 SET MICRO-PROCESSOR INSTRUCTION.
        CLEAR THE BR0 BIT, PERFORM THE JUMP INSTRUCTION,
        VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
        THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
        THEN PORT4 CONTAINS A 37

3318    ********** TEST 26 **********
        CRAM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
        CLEAR THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
        VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
        THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
        THEN PORT4 CONTAINS A 37

3379    ********** TEST 27 **********
        CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
        CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
        VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
        THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
        THEN PORT4 CONTAINS A 37

3440    ********** TEST 30 **********
        CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
        CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
        VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
        IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
        BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
        THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
        THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
        THEN PORT4 CONTAINS A 37
```

```
                        .TITLE  MAINDEC-11-DZKCD
                        ;*COPYRIGHT (C) 1976
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DINESH GORADIA
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                        ;*
                        ;*MAINDEC-11-DZKCD   KMC11 REMOTE CROM, JUMP TESTS
                        ;*COPYRIGHT 1976, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754
                        ;*----------------------------------------------------------------

                        ;STARTING PROCEDURE
                        ;LOAD PROGRAM
                        ;LOAD ADDRESS 000200
                        ;SWR=0  AUTOSIZE KMC11
                        ;SW07=1  USE CURRENT KMC11 PARAMETERS
                        ;SW00=1  INPUT NEW KMC11 PARAMETERS
                        ;PRESS START
                        ;PROGRAM WILL TYPE "MAINDEC-11-DZKCD   KMC11 REMOTE CROM, JUMP TESTS"
                        ;PROGRAM WILL TYPE STATUS MAP
                        ;PROGRAM WILL TYPE "R" TO INDICATE THAT TESTING HAS STARTED
                        ;AT THE END OF A PASS, PROGRAM WILL TYPE PASS COMPLETE MESSAGE
                        ;AND THEN RESUME TESTING
                        ;SUBSEQUENT RESTARTS WILL NOT TYPE PROGRAM TITLE

                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
001200                  STACK=  1200
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE FOR LINE FEED
000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
```

```
                        ;
                        ;R3=     %3             ;;GENERAL REGISTER
                        ;R4=     %4             ;;GENERAL REGISTER
                        ;R5=     %5             ;;GENERAL REGISTER
                        ;R6=     %6             ;;GENERAL REGISTER
                        ;R7=     %7             ;;GENERAL REGISTER
                        ;SP=     %6             ;;STACK POINTER
                        ;PC=     %7             ;;PROGRAM COUNTER
                        ;
                        ;*PRIORITY LEVEL DEFINITIONS
                        ;PR0=    0              ;;PRIORITY LEVEL 0
                        ;PR1=    40             ;;PRIORITY LEVEL 1
                        ;PR2=    100            ;;PRIORITY LEVEL 2
                        ;PR3=    140            ;;PRIORITY LEVEL 3
                        ;PR4=    200            ;;PRIORITY LEVEL 4
                        ;PR5=    240            ;;PRIORITY LEVEL 5
                        ;PR6=    300            ;;PRIORITY LEVEL 6
                        ;PR7=    340            ;;PRIORITY LEVEL 7
                        ;
                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  ;SW15=   100000
040000                  ;SW14=   40000
020000                  ;SW13=   20000
010000                  ;SW12=   10000
004000                  ;SW11=   4000
002000                  ;SW10=   2000
001000                  ;SW09=   1000
000400                  ;SW08=   400
000200                  ;SW07=   200
000100                  ;SW06=   100
000040                  ;SW05=   40
000020                  ;SW04=   20
000010                  ;SW03=   10
000004                  ;SW02=   4
000002                  ;SW01=   2
000001                  ;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)
100000                  ;BIT15=  100000
040000                  ;BIT14=  40000
020000                  ;BIT13=  20000
010000                  ;BIT12=  10000
004000                  ;BIT11=  4000
002000                  ;BIT10=  2000
001000                  ;BIT09=  1000
000400                  ;BIT08=  400
000200                  ;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
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC=34              ;;"TRAP" TRAP
000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR



                        ;INSTRUCTION DEFINITIONS
                        ;-----------------------

005746                  PUSH1SP=5746    ;DECREMENT PROCESSOR STACK 1 WORD
005726                  POP1SP=5726     ;INCREMENT PROCESSOR STACK 1 WORD
010046                  PUSHR0=10046    ;SAVE R0 ON STACK
012600                  POPR0=12600     ;RESTORE R0 FROM STACK
024646                  PUSH2SP=24646   ;DECREMENT STACK TWICE
022626                  POP2SP=22626    ;INCREMENT STACK TWICE
                        .EQUIV  EMT,HLT ;BASIC DEFINITION OF ERROR CALL
```

```
                                ;:**********************************************************************
                                ;----------------------------------------------------------------------
                                        ;TRAPCATCAER FOR ILLEGAL INTERRUPTS
                                        ;THE STANDARD "TRAP CATCHER" IS PLACED
                                        ;BETWEEN ADDRESS 0 TO ADDRESS 776.
                                        ;IT LOOKS LIKE "PC+2 HALT".
                                ;----------------------------------------------------------------------
                                ;:**********************************************************************

000000                          .=0
000000  000000  000000                  .WORD   0,0
                                ;STANDARD INTERRUPT VECTORS
                                ;-------------------------

000020                          .=20
000020  004134                          $SCOPE          ; SCOPE LOOP HANDLER.
000022  000340                          PR7             ; SERVICE AT LEVEL 7.
000024  007126                          $PWRDN                  ;POWER FAIL HANDLER
000026  000340                          PR7                     ;SERVICE AT LEVEL 7
000030  006512                          $ERROR                  ;ERROR HANDLER
000032  000340                          PR7                     ;SERVICE AT LEVEL 7
000034  006414                          $TRAP                   ;GENERAL HANDLER DISPATCH SERVICE
000036  000340                          PR7                     ;SERVICE AT LEVEL 7
                                .SBTTL  ACT11 HOOKS

                                ;;**********************************************************************
                                ;HOOKS REQUIRED BY ACT11
000040                                  $SVPC=.                 ;SAVE PC
000046                                  .=46
000046  004070                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052                                  .=52
000052  040000                          .WORD   BIT14           ;;2)SET LOC.52 TO BIT14
000040                                  .=$SVPC                 ;; RESTORE PC
                                                                ;BIT14=1 PROGRAM EXECUTION TIME
                                                                ;IS MEMORY SIZE DEPENDENT

000174                          .=174
000174  000000                 DISPREG:0                       ;SOFTWARE DISPLAY REGISTER
000176  000000                 SWREG:  0                       ;SOFTWARE SWITCH REGISTER

000200                          .=200
000200  000137  002402                  JMP     .START          ;GO TO START OF PROGRAM


001000                          .=1000
001000  005200  040515  047111  MTITLE: .ASCII  <200><12>/MAINDEC-11-DZKCD/<200>
(2) 001023     113  041515  030461          .ASCIZ  /KMC11 REMOTE CROM, JUMP TESTS/<200>
(2)
177570                          DSWR    =       177570
177570                          DDISP   =       177570
```

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

                        001200          .=1200
001200                  $CMTAG:                         ;;START OF COMMON TAGS
001200  000000                  .WORD   0
001202      000         $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001203      000         $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001204  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001206  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001210  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001212  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001214      000         $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001215      001         $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001216  000000          $ERRPC  .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001220  000000          $GDADR  .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001222  000000          $BDADR  .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001224  000000          $GDDAT. .WORD   0               ;;CONTAINS 'GOOD' DATA
001226  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001230  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001232  000000                  .WORD   0
001234      000         $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001235      000         $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001236  000000                  .WORD   0
001240  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001242  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001244  177560          $TKS:   177560                  ;;TTY KBD STATUS
001246  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001250  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001252  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001254      000         $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001255      002         $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001256      012         $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001257      000         $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001260  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001262  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001264  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001266  000000          $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001270  000000          $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001272  000000          $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001274  000000          $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001276  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001300  000000          $TMP1:  .WORD   0               ;;USER DEFINED
001302  000000          $TMP2:  .WORD   0               ;;USER DEFINED
001304  000000          $TMP3:  .WORD   0               ;;USER DEFINED
001306  000000          $TMP4:  .WORD   0               ;;USER DEFINED
001310  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001312      077         $QUES:  .ASCII  /?/             ;;QUESTION MARK
001313      015         $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001314  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;**********************************************************************
                                .SBTTL  APT MAILBOX-ETABLE
```

```
                                ;;*****************************************************************
                                ;EVEN
001316                          $MAIL:                  ;;APT MAILBOX
001316  000000                  $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
001320  000000                  $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
001322  000000                  $TESTN: .WORD   ATESTN  ;;TEST NUMBER
001324  000000                  $PASS:  .WORD   APASS   ;;PASS COUNT
001326  000000                  $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
001330  000000                  $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
001332  000000                  $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
001334  000000                  $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001336                          $ETABLE:                ;;APT ENVIRONMENT TABLE
001336     002                  $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001337     000                  $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001340  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001342  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001344  000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                                ;*                      BITS 15-11=CPU TYPE
                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                              11/70=06,PDQ=07,Q=10
                                ;*                      BIT 10=REAL TIME CLOCK
                                ;*                      BIT  9=FLOATING POINT PROCESSOR
                                ;*                      BIT  8=MEMORY MANAGEMENT
001346     000                  $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001347     000                  $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                              900 NSEC CORE=001
                                ;*                              300 NSEC BIPOLAR=002
                                ;*                              500 NSEC MOS=003
001350  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001352     000                  $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001353     000                  $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001354  000000                  $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001356     000                  $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001357     000                  $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001360  000000                  $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001362     000                  $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001363     000                  $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001364  000000                  $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001366  000000                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001370  000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001372  000000                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001374  000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
001376  000000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001400  000000                  $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001402  000000                  $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001404  000000                  $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001406  000000                  $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001410  000000                  $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001412  000000                  $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001414  000000                  $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001416  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001420  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001422  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
```

```
000322  001424  000000                  $DQW9:  .WORD   ADQW9   ;;DEVICE DESCRIPTOR WORD#9
000323  001426  000000                  $DQW10: .WORD   ADQW10  ;;DEVICE DESCRIPTOR WORD#10
000324  001430  000000                  $DQW11: .WORD   ADQW11  ;;DEVICE DESCRIPTOR WORD#11
000325  001432  000000                  $DQW12: .WORD   ADQW12  ;;DEVICE DESCRIPTOR WORD#12
000326  001434  000000                  $DQW13: .WORD   ADQW13  ;;DEVICE DESCRIPTOR WORD#13
000327  001436  000000                  $DQW14: .WORD   ADQW14  ;;DEVICE DESCRIPTOR WORD#14
000328  001440  000000                  $DQW15: .WORD   ADQW15  ;;DEVICE DESCRIPTOR WORD#15
000329
000330
000331  001442                          $ETEND:
000332
000333                                  ;
000334                                  ;  PROGRAM CONTROL PARAMETERS
000335                                  ;-------------------------------
000336  001442  000000                  NEXT:   .WORD   0               ; ADDRSS OF NEXT TEST TO BE EXECUTED
000337  001444  000000                  LOCK:   .WORD   0               ; ADDRESS FOR LOCK CURRENT DATA
000338                                  ;
000339                                  ;   PROGRAM VARIABLES
000340                                  ;----------------------
000341  001446  000000                  STRTSW: .WORD   0               ; SWITCHES AT START OF PROGRAM
000342  001450  000000                  STAT:   .WORD   0               ; KM STATUS WORD STORAGE
000343  001452  000000                  CLKX:   .WORD   0               ;
000344  001454  000000                  MASKX:  .WORD   0               ;
000345  001456  000000                  SAVSP:  .WORD   0               ; STACK POINTER STORAGE
000346  001460  000000                  SAVPC:  .WORD   0               ; PROGRAM COUNTER STORAGE
000347  001462  000000                  ZERO:   .WORD   0               ;
000348  001464  000001                  ONE:    .WORD   1               ;
000349  001466  000000                  MEMLIM: .WORD   0               ; HIGHEST LOCATION FOR NPR'S
000350  001470  000001                  KMACTV: .BLKW   1               ; KMC11 SELECTED ACTIVE
000351  001472  000001                  KMNUM:  .BLKW   1               ; OCTAL NUMBER OF KMC11'S
000352  001474  000001                  SAVACT: .BLKW   1               ; ORIGINAL ACTIVE DEVICES.
000353  001476  000001                  SAVNUM: .BLKW   1               ; WORKABLE NUMBER.
000354  001500  000000                  RUN:    .WORD   0               ; POINTER TO RUNNING DEVICES
000355                                          .EVEN
000356  001502  002072                  CREAM:  .WORD   KM.MAP-6        ; TABLE POINTER
000357  001504  002276                  MILK:   .WORD   CNT.MAP-4       ; TABLE POINTER
000358                                  ;
000359                                  ;  PROGRAM CONTROL FLAGS
000360                                  ; ----------------------
000361  001506     000                  INIFLG: .BYTE   0               ; PROGRAM INITIALIZING FLAG
000362          001510                          .EVEN
000363  001510     000                  LOKFLG: .BYTE   0               ; LOCK ON CURRENT TEST FLAG
000364  001511     000                  QV.FLG: .BYTE   0               ; QUICK VERIFY FLAG
000365                                                                  ; ON FIRST PASS OF EACH KMC11 ITERATIONS WILL BE SUPPRES
000366                                          .EVEN
```

```
                                .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 SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (SERRPC).
                                ;*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


001512                          $ERRTB:
                                .EVEN
                                ;*      DF              ;; DOES NOT APPLY IN THIS DIAGNOSTIC.
001512  000000                          0
001514  000000                          0
001516  000000                          0
001520  021330                          EM1
001522  021544                          DH1             ; ERROR 1
001524  021620                          DT1
001526  021351                          EM2
001530  021544                          DH1             ; ERROR 2
001532  021620                          DT1
001534  021330                          EM1
001536  021544                          DH1             ; ERROR 3
001540  021636                          DT2
001542  021405                          EM3
001544  021576                          DH2             ; ERROR 4
001546  021654                          DT3
001550  021421                          EM4
001552  021576                          DH2             ; ERROR 5
001554  021654                          DT3
001556  021453                          EM5
001560  021544                          DH1             ; ERROR 6
001562  021666                          DT4
001564  021501                          EM6
001566  021544                          DH1             ; ERROR 7
001570  021704                          DT5
001572  021517                          EM7
001574  021576                          DH2             ; ERROR 10
001576  021654                          DT3
                002034          .=2034
                                .SBTTL  APT PARAMETER BLOCK

                                ;;*****************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;*****************************************************************
                002034                  .$X=.   ;;SAVE CURRENT LOCATION
                000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
                000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  002034                          $APTHDR ;;POINT TO APT HEADER BLOCK
                002034                  .=.$X   ;;RESET LOCATION COUNTER
```

```
                        ;;*****************************************************************
                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                        ;INTERFACE SPEC.
                        
002034                  $APTHD:
002034  000000          $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
002036  001316          $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
002040  000132          $TSTM:  .WORD   90.     ;;RUN TIM OF LONGEST TEST
002042  000137          $PASTM: .WORD   95.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
002044  000137          $UNITM: .WORD   95.     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
002046  000052                  .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                ;KMC11 CONTROL INDICATORS FOR CURRENT KMC11 UNDER TEST
                                ;-----------------------------------------------------

002050  000000                  STAT1:  0
002052  000000                  STAT2:  0
002054  000000                  STAT3:  0

                                ;KMC11 VECTOR AND REGISTER INDIRECT POINTERS
                                ;-------------------------------------------

002056  000000                  KMRVEC: 0               ;POINTER TO KMC11 RECEIVER INTERRUPT VECTOR
002060  000000                  KMRLVL: 0               ;POINTER TO KMC11 RECEIVER INTERRUPT SERVICE PS
002062  000000                  KMTVEC: 0               ;POINTER TO KMC11 TRANSMITTER INTERRUPT VECTOR
002064  000000                  KMTLVL: 0               ;POINTER TO KMC11 TRANSMITTER INTERRUPT SERVICE PS
002066  000000                  KMCSR:  0               ;POINTER TO KMC11 CONTROL STATUS REGISTER
002070  000000                  KMCSRH: 0               ;POINTER TO KMC11 CONTROL STATUS REGISTER HIGH BYTE.
002072  000000                  KMCTL:  0               ;POINTER TO KMC11 CONTOL OUT REGISTER
002074  000000                  KMP04:  0               ;POINTER TO KMC11 PORT REGISTER(SEL 4)
002076  000000                  KMP06:  0               ;POINTER TO KMC11 PORT REGISTER(SEL 6)

                                ;TEMP STORAGE
                                ;------------

                                ;TEMP:  0
                                ;.=.+40

                                ;KMC11 STATUS TABLE AND ADDRESS ASSIGNMENTS
                                ;------------------------------------------

        002100                  .=2100
002100                          KM.MAP:
002100  000001                  KMCR00: .BLKW   1               ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 00
002102  000001                  KMS100: .BLKW   1               ;VECTOR FOR KMC11 NUMBER 00
002104  000001                  KMS200: .BLKW   1               ;DDCMP LINE# FOR KMC11 NUMBER 00
002106  000001                  KMS300: .BLKW   1               ;3RD STATUS WORD

002110  000001                  KMCR01: .BLKW   1               ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 01
002112  000001                  KMS101: .BLKW   1               ;VECTOR FOR KMC11 NUMBER 01
002114  000001                  KMS201: .BLKW   1               ;DDCMP LINE# FOR KMC11 NUMBER 01
002116  000001                  KMS301: .BLKW   1               ;3RD STATUS WORD

002120  000001                  KMCR02: .BLKW   1               ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 02
002122  000001                  KMS102: .BLKW   1               ;VECTOR FOR KMC11 NUMBER 02
002124  000001                  KMS202: .BLKW   1               ;DDCMP LINE# FOR KMC11 NUMBER 02
002126  000001                  KMS302: .BLKW   1               ;3RD STATUS WORD

002130  000001                  KMCR03: .BLKW   1               ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 03
002132  000001                  KMS103: .BLKW   1               ;VECTOR FOR KMC11 NUMBER 03
002134  000001                  KMS203: .BLKW   1               ;DDCMP LINE# FOR KMC11 NUMBER 03
002136  000001                  KMS303: .BLKW   1               ;3RD STATUS WORD

002140  000001                  KMCR04: .BLKW   1               ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 04
002142  000001                  KMS104: .BLKW   1               ;VECTOR FOR KMC11 NUMBER 04
002144  000001                  KMS204: .BLKW   1               ;DDCMP LINE# FOR KMC11 NUMBER 04
002146  000001                  KMS304: .BLKW   1               ;3RD STATUS WORD
```

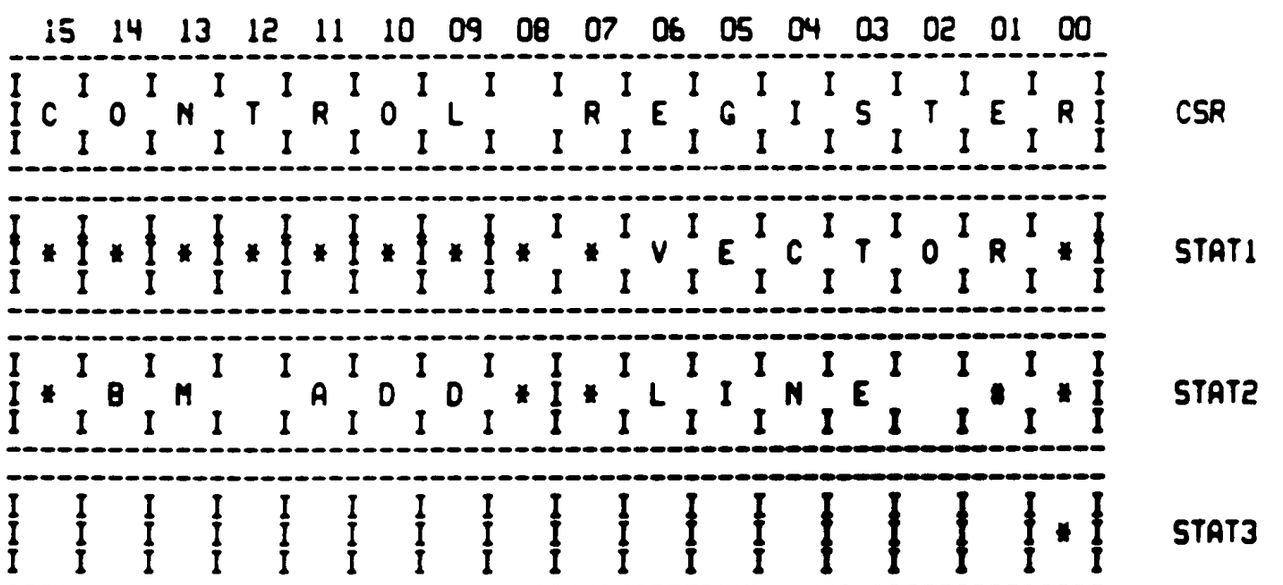
```
491
492  002150  000001  KMCR05: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 05
493  002152  000001  KMS105: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 05
494  002154  000001  KMS205: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 05
495  002156  000001  KMS305: .BLKW  1  ;3RD STATUS WORD
496
497  002160  000001  KMCR06: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 06
498  002162  000001  KMS106: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 06
499  002164  000001  KMS206: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 06
500  002166  000001  KMS306: .BLKW  1  ;3RD STATUS WORD
501
502  002170  000001  KMCR07: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 07
503  002172  000001  KMS107: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 07
504  002174  000001  KMS207: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 07
505  002176  000001  KMS307: .BLKW  1  ;3RD STATUS WORD
506
507  002200  000001  KMCR10: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 10
508  002202  000001  KMS110: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 10
509  002204  000001  KMS210: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 10
510  002206  000001  KMS310: .BLKW  1  ;3RD STATUS WORD
511
512  002210  000001  KMCR11: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 11
513  002212  000001  KMS111: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 11
514  002214  000001  KMS211: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 11
515  002216  000001  KMS311: .BLKW  1  ;3RD STATUS WORD
516
517  002220  000001  KMCR12: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 12
518  002222  000001  KMS112: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 12
519  002224  000001  KMS212: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 12
520  002226  000001  KMS312: .BLKW  1  ;3RD STATUS WORD
521
522  002230  000001  KMCR13: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 13
523  002232  000001  KMS113: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 13
524  002234  000001  KMS213: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 13
525  002236  000001  KMS313: .BLKW  1  ;3RD STATUS WORD
526
527  002240  000001  KMCR14: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 14
528  002242  000001  KMS114: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 14
529  002244  000001  KMS214: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 14
530  002246  000001  KMS314: .BLKW  1  ;3RD STATUS WORD
531
532  002250  000001  KMCR15: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 15
533  002252  000001  KMS115: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 15
534  002254  000001  KMS215: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 15
535  002256  000001  KMS315: .BLKW  1  ;3RD STATUS WORD
536
537  002260  000001  KMCR16: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 16
538  002262  000001  KMS116: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 16
539  002264  000001  KMS216: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 16
540  002266  000001  KMS316: .BLKW  1  ;3RD STATUS WORD
541
542  002270  000001  KMCR17: .BLKW  1  ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 17
543  002272  000001  KMS117: .BLKW  1  ;VECTOR FOR KMC11 NUMBER 17
544  002274  000001  KMS217: .BLKW  1  ;DDCMP LINE# FOR KMC11 NUMBER 17
545  002276  000001  KMS317: .BLKW  1  ;3RD STATUS WORD
546
```

```
   547  002300  000000                KM.END: 000000
```

```
548
549                             ;KMC11 PASS COUNT AND ERROR COUNT TABLE
550                             ;-------------------------------------
551
552  002302                     CNT.MAP:
553  002302  000000             PACT00: 0               ;PASS COUNT FOR KMC11 NUMBER 00
554  002304  000000             ERCT00: 0               ;ERROR COUNT FOR KMC11 NUMBER 00
555
556  002306  000000             PACT01: 0               ;PASS COUNT FOR KMC11 NUMBER 01
557  002310  000000             ERCT01: 0               ;ERROR COUNT FOR KMC11 NUMBER 01
558
559  002312  000000             PACT02: 0               ;PASS COUNT FOR KMC11 NUMBER 02
560  002314  000000             ERCT02: 0               ;ERROR COUNT FOR KMC11 NUMBER 02
561
562  002316  000000             PACT03: 0               ;PASS COUNT FOR KMC11 NUMBER 03
563  002320  000000             ERCT03: 0               ;ERROR COUNT FOR KMC11 NUMBER 03
564
565  002322  000000             PACT04: 0               ;PASS COUNT FOR KMC11 NUMBER 04
566  002324  000000             ERCT04: 0               ;ERROR COUNT FOR KMC11 NUMBER 04
567
568  002326  000000             PACT05: 0               ;PASS COUNT FOR KMC11 NUMBER 05
569  002330  000000             ERCT05: 0               ;ERROR COUNT FOR KMC11 NUMBER 05
570
571  002332  000000             PACT06: 0               ;PASS COUNT FOR KMC11 NUMBER 06
572  002334  000000             ERCT06: 0               ;ERROR COUNT FOR KMC11 NUMBER 06
573
574  002336  000000             PACT07: 0               ;PASS COUNT FOR KMC11 NUMBER 07
575  002340  000000             ERCT07: 0               ;ERROR COUNT FOR KMC11 NUMBER 07
576
577  002342  000000             PACT10: 0               ;PASS COUNT FOR KMC11 NUMBER 10
578  002344  000000             ERCT10: 0               ;ERROR COUNT FOR KMC11 NUMBER 10
579
580  002346  000000             PACT11: 0               ;PASS COUNT FOR KMC11 NUMBER 11
581  002350  000000             ERCT11: 0               ;ERROR COUNT FOR KMC11 NUMBER 11
582
583  002352  000000             PACT12: 0               ;PASS COUNT FOR KMC11 NUMBER 12
584  002354  000000             ERCT12: 0               ;ERROR COUNT FOR KMC11 NUMBER 12
585
586  002356  000000             PACT13: 0               ;PASS COUNT FOR KMC11 NUMBER 13
587  002360  000000             ERCT13: 0               ;ERROR COUNT FOR KMC11 NUMBER 13
588
589  002362  000000             PACT14: 0               ;PASS COUNT FOR KMC11 NUMBER 14
590  002364  000000             ERCT14: 0               ;ERROR COUNT FOR KMC11 NUMBER 14
591
592  002366  000000             PACT15: 0               ;PASS COUNT FOR KMC11 NUMBER 15
593  002370  000000             ERCT15: 0               ;ERROR COUNT FOR KMC11 NUMBER 15
594
595  002372  000000             PACT16: 0               ;PASS COUNT FOR KMC11 NUMBER 16
596  002374  000000             ERCT16: 0               ;ERROR COUNT FOR KMC11 NUMBER 16
597
598  002376  000000             PACT17: 0               ;PASS COUNT FOR KMC11 NUMBER 17
599  002400  000000             ERCT17: 0               ;ERROR COUNT FOR KMC11 NUMBER 17
600
```

```
601
602
603
604
605
606
```

FORMAT OF STATUS TABLE



DEFINITION OF FORMAT

CSR: CONTAINS KMC11 CSR ADDRESS

STAT1: BITS 00-08 IS KMC11 VECTOR ADDRESS
BIT14=1 ???? TURNAROUND CONNECTOR IS ON
BIT14=0 NO TURNAROUND CONNECTOR
BIT13=0 LINE UNIT IS AN M8201
BIT13=1 LINE UNIT IS AN M8202
BIT12=1 NO LINE UNIT
BITS 09-11 IS KMC11 BR PRIORITY LEVEL

STAT2: LOW BYTE IS SWITCH PAC#1 (DDCMP LINE NUMBER)
HIGH BYTE IS SWITCH PAC#2 (BM873 BOOT ADD)

STAT3: BIT0=1 DO FREE RUNNING TESTS ON KMC
(MUST BE SET TO A ONE MANUALLY [PROGRAMS G AND H ONLY])

```
                                        ;PROGRAM INITIALIZATION
                                        ;LOCK OUT INTERRUPTS
                                        ;SET UP PROCESSOR STACK
                                        ;SET UP POWER FAIL VECTOR
                                        ;CLEAR PROGRAM CONTROL FLAGS AND COUNTS
                                        ;TYPE TITLE MESSAGE

002402  012737  000340  177776  .START: MOV     #340,PS         ;LOCK OUT INTERRUPTS
002410  012706  001200                  MOV     #STACK,SP       ;SET UP STACK
002414  012737  007126  000024          MOV     #SPWRDN,@#24    ;SET UP POWER FAIL VECTOR
002422  013737  001472  001476          MOV     KMNUM,SAVNUM    ;SAVE NUMBER OF DEVICES IN SYSTEM.
002430  005037  011416                  CLR     SWFLG           ;CLEAR SOFT TYPEOUT FLAG
002434  105037  001203                  CLRB    SERFLG          ;CLEAR ERROR FLAG
002440  105037  001511                  CLRB    QV.FLG          ;ZERO QUICK VERIFY FLAG
002444  012737  002070  001502          MOV     #KM.MAP-10,CREAM;GET MAP POINTER.
002452  012737  002276  001504          MOV     #CNT.MAP-4,MILK ;GET PASS COUNT MAP POINTER
002460  012737  100000  001500          MOV     #BIT15,RUN      ;POINT POINTER TO FIRST DEVICE.
002466  012700  002302                  MOV     #CNT.MAP,R0     ;PASS COUNT POINTER TO R0
002472  005020                  23$:    CLR     (R0)+           ;CLEAR TABLE
002474  022700  002402                  CMP     #CNT.MAP+100,R0 ;DONE YET?
002500  001374                          BNE     23$             ;KEEP GOING
002502  005037  001216                  CLR     $ERRPC          ;CLEAR LAST ERROR POINTER
002506  012737  000001  001202          MOV     #1,$TSTNM       ;SET UP FOR TEST 1
002514  012737  002402  001206          MOV     #.START,$LPADR  ;SET UP FOR POWER FAIL BEFORE
                                                                ;TESTING STARTS
002522  132737  000001  001336          BITB    #1,$ENV         ; IS IT RUNNING UNDER APT?
002530  001404                          BEQ     3$              ; IF NOT CHECK FOR TYPE OF SWITCH REGISTER.
002532  013737  001340  000176          MOV     $SWREG,SWREG    ; LOAD SOFTWARE SWITCH REG.
002540  000423                          BR      6$+2            ; GO SET UP SOFTWARE SWITCH REG.
002542  013746  000006          3$:     MOV     @#6,-(SP)       ;SAVE CURRENT VECTORS
002546  013746  000004                  MOV     @#4,-(SP)       ;
002552  012737  002606  000004          MOV     #6$,@#4         ;SET UP FOR TIMEOUT
002560  012737  177570  001240          MOV     #177570,SWR     ;SET SWR TO HARD SWR ADDRESS
002566  012737  177570  001242          MOV     #177570,DISPLAY ;SET DISPLAY TO HARD SWR ADDRESS
002574  022777  177777  176436          CMP     #-1,@SWR        ;REFERENCE HARDWARE SWITCH REGISTER
002602  001402                          BEQ     6$+2            ;IF = -1 USE SOFT SWR ANYWAY
002604  000407                          BR      7$              ;IF IT EXISTS AND NOT = -1 USE HARD SWR
002606  022626                  6$:     CMP     (SP)+,(SP)+     ;ADJUST STACK
002610  012737  000176  001240          MOV     #SWREG,SWR      ;POINTER TO SOFT SWR
002616  012737  000174  001242          MOV     #DISPREG,DISPLAY;POINTER TO SOFT DISPLAY REG
002624  012637  000004          7$:     MOV     (SP)+,@#4       ;RESTORE VECTORS
002630  012637  000006                  MOV     (SP)+,@#6       ;
002634  105737  001506                  TSTB    INIFLG          ;HAS INITIALIZATION BEEN PERFORMED
002640  001006                          BNE     20$             ;BR IF YES
002642  022737  004070  000042          CMP     #$ENDAD,@#42    ;IF ACT-11 AUTOMATIC MODE, DON'T TYPE ID
002650  001402                          BEQ     20$
002652  104401  001000                  TYPE    ,MTITLE         ;TYPE TITLE MESSAGE
002656  004737  011212          20$:    JSR     PC,CKSWR                ;CHECK FOR SOFT SWR
002662  017737  176352  001446          MOV     @SWR,STRTSW     ;STORE STARTING SWITCHES
002670  005737  000042                  TST     @#42            ;IS IT RUNNING IN AUTO MODE?
002674  001402                          BEQ     .+6             ;BR IF NO
002676  005037  001446                  CLR     STRTSW          ;IF YES, CLEAR SWITCHES
002702  032737  000001  001446          BIT     #SW00,STRTSW    ;IF SW00=1, QUESTIONS ARE ASKED.
002710  001012                          BNE     17$             ;BR IF SW00=1
002712  105737  001446                  TSTB    STRTSW          ;BIT7=1??
```

```
                                BPL     17$             ;BR IF SW07=0
                                TST     KMACTV          ;ARE ANY DEVICES SELECTED?
                                BNE     16$             ;BR IF YES
                                TYPE,   NOACT           ;NO DEVICES SELECTED.
                                HALT                    ;STOP THE SHOW
                                BR      .-2             ;DISQUALIFY CONTINUE SWITCH
                        17$:    TSTB    $ENV            ; IS IT UNDER APT DUMP MODE?
                                BEQ     27$             ; YES, CHECK IF APT SIZED IT?
                                BITB    #1,$ENV         ; IS IT UNDER Q,V OR RUN MODE?
                                BNE     30$             ; YES, NEEDS ONLY APT SIZING.
                                BR      33$             ; NO, NEEDS REGULAR AUTO.SIZE.
                        27$:    TSTB    $ENVM           ; IS IT SIZED BY APT?
                                BMI     30$             ; YES, NEEDS ONLY APT SIZING.
                                BIC     #SWOO,STRTSW    ; SIZE ONLY IN AUTO MODE.
                        33$:    JSR     PC,AUTO.SIZE    ; GO DO THE AUTO.SIZE.
                                BR      16$             ; GO PRINT THE MAP.
                        30$:    JSR     PC,APT.SIZE     ; GO DO THE APT SIZING.
                        16$:    TSTB    INIFLG          ;FIRST TIME?
                                BEQ     21$             ;BR IF YES
                                TSTB    STRTSW          ;IF USING SAME PARAMETERS DONT TYPE MAP
                                BMI     1$
                                BIT     #BIT1!BIT2,STRTSW;IS TEST NO. OR LOCK SELECTED
                                BEQ     24$             ;IF NO THEN TYPE STATUS
                                BR      1$              ;IF YES DO NOT TYPE STATUS
                        21$:    COMB    INIFLG          ;SET FLAG
                        24$:    TYPE    ,XHEAD          ;TYPE HEADER
                                MOV     #KM.MAP,R4      ;SET POINTER
                        5$:     MOV     R4,$TMP0        ;SET ADDRESS
                                MOV     (R4)+,$TMP1     ;SET CSR
                                BEQ     1$              ;ALL DONE IF ZERO
                                MOV     (R4)+,$TMP2     ;SET STAT1
                                MOV     (R4)+,$TMP3     ;SET STAT2
                                MOV     (R4)+,$TMP4     ;SET STAT3
                                CONVRT                  ;TYPE OUT STATUS MAP
                                XSTATQ                  ;
                                BR      5$
                        1$:     MOV     #KM.MAP,R0      ;R0 POINTS TO STATUS TABLE

;;**************************************************************************
;;*AUTO SIZE TEST
;;*THIS TEST VERIFYS THAT THE KMC11S AND/OR KMC11S ARE AT THE CORRECT FLOATING
;;*ADDRESSES FOR YOUR SYSTEM. IF THIS TEST FAILS, IT IS NOT A HARDWARE ERROR.
;;*CHECK THE ADDRESSES OF ALL FLOATING DEVICES (DJ,DH,DQ,DU,DUP,LK,DMC,DZ,KMC).
;;*IF THERE ARE NO OTHER FLOATING DEVICES BEFORE THE KMC11, THE FIRST
;;* KMC11 IS 760110. NO DEVICE SHOULD EVER BE AT
;;*ADDRESS 760000.
;;**************************************************************************

                                MOV     @#4,-(SP)       ;SAVE LOC 4
                                MOV     @#6,-(SP)       ;SAVE LOC 6
                                CLR     @#6             ;CLEAR VEC+2
                                CLR     $TMP2           ;CLEAR FLAG
                        AUSTRT: MOV     (R0),KMCSR      ;GET NEXT KMC CSR
                                BEQ     AUDONE          ;BR IF DONE
                        2$:     MOV     #NODEV,@#4      ;SET UP FOR TIMEOUT
                        3$:     MOV     #10,R3          ;R3 IS COUNT OF DEVICES BEFORE KMC
```

```
                                4$:     MOV     #DEVTAB,R2      ;R2 IS DEVICE TABLE PONTER
                                        MOV     #160010,R1      ;START WITH ADDRESS 160010
                                FLOAT:  TST     (R1)            ;CHECK ADDRESS IN R1
                                        MOVB    (R2),R4         ;IF NO TIMEOUT, GET NEXT ADDRESS
                                        ADD     R4,R1           ;IN R1
                                        INC     R1              ;
                                        BIC     R4,R1           ;
                                        TST     R3              ;ANY MORE DEVICES TO CHECK FOR?
                                        BNE     FLOAT           ;BR IF YES
                                        MOV     #ERR,@#4        ;OK ONLY KMC'S ARE LEFT, SET UP FOR TIMEOUT
                                FY:     TST     (R1)            ;CHECK KMC ADDRESS
                                        CMP     R1,KMCSR        ;DOES IT MATCH
                                        BEQ     OK              ;BR IF YES
                                        ADD     #10,R1          ;GET NEXT KMC ADDRESS
                                        BR      FY              ;DO IT AGAIN
                                OK:     ADD     #10,R0          ;SKIP TO NEXT KMC CSR
                                        ADD     #10,R1          ; GET NEXT KMC ADDRESS
                                        MOV     (R0),KMCSR      ; GET NEXT KMC CSR
                                        BEQ     AUDONE          ; BRANCH IF ALL DONE.
                                        BR      FY              ; DO IT AGAIN.
                                NODEV:  CMPB    (R2)+,-(R3)     ;ON TIMEOUT, INC R2, DEC R3
                                        RTI                     ;$LPADR
                                ERR:    TST     $TMP2           ;CHECK FLAG IF = 0 TYPE HEADER
                                        BNE     1$              ;SKIP HEADER
                                        TYPE                    ;TYPEOUT HEADER MESSAGE
                                        CONERR                  ;CONFIGURATION ERROR!!!!
                                        MOV     #ERR,SAVPC      ;SAVE PC FOR TYPEOUT
                                        CNVRT                   ;TYPE OUT ERROR PC
                                        ERRPC                   ;
                                        TYPE                    ;TYPE REST OF HEADER
                                        CNERR                   ;
                                        MOV     #-1,$TMP2       ;SET FLAG SO IT ONLY GETS TYPED ONCE
                                1$:     MOV     R1,$REG1        ;SAVE R1 FOR TYPEOUT
                                        CONVRT                  ;
                                        CONTAB                  ;TYPE CSR VALUES
                                3$:     TYPE                    ;
                                        KMCM                    ;
                                4$:     CMP     (SP)+,(SP)+     ;ADJUST STACK
                                        BR      OK              ;BR TO GET OUT
                                ERRPC:  1                       ;
                                        .BYTE   6,2             ;
                                        SAVPC                   ;
                                CONTAB: 2                       ;
                                        .BYTE   6,4             ;
                                        $REG1                   ;
                                        .BYTE   6,2             ;
                                        KMCSR                   ;
                                DEVTAB: .BYTE   7               ;DJ
                                        .BYTE   17              ;DH
                                        .BYTE   7               ;DQ
                                        .BYTE   7               ;DU
                                        .BYTE   7               ;DUP
                                        .BYTE   7               ;LK
                                        .BYTE   7               ;DMC
                                        .BYTE   7               ;DZ
                                        .BYTE   7               ;KMC
```

```
                003354                  .EVEN
003354                          AUDONE:
003354  012637  000006          1$:     MOV     (SP)+,@#6       ;RESTORE LOC 6
003360  012637  000004                  MOV     (SP)+,@#4       ;RESTORE LOC 4
003364  032737  000010  001446          BIT     #SW03,STRTSW    ;SELECT SPECIFIC DEVICES??
003372  001422                          BEQ     3$              ;BR IF NO.
003374  104401  010017                  TYPE    ,MNEW           ;TYPE THE MESSAGE.
003400  005000                          CLR     R0              ;ZERO DATA LIGHTS
003402  000000                          HALT                    ;WAIT FOR USER TO TELL WHAT DEVICES TO RUN
003404  027737  175630  001474          CMP     @SWR,SAVACT     ;IS THE NUMBER VALID?
003412  101404                          BLOS    2$              ;BR IF NUMBER IS OK.
003414  104401  007672                  TYPE    ,MERR3          ;TELL USER OF INVALID NUMBER.
003420  000000                          HALT                    ;STOP EVERY THING.
003422  000776                          BR      .-2             ;RESTART THE PROGRAM AGAIN.
003424  017737  175610  001470  2$:     MOV     @SWR,KMACTV     ;GET NEW DEVICE PATTERN
003432  013700  001470                  MOV     KMACTV,R0       ;SHOW THE USER WHAT HE SELECTED.
003436  000000                          HALT                    ;CONTINUE DYNAMIC SWITCHES.
003440  012700  000300          3$:     MOV     #300,R0         ;PREPARE TO CLEAR THE FLOATING
003444  012701  000302                  MOV     #302,R1         ;VECTOR AREA. 300-776
003450  010120                  4$:     MOV     R1,(R0)+        ;START PUTTING "PC+2 - HALT"
003452  005021                          CLR     (R1)+           ;IN VECTOR AREA.
003454  022021                          CMP     (R0)+,(R1)+     ;POP POINTERS
003456  022700  001000                  CMP     #1000,R0        ;ALL DONE??
003462  001372                          BNE     4$              ;BR IF NO.

                                ;TEST START AND RESTART
                                ;----------------------

003464  012706  001200          .BEGIN: MOV     #STACK,SP       ;SET UP STACK
003470  013746  000006                  MOV     @#6,-(SP)       ;SAVE LOC 6
003474  013746  000004                  MOV     @#4,-(SP)       ;SAVE LOC 4
003500  005000                          CLR     R0              ;START AT 0
003502  012737  003546  000004          MOV     #2$,@#4         ;SET UP FOR TIME OUT
003510  005037  000006                  CLR     @#6             ;TO AUTOSIZE MEMORY
003514  005720                  6$:     TST     (R0)+           ;CHECK ADDRESS IN R0
003516  022700  157776                  CMP     #157776,R0      ;IS IT AT LEAST 28K
003522  001374                          BNE     6$              ;BR IF NO
003524  162700  007776                  SUB     #7776,R0        ;SAVE 2K FOR MONITORS
003530  010037  001466          7$:     MOV     R0,MEMLIM       ;STORE MEMORY LIMIT
003534  012637  000004                  MOV     (SP)+,@#4       ;RESTORE LOC 4
003540  012637  000006                  MOV     (SP)+,@#6       ;RESTORE LOC 6
003544  000413                          BR      10$             ;CONTINUE
003546  022626                  2$:     CMP     (SP)+,(SP)+     ;ADJUST STACK
003550  162700  000004                  SUB     #4,R0           ;GET LAST GOOD ADDRESS
003554  162700  007776                  SUB     #7776,R0        ;SAVE 2K FOR MONITORS
0035[illegible]  022700  030000          CMP     #30000,R0       ;IS IT 8K?
003564  001361                          BNE     7$              ;BR IF NO
003566  012700  037400                  MOV     #37400,R0       ;IF 8K DON'T SAVE 2K
003572  000756                          BR      7$              ;
003574  012737  000340  177776  10$:     MOV     #340,PS         ;LOCK OUT INTERRUPTS
003602  032737  000004  001446          BIT     #BIT2,STRTSW    ;CHECK FOR LOCK ON TEST
003610  001406                          BEQ     1$              ;BR IF NO LOCK DESIRED.
003612  104401  007716                  TYPE    ,MLOCK          ;TYPE LOCK SELECTED.
003616  012737  000240  004146          MOV     #NOP,TTST       ;SET UP TO LOCK
003624  000403                          BR      3$              ;CONTINUE ALONG.
003626  013737  004360  004146  1$:     MOV     BRW,TTST        ;PREPARE NORMAL SCOPE ROUTINE
```

```
  879  003634  012737  011460  001206  3$:     MOV     #CYCLE,$LPADR   ;START AT "CYCLE" FIND WHICH DEVICE TO TEST
  880  003642  032737  000002  001446  4$:     BIT     #SW01,STRTSW    ;IS TEST NO. SELECTED?
  881  003650  001002                          BNE     5$              ;BR IF YES
  882  003652  104401  007642                  TYPE    ,MR             ;TYPE R
  883  003656  000177  175324          5$:     JMP     @$LPADR         ;START TESTING
```

```
                        ;END OF PASS ROUTINE

                        .SBTTL  END OF PASS ROUTINE

                        ;;*****************************************************************
                        ;*INCREMENT THE PASS NUMBER ($PASS)
                        ;*IF THERES A MONITOR GO TO IT
                        ;*IF THERE ISN'T JUMP TO CYCLE

003662                  $EOP:
003662  000005                  RESET
003664  005237  001324          INC     $PASS           ; INCREMENT THE PASS COUNT
003670  105037  001203          CLRB    $ERFLG          ; CLEAR ERROR FLAG
003674  104401  007620          TYPE    ,MEPASS         ; TYPE END PASS.
003700  104401  007745          TYPE    ,MCSRX          ; TYPE "CSR"
003704  104417  004104          CNVRT   ,XCSR           ; SHOW IT.
003710  104401  007753          TYPE    ,MVECX          ; TYPE VECTOR.
003714  104417  004112          CNVRT   ,XVEC           ; SHOW IT.
003720  104401  007761          TYPE    ,MPASSX         ; TYPE " PASSES "
003724  104417  004120          CNVRT   ,XPASS          ; SHOW IT.
003730  104401  007772          TYPE    ,MERRX          ; TYPE " ERRORS "
003734  104417  004126          CNVRT   ,XERR           ; SHOW IT.
003740  013700  001504          MOV     #ILK,R0         ; SET POINTER TO PASSCNT.
003744  013720  001324          MOV     $PASS,(R0)+     ; SAVE THE PASS COUNT.
003750  013720  001212          MOV     $ERTTL,(R0)+    ; SAVE ERROR COUNT
003754  013777  002060  176074  MOV     KMRLVL,@KMRVEC  ; RESTORE THE RECEIVER INTERRUPT VECTOR.
003762  005077  176072          CLR     @KMRLVL         ; RESTORE RECEIVER LEVEL
003766  013777  002064  176066  MOV     KMTLVL,@KMTVEC  ; RESTORE THE TRANSMIT INTERRUPT VECTOR.
003774  005077  176064          CLR     @KMTLVL         ; RESTORE TRANSMITTER LEVEL
004000  005337  001476          DEC     SAVNUM          ; ALL DEVICE TESTED?
004004  001035                  BNE     $DOAGN          ; BRANCH IF NO.
004006  112737  000377  001511  MOVB    #377,QV.FLG     ; SET QUICK VERIFY FLAG.
004014  013737  001472  001476  MOV     KMNUM,SAVNUM    ; RESTORE DEVICE COUNT.
004022  005037  001216          CLR     $ERRPC          ; CLEAR LAST ERROR PC
004026  005037  001310          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
004032  005237  001324          INC     $PASS           ;;INCREMENT THE PASS NUMBER
004036  042737  100000  001324  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
004044  005327                  DEC     (PC)+           ;;LOOP?
004046  000001          $EOPCT: .WORD   1
004050  003013                  BGT     $DOAGN          ;;YES
004052  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
004054  000001          $ENDCT: .WORD   1
004056  004046                  $EOPCT
004060  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
004064  001405                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
004066  000005                  RESET                   ;;CLEAR THE WORLD
004070  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
004072  000240                  NOP                     ;;SAVE ROOM
004074  000240                  NOP                     ;;FOR
004076  000240                  NOP                     ;;ACT11
004100                  $DOAGN:
004100  000137                  JMP     @(PC)+          ;;RETURN
```

```
940  004102  011460                  $RTNAD: .WORD   CYCLE
941  004104  000001                  XCSR:   1
942  004106     006      002                 .BYTE   6,2
943  004110  002066                         KMCSR
944  004112  000001                  XVEC:   1
945  004114     004      002                 .BYTE   4,2
946  004116  002056                         KMRVEC
947  004120  000001                  XPASS:  1
948  004122     006      002                 .BYTE   6,2
949  004124  001324                         $PASS
950  004126  000001                  XERR:   1
951  004130     006      002                 .BYTE   6,2
952  004132  001212                         $ERTTL
953
954                                         ;SCOPE LOOP AND INTERATION HANDLER
955                                         ;-------------------------------
956
957                                 .SBTTL  SCOPE HANDLER ROUTINE
958
959                                 ;;**********************************************************************
960                                 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
961                                 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
962                                 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
963                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
964                                 ;*SW14=1         LOOP ON TEST
965                                 ;*SW11=1         INHIBIT ITERATIONS
966                                 ;*CALL
967                                 ;*      SCOPE           ;;SCOPE=IOT
968
969  004134                         $SCOPE:
970  004134  005037  001216                 CLR     $ERRPC                  ; CLEAR LAST ERROR PC
971  004140  023716  013734                 CMP     TST1+2,(SP)             ; IS THIS TEST #1 ?
972  004144  001413                         BEQ     $XTSTR                  ; IF SO DON'T LOOP.
973  004146  000406                 TTST:   BR      1$                      ;
974  004150  105777  175070                 TSTB    @#TKS                   ; KEYBOARD DONE ?
975  004154  100067                         BPL     $OVER                   ; IF NO DONT WAIT.
976  004156  017766  175064  177776         MOV     @#TKB,-2(SP)            ;
977  004164  032777  040000  175046 1$:      BIT     #BIT14,@#SWR    ;;LOOP ON PRESENT TEST?
978  004172  001060                         BNE     $OVER           ;;YES IF SW14=1
979                                 ;#####START OF CODE FOR THE XOR TESTER#####
980  004174  000416                 $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
981                                                                 ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
982  004176  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
983  004202  012737  004222  000004         MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
984  004210  005737  177060                 TST     @#177060        ;;TIME OUT ON XOR?
985  004214  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
986  004220  000436                         BR      $SVLAD          ;;GO TO THE NEXT TEST
987  004222  022626                 5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
988  004224  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
989  004230  000441                         BR      $OVER           ;;LOOP ON THE PRESENT TEST
990  004232                         6$:;#####END OF CODE FOR THE XOR TESTER#####
991  004232  105737  001203         2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
992  004236  001404                         BEQ     3$              ;;BR IF NO
993  004240  105037  001203         4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
994  004244  005037  001310                 CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
995  004250  032777  004000  174762 3$:     BIT     #BIT11,@#SWR    ;;INHIBIT ITERATIONS?
```

```
                                        BNE     1$                      ;;BR IF YES
                                        TST     $PASS                   ;;IF FIRST PASS OF PROGRAM
                                        BEQ     1$                      ;;      INHIBIT ITERATIONS
                                        INC     $ICNT                   ;;INCREMENT ITERATION COUNT
                                        CMP     $TIMES,$ICNT            ;;CHECK THE NUMBER OF ITERATIONS MADE
                                        BGE     $OVER                   ;;BR IF MORE ITERATION REQUIRED
                                1$:     MOV     #1,$ICNT                ;;REINITIALIZE THE ITERATION COUNTER
                                        MOV     $MXCNT,$TIMES           ;;SET NUMBER OF ITERATIONS TO DO
                                $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
                                        MOVB    $TSTNM,$TESTN           ;;SET TEST NUMBER IN APT MAILBOX
                                        MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
                                $OVER:  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER
                                        MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
                                        CLR     LOCK                            ; RESET LOCK ON DATA.
                                        MOV     KMCSR,R1                        ; R1 CONTAINS BASE KMC ADDRESS.
                                        RTI
                                BRW:    .WORD   406
                                $MXCNT: 20                              ;;MAX. NUMBER OF ITERATIONS

                                        ;CHECK FOR FREEZE ON CURRENT DATA
                                        ;--------------------------------

                                .SCOP1: JSR     PC,CKSWR                        ;CHECK FOR SOFT SWR
                                        BIT     #SW09,@SWR              ;IS SW09=1(SET)?
                                        BEQ     1$                      ;BR IF NOT SET.
                                        TST     LOCK
                                        BEQ     1$
                                        MOV     LOCK,(SP)               ;GOTO THE ADDRESS IN LOCK.
                                1$:     RTI                             ;GO BACK.

                                        ;TELETYPE OUTPUT ROUTINE
                                        ;-----------------------

                                .SBTTL  TYPE ROUTINE

                                ;;**********************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
                                ;*      TYPE    ,MESADR                 ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                ;*OR
                                ;*      TYPE
                                ;*      MESADR
                                ;*

                                $TYPE:  TSTB    $TPFLG                  ;;IS THERE A TERMINAL?
                                        BPL     1$                      ;;BR IF YES
                                        HALT                            ;;HALT HERE IF NO TERMINAL
                                        BR      3$                      ;;LEAVE
                                1$:     MOV     R0,-(SP)                ;;SAVE R0
                                        MOV     @2(SP),R0               ;;GET ADDRESS OF ASCIZ STRING
```

```
1052  004434  122737  000001  001336          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
1053  004442  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
1054  004444  132737  000100  001337          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
1055  004452  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
1056  004454  010037  004464                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
1057  004460  004737  004704                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
1058  004464  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
1059  004466  132737  000040  001337  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
1060  004474  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
1061  004476  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1062  004500  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1063  004502  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1064  004504  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
1065  004506  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1066  004512  000002                          RTI                     ;;RETURN
1067  004514  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1068  004520  001430                          BEQ     8$
1069  004522  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
1070  004526  001006                          BNE     5$
1071  004530  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
1072  004532  104401                          TYPE                    ;;TYPE A CR AND LF
1073  004534  001313                          $CRLF
1074  004536  105037  004672                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
1075  004542  000755                          BR      2$              ;;GET NEXT CHARACTER
1076  004544  004737  004626          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1077  004550  123726  001256          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1078  004554  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
1079  004556  013746  001254                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1080                                                                  ;;AND THE NULL CHAR.
1081  004562  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1082  004566  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1083  004570  004737  004626                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
1084  004574  105337  004672                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
1085  004600  000770                          BR      7$              ;;LOOP
1086
1087                                  ;HORIZONTAL TAB PROCESSOR
1088
1089  004602  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1090  004606  004737  004626          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1091  004612  132737  000007  004672          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1092  004620  001372                          BNE     9$              ;;TAB STOP
1093  004622  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
1094  004624  000724                          BR      2$              ;;GET NEXT CHARACTER
1095  004626  105777  174416          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
1096  004632  100375                          BPL     $TYPEC
1097  004634  116677  000002  174410          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1098  004642  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1099  004650  001003                          BNE     1$              ;;BRANCH IF NO
1100  004652  105037  004672                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1101  004656  000406                          BR      $TYPEX          ;;EXIT
1102  004660  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1103  004666  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
1104  004670  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
1105  004672  000000          $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
1106  004674  000207          $TYPEX: RTS     PC
1107
```

```
                                .SBTTL  APT COMMUNICATIONS ROUTINE

                                ;;*****************************************************************
004676  112737  000001  005142  $ATY1:   MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
004704  112737  000001  005140  $ATY3:   MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
004712  000403                  BR      $ATYC
004714  112737  000001  005142  $ATY4:   MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
004722                          $ATYC:
004722  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
004724  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
004726  105737  005140          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
004732  001450                  BEQ     5$              ;;IF NOT:  BR
004734  122737  000001  001336  CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
004742  001031                  BNE     3$              ;;IF NOT:  BR
004744  132737  000100  001337  BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
004752  001425                  BEQ     3$              ;;IF NOT:  BR
004754  017600  000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
004760  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
004766  005737  001316          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
004772  001375                  BNE     1$              ;;IF NOT:  WAIT
004774  010037  001332          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
005000  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
005002  001376                  BNE     2$
005004  163700  001332          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
005010  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
005012  010037  001334          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
005016  012737  000004  001316  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
005024  000413                  BR      5$
005026  017637  000004  005052  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
005034  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
005042  013746  177776          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
005046  004737  004414          JSR     PC,$TYPE        ;;CALL TYPE MACRO
005052  000000                  4$:     .WORD   0
005054                          5$:
005054  105737  005142          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
005060  001416                  BEQ     12$             ;;IF NOT:  BR
005062  005737  001336          TST     $ENV            ;;RUNNING UNDER APT?
005066  001413                  BEQ     12$             ;;IF NOT:  BR
005070  005737  001316          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
005074  001375                  BNE     11$             ;;IF NOT:  WAIT
005076  017637  000004  001220  MOV     @4(SP),$FATAL   ;;GET ERROR #
005104  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
005112  005237  001316          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
005116  105037  005142          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
005122  105037  005141          CLRB    $LFLG           ;;CLEAR LOG FLAG
005126  105037  005140          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
005132  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
005134  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
005136  000207                  RTS     PC              ;;RETURN
005140     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
005141     000          $LFLG:  .BYTE   0               ;;LOG FLAG
005142     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
005144                          .EVEN
000200                  APTSIZE=200
000001                  APTENV=001
000100                  APTSPOOL=100
```

```
                        000040                  APTCSUP=040
                                                        ;-------------------------
                                                .SBTTL  TTY INPUT ROUTINE
                                                ;;**********************************************************************
                                                .ENABL  LSB
                                                .DSABL  LSB
                                                ;;**********************************************************************
                                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                                ;*CALL:
                                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                                ;
005144  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
005146  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
005154  105777  174064          1$:     TSTB    @#TKS           ;;WAIT FOR
005160  100375                          BPL     1$              ;;A CHARACTER
005162  117766  174060  000004          MOVB    @#TKB,4(SP)     ;;READ THE TTY
005170  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
005176  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
005204  001013                          BNE     3$              ;;BRANCH IF NO
005206  105777  174032          2$:     TSTB    @#TKS           ;;WAIT FOR A CHARACTER
005212  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
005214  117746  174026                  MOVB    @#TKB,-(SP)     ;;GET CHARACTER
005220  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
005224  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
005230  001366                          BNE     2$              ;;IF NOT DISCARD IT
005232  000750                          BR      1$              ;;YES, RESUME
005234  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
005242  002407                          BLT     4$              ;;BRANCH IF YES
005244  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
005252  003003                          BGT     4$              ;;BRANCH IF YES
005254  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
005262  000002                  4$:     RTI                     ;;GO BACK TO USER
                                                ;;**********************************************************************
                                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                                ;*CALL:
                                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
005264  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
005266  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
005270  012703  005520          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
005274  022703  005527          2$:     CMP     #$TTYIN+7,R3    ;;BUFFER FULL?
005300  101456                          BLOS    4$              ;;BR IF YES
005302  104402                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
005304  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
005306  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
005312  001022                          BNE     5$              ;;BR IF NO
```

```
1220  005314  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
1221  005316  001007                          BNE     6$              ;;BR IF NO
1222  005320  112737  000134  005516          MOVB    #'\,9$          ;;TYPE A BACK SLASH
1223  005326  104401  005516                  TYPE    9$
1224  005332  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
1225  005336  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
1226  005340  020327  005520                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
1227  005344  103434                          BLO     4$              ;;BR IF YES
1228  005346  111337  005516                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
1229  005352  104401  005516                  TYPE    9$              ;;GO TYPE
1230  005356  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
1231  005360  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
1232  005362  001406                          BEQ     7$              ;;BR IF NO
1233  005364  112737  000134  005516          MOVB    #'\,9$          ;;TYPE A BACK SLASH
1234  005372  104401  005516                  TYPE    9$
1235  005376  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
1236  005400  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
1237  005404  001003                          BNE     8$              ;;BR IF NO
1238  005406  104401  005527                  TYPE    $CNTLU          ;;TYPE A CONTROL "U"
1239  005412  000726                          BR      1$              ;;GO START OVER
1240  005414  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "^R"?
1241  005420  001011                          BNE     3$              ;;BRANCH IF NO
1242  005422  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
1243  005424  104401  001313                  TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
1244  005430  104401  005520                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
1245  005434  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
1246  005436  104401  001312          4$:     TYPE    $QUES           ;;TYPE A '?'
1247  005442  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
1248  005444  111337  005516          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
1249  005450  104401  005516                  TYPE    9$
1250  005454  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
1251  005460  001305                          BNE     2$              ;;LOOP IF NOT RETURN
1252  005462  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
1253  005466  104401  001314                  TYPE    $LF             ;;TYPE A LINE FEED
1254  005472  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
1255  005474  012603                          MOV     (SP)+,R3        ;;RESTORE R3
1256  005476  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
1257  005500  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
1258  005506  012766  005520  000004          MOV     #$TTYIN,4(SP)
1259  005514  000002                          RTI                     ;;RETURN
1260  005516     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
1261  005517     000                          .BYTE   0               ;;TERMINATOR
1262  005520  000007                  $TTYIN: .BLKB   7               ;;RESERVE 7 BYTES FOR TTY INPUT
1263  005527     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
1264  005534  043536  005515     000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
1265  005541     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
1266  005546  036440  000040
1267  005552  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
1268  005560  020075     000
1269          005564                  .EVEN
1270                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
1271
1272                                  ;;**************************************************************
1273                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
1274                                  ;*CHANGE IT TO BINARY.
1275                                  ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
```

```
1276                                    ;*OCTAL DIGITS, IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
1277                                    ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
1278                                    ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
1279                                    ;*CALL:
1280                                    ;*      RDOCT                   ;;READ AN OCTAL NUMBER
1281                                    ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
1282                                    ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
1283
1284  005564  011646                    $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
1285  005566  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
1286  005574  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1287  005576  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1288  005600  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1289  005602  104403                    1$:     RDLIN                   ;;READ AN ASCIZ LINE
1290  005604  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
1291  005606  010037  005712                    MOV     R0,5$           ;;AND SAVE IT
1292  005612  005001                            CLR     R1              ;;CLEAR DATA WORD
1293  005614  005002                            CLR     R2
1294  005616  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
1295  005620  001420                            BEQ     3$              ;;IF ZERO GET OUT
1296  005622  122716  000060                    CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
1297  005626  003026                            BGT     4$              ;;IS AN OCTAL DIGIT
1298  005630  122716  000067                    CMPB    #'7,(SP)
1299  005634  002423                            BLT     4$
1300  005636  006301                            ASL     R1              ;;*2
1301  005640  006102                            ROL     R2
1302  005642  006301                            ASL     R1              ;;*4
1303  005644  006102                            ROL     R2
1304  005646  006301                            ASL     R1              ;;*8
1305  005650  006102                            ROL     R2
1306  005652  042716  177770                    BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
1307  005656  062601                            ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
1308  005660  000756                            BR      2$              ;;LOOP
1309  005662  005726                    3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
1310  005664  010166  000012                    MOV     R1,12(SP)       ;;SAVE THE RESULT
1311  005670  010237  005722                    MOV     R2,$HIOCT
1312  005674  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
1313  005676  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
1314  005700  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
1315  005702  000002                            RTI                     ;;RETURN
1316  005704  005726                    4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
1317  005706  105010                            CLRB    (R0)            ;;SET A TERMINATOR
1318  005710  104401                            TYPE                    ;;TYPE UP THRU THE BAD CHAR.
1319  005712  000000                    5$:     .WORD   0
1320  005714  104401  001312                    TYPE    ,$QUES          ;;"?" "CR" & "LF"
1321  005720  000730                            BR      1$              ;;TRY AGAIN
1322  005722  000000                    $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
1323                                    ;
1324                                    ;    INPUT OCTAL NUMBER ROUTINE
1325                                    ;------------ --------------------
1326                                    ;
1327  005724  010546                    $INPUT: MOV     R5,-(SP)                ; SAVE REGISTER R5.
1328  005726  016605  000002                    MOV     2(SP),R5                ; GET FIRST PARAMETER ADDRESS.
1329  005732  012537  005770                    MOV     (R5)+,WHAT              ; GET MESSAGE ADDRESS.
1330  005736  012537  006050                    MOV     (R5)+,LOLIM             ; GET LOW LIMIT FOR THE  #
1331  005742  012537  006052                    MOV     (R5)+,HILIM             ; GET HIGH LIMIT FOR THE #.
```

```
1332  005746  012537  006054          MOV     (R5)+,WHERE     ; GET ADDRESS OF INBUFFER
1333  005752  112537  006056          MOVB    (R5)+,LOBITS    ; GET LOWMASK BITS.
1334  005756  112537  006057          MOVB    (R5)+,ADRCNT    ; GET # OF #'S TO BE GENERATED.
1335  005762  010566  000002          MOV     R5,2(SP)        ; SAVE THE RETURN ADDRESS.
1336  005766  104401            INLP1: TYPE                    ; TYPE THE MESSAGE.
1337  005770  000000            WHAT:  .WORD   0
1338  005772  104404                   RDOCT                   ; READ OCTAL # FROM KEYBOARD.
1339  005774  021637  006052          CMP     (SP),HILIM      ; IS IT IN HIGH LIMIT?
1340  006000  003003                   BGT     2$              ; BRANCH IF NO.
1341  006002  021637  006050          CMP     (SP),LOLIM      ; IS IT MORE THAN LOW LIMIT.
1342  006006  002005                   BGE     3$              ; BRANCH IF YES.
1343  006010  104401  001312    2$:    TYPE    ,$QUES          ; TYPE  " ? "
1344  006014  104401  001313          TYPE    ,$CRLF          ; TYPE (CR),(LF)
1345  006020  000762                   BR      INLP1           ;
1346  006022  013705  006054    3$:    MOV     WHERE,R5        ; GET BUFFER ADDRESS.
1347  006026  011625            4$:    MOV     (SP),(R5)+      ; SAVE THE # IN RIGHT PLACE.
1348  006030  062716  000002          ADD     #2,(SP)         ; NEXT SEQUENTIAL NUMBER.
1349  006034  105337  006057          DECB    ADRCNT          ; COUNT BY 1.
1350  006040  001372                   BNE     4$              ; BRANCH IF NOT DONE.
1351  006042  005726                   TST     (SP)+           ; POP THE STACK POINTER.
1352  006044  012605                   MOV     (SP)+,R5        ; POP THE REG.5
1353  006046  000002                   RTI
1354  006050  000000            LOLIM: .WORD   0
1355  006052  000000            HILIM: .WORD   0
1356  006054  000000            WHERE: .WORD   0
1357  006056  000               LOBITS: .BYTE   0
1358  006057  000               ADRCNT: .BYTE   0
1359
1360                                    ;  ADVANCE TO NEXT TEST HANDLER
1361                                    ;---------------------------------
1362                                    ;
1363  006060  013716  001442    .ADVANCE;       MOV     NEXT,(SP)               ; CRUNCH STACK WITH ADDRESSOF SCOPE CALL
1364  006064  005037  001444            CLR     LOCK                    ; RESET TIGHT LOOP ADDRESS
1365  006070  000002                    RTI                             ; CHECK TO SEE IF OLD TEST GETS REPEATED
1366
1367                                    ;SAVE PC OF TEST THAT FAILED AND R0-R5
1368                                    ;-------------------------------------
1369
1370  006072  016637  000004  001460 .SAVOS: MOV     4(SP),SAVPC     ;SAVE R7 (PC)
1371
1372                                    ;SAVE R0-R5
1373
1374  006100  010537  001274    SVOS:   MOV     R5,$REG5        ;SAVE R5
1375  006104  010437  001272            MOV     R4,$REG4        ;SAVE R4
1376  006110  010337  001270            MOV     R3,$REG3        ;SAVE R3
1377  006114  010237  001266            MOV     R2,$REG2        ;SAVE R2
1378  006120  010137  001264            MOV     R1,$REG1        ;SAVE R1
1379  006124  010037  001262            MOV     R0,$REG0        ;SAVE R0
1380  006130  000002                    RTI                     ;LEAVE.
1381
1382                                    ;RESTORE R0-R5
1383
1384  006132  013700  001262    .RESOS: MOV     $REG0,R0        ;RESTORE R0
1385  006136  013701  001264            MOV     $REG1,R1        ;RESTORE R1
1386  006142  013702  001266            MOV     $REG2,R2        ;RESTORE R2
1387  006146  013703  001270            MOV     $REG3,R3        ;RESTORE R3
```

```
 1388  006152  013704  001272                  MOV     $REG4,R4        ;RESTORE R4
 1389  006156  013705  001274                  MOV     $REG5,R5        ;RESTORE R5
 1390  006162  000002                          RTI                     ;LEAVE
 1391
 1392                                  ;       ;CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
 1393                                  ;       ;-----------------------------------------------------
 1394                                  ;
 1395  006164  104401  001313          .CONVR: TYPE    ,$CRLF
 1396  006170  010046                  .CNVRT: MOV     R0,-(SP)
 1397  006172  010146                          MOV     R1,-(SP)
 1398  006174  010346                          MOV     R3,-(SP)
 1399  006176  010446                          MOV     R4,-(SP)
 1400  006200  010546                          MOV     R5,-(SP)
 1401  006202  017601  000012                  MOV     @12(SP),R1
 1402  006206  062766  000002  000012          ADD     #2,12(SP)
 1403  006214  012137  006406                  MOV     (R1)+,WRDCNT
 1404  006220  112137  006410          1$:     MOVB    (R1)+,CHRCNT
 1405  006224  112137  006411                  MOVB    (R1)+,SPACNT
 1406  006230  013137  006412                  MOV     @(R1)+,BINWRD
 1407  006234  122737  000003  006410          CMPB    #3,CHRCNT
 1408  006242  001003                          BNE     2$
 1409  006244  042737  177400  006412          BIC     #177400,BINWRD
 1410  006252  013704  006412          2$:     MOV     BINWRD,R4
 1411  006256  113705  006410                  MOVB    CHRCNT,R5
 1412  006262  012700  011106                  MOV     #TEMP,R0
 1413  006266  010403                  3$:     MOV     R4,R3
 1414  006270  042703  177770                  BIC     #177770,R3
 1415  006274  062703  000060                  ADD     #060,R3
 1416  006300  110320                          MOVB    R3,(R0)+
 1417  006302  000241                          CLC
 1418  006304  006004                          ROR     R4
 1419  006306  000241                          CLC
 1420  006310  006004                          ROR     R4
 1421  006312  000241                          CLC
 1422  006314  006004                          ROR     R4
 1423  006316  005305                          DEC     R5
 1424  006320  001362                          BNE     3$
 1425  006322  012703  011150                  MOV     #MDATA,R3
 1426  006326  114023                  4$:     MOVB    -(R0),(R3)+
 1427  006330  105337  006410                  DECB    CHRCNT
 1428  006334  001374                          BNE     4$
 1429  006336  105737  006411                  TSTB    SPACNT
 1430  006342  001405                          BEQ     6$
 1431  006344  112723  000040          5$:     MOVB    #040,(R3)+
 1432  006350  105337  006411                  DECB    SPACNT
 1433  006354  001373                          BNE     5$
 1434  006356  105013                  6$:     CLRB    (R3)
 1435  006360  104401  011150                  TYPE    ,MDATA
 1436  006364  005337  006406                  DEC     WRDCNT
 1437  006370  001313                          BNE     1$
 1438  006372  012605                          MOV     (SP)+,R5
 1439  006374  012604                          MOV     (SP)+,R4
 1440  006376  012603                          MOV     (SP)+,R3
 1441  006400  012601                          MOV     (SP)+,R1
 1442  006402  012600                          MOV     (SP)+,R0
 1443  006404  000002                          RTI
```

```
1444  006406  000000                  WRDCNT: 0
1445  006410  000000                  CHRCNT: 0
1446          006411                  SPACNT=CHRCNT+1
1447  006412  000000                  BINWRD: 0
1448
1449
1450                                          ;TRAP DISPATCH SERVICE
1451                                          ;ARGUMENT OF TRAP IS EXTRACTED
1452                                          ;AND USED AS OFFSET TO OBTAIN POINTER
1453                                          ;TO SELECTED SUBROUTINE
1454
1455                                  .SBTTL  TRAP DECODER
1456
1457                                  ;;******************************************************************
1458                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1459                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1460                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1461                                  ;*GO TO THAT ROUTINE.
1462
1463  006414  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
1464  006416  016600  000002         MOV     2(SP),R0        ;;GET TRAP ADDRESS
1465  006422  005740                 TST     -(R0)           ;;BACKUP BY 2
1466  006424  111000                 MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
1467  006426  006300                 ASL     R0              ;;POSITION FOR INDEXING
1468  006430  016000  006450         MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
1469  006434  000200                 RTS     R0              ;;GO TO ROUTINE
1470
1471
1472                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
1473
1474  006436  011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
1475  006440  016666  000004  000002         MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
1476  006446  000002                 RTI                     ;;RESTORE THE PSW
1477
1478                                  .SBTTL  TRAP TABLE
1479
1480                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1481                                  ;*BY THE "TRAP" INSTRUCTION.
1482
1483                                  ;       ROUTINE
1484                                  ;       -------
1485  006450  006436          $TRPAD: .WORD   $TRAP2
1486  006452  004414                 $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
1487
1488
1489  006454  005144                 $RDCHR  ;;CALL=RDCHR     TRAP+2(104402)  TTY TYPEIN CHARACTER ROUTINE
1490  006456  005264                 $RDLIN  ;;CALL=RDLIN     TRAP+3(104403)  TTY TYPEIN STRING ROUTINE
1491  006460  005564                 $RDOCT  ;;CALL=RDOCT     TRAP+4(104404)  READ AN OCTAL NUMBER FROM TTY
1492  006462  004364                 .SCOP1  ;;CALL=SCOP1     TRAP+5(104405)  CALL TO LOOP ON CURRENT DATA HANDLER
1493  006464  006072                 .SAVOS  ;;CALL=SAVOS     TRAP+6(104406)  CALL TO REGISTER SAVE ROUTINE
1494  006466  006132                 .RESOS  ;;CALL=RESOS     TRAP+7(104407)  CALL TO REGISTER RESTORE ROUTINE
1495  006470  007362                 .MSTCLR ;;CALL=MSTCLR    TRAP+10(104410) CALL TO ISSUE A MASTER CLEAR
1496  006472  007332                 .DELAY  ;;CALL=DELAY     TRAP+11(104411) CALL TO DELAY
1497  006474  007400                 .ROMCLK ;;CALL=ROMCLK    TRAP+12(104412) CALL TO CLOCK ROM ONCE
1498  006476  007446                 .DATACLK        ;;CALL=DATACLK  TRAP+13(104413) CALL TO CLOCK DATA
1499  006500  007512                 .TIMER  ;;CALL=TIMER     TRAP+14(104414) CALL TO DELAY A CLOCK TICK
```

```
1500  006502  005724                  $INPUT  ;;CALL=INPUT    TRAP+15(104415) CALL TO OCTAL # INPUT ROUTINE
1501  006504  006164                  .CONVRT ;;CALL=CONVRT   TRAP+16(104416) CALL TO    .....
1502  006506  006170                  .CNVRT  ;;CALL=CNVRT    TRAP+17(104417) CALL TO    .....
1503  006510  006060                  .ADVANCE        ;;CALL=ADVANCE  TRAP+20(104420) CALL TO ADVANCE TO NEXT TEST
1504                            ;
1505                            ;-------------------------------------------------------------
1506                            ;:*************************************************************
1507                                    ;ERROR HANDLER
1508                                    ;-------------
1509
1510  006512  004737  011212    $ERROR: JSR     PC,CKSWR        ;CHECK FOR SOFT SWR
1511  006516  032777  010000  172514    BIT     #SW12,@SWR      ;BELL ON ERROR?
1512  006524  001406                    BEQ     XBX             ;BR IF NO BELL
1513  006526  105777  172516            TSTB    @$TPS           ;TTY READY.
1514  006532  100003                    BPL     XBX             ;DON'T WAIT IF TTY NOT READY.
1515  006534  112777  000207  172510    MOVB    #207,@$TPB      ;PUSH A BELL AT THE TTY.
1516  006542  032777  020000  172470  XBX:    BIT     #SW13,@SWR      ;DELETE ERROR PRINT OUT?
1517  006550  001107                    BNE     HALT$           ;BR IF NO PRINT OUT WANTED.
1518  006552  021637  001216            CMP     (SP),$ERRPC     ;WAS THIS ERROR FOUND LAST TIME?
1519  006556  001404                    BEQ     1$              ;BR IF YES
1520  006560  011637  001216            MOV     (SP),$ERRPC     ;RECORD BEING HERE
1521  006564  105037  001203            CLRB    $ERFLG          ;PREPARE HEADER
1522  006570  104406                1$:     SAVO5                   ;SAVE ALL PROC REGISTERS
1523  006572  011605                    MOV     (SP),R5         ;GET THE PC OF ERROR
1524  006574  162705  000002            SUB     #2,R5           ;GET ADDRESS OF TRAP CALL
1525  006600  011504                    MOV     (R5),R4         ;GET ERROR INSTRUCTION
1526  006602  110437  001214            MOVB    R4,$ITEMB       ;COPY ERROR # FOR APT HANDLING
1527  006606  006304                    ASL     R4              ;MULT BY TWO
1528  006610  061504                    ADD     (R5),R4         ;DOUBLE IT
1529  006612  006304                    ASL     R4              ;MULT AGAIN
1530  006614  042704  177001            BIC     #177001,R4      ;CLEAR JUNK
1531  006620  062704  001512            ADD     #$ERRTB,R4      ;GET POINTER
1532  006624  012437  006740            MOV     (R4)+,ERRMSG    ;GET ERROR MESSAGE
1533  006630  012437  006752            MOV     (R4)+,DATAHD    ;GET DATA HEADRER
1534  006634  011437  006764            MOV     (R4),DATABP     ;GET DATA TABLE
1535  006640  105737  001203            TSTB    $ERFLG          ;TYPE HEADREER
1536  006644  001403                    BEQ     TYPMSG          ;BR IF YES
1537  006646  005737  006764            TST     DATABP          ;DOES DATA TABLE EXIST?
1538  006652  001040                    BNE     TYPDAT          ;BR IF YES.
1539  006654  104401  001313    TYPMSG: TYPE    ,$CRLF
1540  006660  104401  001313            TYPE    ,$CRLF
1541  006664  005737  001444            TST     LOCK
1542  006670  001402                    BEQ     1$
1543  006672  104401  010015            TYPE    ,MASTEK
1544  006676  104401  010003    1$:     TYPE    ,MTSTN
1545  006702  104417  007120            CNVRT   ,XTSTN          ;SHOW IT
1546  006706  104401  010072            TYPE    ,MERRPC         ;TYPE PC.
1547  006712  104417  007112            CNVRT   ,ERTABO         ;SHOW IT
1548  006716  104401  001313            TYPE    ,$CRLF          ;GIVE A CR/LF
1549  006722  112737  177777  001203    MOVB    #-1,$ERFLG      ;NO MORE HEADER UNLESS NO DATA TABLE.
1550  006730  005737  006740            TST     ERRMSG          ;IS THERE AN ERROR MESSAGE?
1551  006734  001402                    BEQ     WRKO.FM         ;BR IF NO.
1552  006736  104401                    TYPE                    ;TYPE
1553  006740  000000            ERRMSG: 0                       ;    ERROR MESSAGE
1554  006742                    WRKO.FM:
1555  006742  005737  006752            TST     DATAHD          ;DATA HEADER?
```

```
1556  006746  001402                        BEQ     TYPDAT          ;BR IF NO
1557  006750  104401                        TYPE                    ;TYPE
1558  006752  000000              DATAHD:   0                       ;    DATA HEADER
1559  006754  005737  006764      TYPDAT:   TST     DATABP          ;DATA TABLE?
1560  006760  001402                        BEQ     RESREG          ;BR IF NO.
1561  006762  104416                        CONVRT                  ;SHOW
1562  006764  000000              DATABP:   0                       ;    DATA TABLE
1563  006766  104407              RESREG:   RESCS                   ;RESTORE PROC REGISTERS
1564  006770  122737  000001 001336 HALTS:  CMPB    #APTENV,$ENV    ; IS APT RUNNING ?
1565  006776  001007                        BNE     3$              ; SKIP APT CALL IF NOT.
1566  007000  113737  001214 007012        MOVB    $ITEMB,6$       ; COPY ERROR #.
1567  007006  004737  004714              JSR     PC,$ATY4        ; CALL APT SERVICES.
1568  007012  000000              6$:       .WORD   0               ; ERROR # GOES HERE.
1569  007014  000777              9$:       BR      9$              ; LOCK HERE.
1570  007016  022737  004070 000042 3$:     CMP     #$ENDAD,@#42    ;IF ACT-11 AUTOMATIC MODE, HALT!!
1571  007024  001403                        BEQ     1$
1572  007026  005777  172206              TST     @SWR            ;HALT ON ERROR?
1573  007032  100005                        BPL     EXITER          ;BR IF NO HALT ON ERROR
1574  007034  010046              1$:       PUSHRO                  ;SAVE R0
1575  007036  016600  000002              MOV     2(SP),R0        ;SHOW ERROR PC IN DATA LIGHTS
1576  007042  000000                        HALT                    ;HALT
1577  007044  012600                        POPR0                   ;GET R0
1578  007046  005237  001212      EXITER:   INC     $ERTTL          ;UPDATE ERROR COUNT
1579  007052  032777  000400 172160        BIT     #SW08,@SWR      ;GOTO TOP OF TEST?
1580  007060  001007                        BNE     1$              ;BR IF YES
1581  007062  032777  002000 172150        BIT     #SW10,@SWR      ;GOTO NEXT TEST?
1582  007070  001407                        BEQ     2$              ;BR IF NO
1583  007072  013737  001442 001206        MOV     NEXT,$LPADR     ;SET FOR NEXT TEST
1584  007100  012706  001200              MOV     #STACK,SP       ;RESET SP
1585  007104  000177  172076              JMP     @$LPADR         ;GOTO SPECIFIED TEST
1586  007110  000002              2$:       RTI                     ;$LPADR
1587  007112  000001              ERTABO:   1
1588  007114     006     002              .BYTE   6,2
1589  007116  001460                        SAVPC
1590  007120  000001              XTSTN:    1
1591  007122     003     002              .BYTE   3,2
1592  007124  001202                        $TSTNM
1593                                        ;ENTER HERE ON POWER FAILURE
1594                                        ;--------------------------
1595
1596                              .SBTTL  POWER DOWN AND UP ROUTINES
1597
1598                              ;;*****************************************************************
1599                              ;POWER DOWN ROUTINE
1600  007126  012737  007316 000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
1601  007134  012737  000340 000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
1602  007142  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1603  007144  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1604  007146  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1605  007150  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1606  007152  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1607  007154  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1608  007156  017746  172056              MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
1609  007162  010637  007322              MOV     SP,$SAVR6       ;;SAVE SP
1610  007166  012737  007200 000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
1611  007174  000000                        HALT
```

```
1612  007176  000776                        BR      .-2             ;;HANG UP
1613
1614                                  ;;*****************************************************************
1615                                  ;POWER UP ROUTINE
1616  007200  012737  007316  000024 $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
1617  007206  013706  007322                MOV     $SAVR6,SP       ;;GET SP
1618  007212  005037  007322                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
1619  007216  005237  007322          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
1620  007222  001375                        BNE     1$              ;;OF  WORD
1621  007224  104401  007562                TYPE    ,MPFAIL         ;
1622  007230  104417  007324                CNVRT   ,PFTAB          ;
1623  007234  105037  001203                CLRB    $ERFLG          ;CLEAR ERROR FLAG.
1624  007240  005037  001216                CLR     $ERRPC          ; CLEAR LAST ERROR PC
1625  007244  013701  002066                MOV     KMCSR,R1        ; RESTORE DEVICE ADDRESS.
1626  007250  005011                        CLR     (R1)            ; CLEAR THE CSR.
1627  007252  104410                        MSTCLR                  ;
1628  007254  012677  171760                MOV     (SP)+,@SWR      ;POP STACK INTO @SWR
1629  007260  012605                        MOV     (SP)+,R5        ;POP STACK INTO R5
1630  007262  012604                        MOV     (SP)+,R4        ;POP STACK INTO R4
1631  007264  012603                        MOV     (SP)+,R3        ;POP STACK INTO R3
1632  007266  012602                        MOV     (SP)+,R2        ;POP STACK INTO R2
1633  007270  012601                        MOV     (SP)+,R1        ;POP STACK INTO R1
1634  007272  012600                        MOV     (SP)+,R0        ;POP STACK INTO R0
1635  007274  012737  007126  000024        MOV     #SPWRON,@#PWRVEC ;SET UP THE POWER DOWN VECTOR
1636  007302  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
1637  007310  104401                        TYPE                    ;;REPORT THE POWER FAILURE
1638  007312  007562          $PWRMG: .WORD   MPFAIL                  ;;POWER FAIL MESSAGE POINTER
1639  007314  000002                        RTI
1640  007316  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
1641  007320  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
1642  007322  000000          $SAVR6: 0                               ;;PUT THE SP HERE
1643
1644  007324  000001          PFTAB:  1
1645  007326     003     002  .BYTE   3,2
1646  007330  001202                        $TSTNM
1647
1648  007332                  .DELAY:
1649  007332  012777  000020  172534        MOV     #20,@KMP04
1650  007340  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1651  007342  121111                        121111                  ;POKE CLOCK DELAY BIT
1652  007344                          1$:
1653  007344  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1654  007346  121224                        121224                  ;PORT4←IBUS#11
1655  007350  032777  000020  172516        BIT     #BIT4,@KMP04    ;IS CLOCK BIT SET?
1656  007356  001772                        BEQ     1$              ;BR IF NO
1657  007360  000002                        RTI
1658
1659  007362                  .MSTCLR:
1660  007362  152777  000100  172500        BISB    #BIT6,@KMCSRH   ;SET MASTER CLEAR
1661  007370  142777  000300  172472        BICB    #BIT6!BIT7,@KMCSRH  ;CLEAR MASTER CLEAR AND RUN
1662  007376  000002                        RTI                     ;RETURN
1663
1664  007400                  .ROMCLK:
1665  007400  152777  000002  172462        BISB    #BIT1,@KMCSRH   ;SET ROMI
1666  007406  013677  172464                MOV     @(SP)+,@KMP06   ;LOAD INSTRUCTION IN SEL6
1667  007412  062746  000002                ADD     #2,-(SP)        ;ADJUST STACK
```

```
1668  007416  032777  000100  171614          BIT     #SW06,@SWR      ;HALT IF SW06 =1
1669  007424  001401                          BEQ     1$              ;BR IF SW06 =0
1670  007426  000000                          HALT                    ;HALT BEFORE CLOCKING INSTRUCTION
1671  007430  152777  000003  172432  1$:     BISB    #BIT1!BIT0,@KMCSRH ;CLOCK INSTRUCTION
1672  007436  142777  000007  172424          BICB    #BIT2!BIT1!BIT0,@KMCSRH  ;CLEAR ROM0, ROM1, STEP
1673  007444  000002                          RTI
1674
1675  007446                          .DATACLK:
1676  007446  013637  011106                  MOV     @(SP)+,TEMP     ;PUT TICK COUNT IN TEMP
1677  007452  062746  000002                  ADD     #2,-(SP)        ;ADJUST STACK
1678  007456  152777  000020  172404  1$:     BISB    #BIT4,@KMCSRH   ;SET STEP LU
1679  007464  027777  172376  172374          CMP     @KMCSR,@KMCSR   ;WASTE TIME
1680  007472  142777  000020  172370          BICB    #BIT4,@KMCSRH   ;CLEAR STEP LU
1681  007500  005337  011106                  DEC     TEMP            ;DEC TICK COUNT
1682  007504  001364                          BNE     1$              ;BR IF NOT DONE
1683  007506  000002                          RTI                     ;RETURN
1684  007510  000001                  3$:     .BLKW 1
1685
1686  007512                          .TIMER:
1687  007512  013637  011106                  MOV     @(SP)+,TEMP     ;MOVE COUNT TO TEMP
1688  007516  062746  000002                  ADD     #2,-(SP)        ;ADJUST STACK
1689  007522                          1$:
1690  007522  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCT1 +1, ROMCLK PC=5304
1691  007524  021364                          021364                  ;PORT4+IBUS# REG11
1692  007526  032777  000002  172340          BIT     #2,@KMP04       ;IS PGM CLOCK BIT CLEAR?
1693  007534  001772                          BEQ     1$              ;BR IF YES
1694  007536                          2$:
1695  007536  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1696  007540  021364                          021364                  ;PORT4+IBUS# REG11
1697  007542  032777  000002  172324          BIT     #2,@KMP04       ;IS PGM CLOCK BIT SET?
1698  007550  001372                          BNE     2$              ;BR IF YES
1699  007552  005337  011106                  DEC     TEMP            ;DEC COUNT
1700  007556  001361                          BNE     1$              ;BR IF NOT DONE
1701  007560  000002                          RTI                     ;RETURN
1702
1703  007562  050200  051127  043040  MPFAIL: .ASCIZ  <200>/PWR FAILED. RESTART AT TEST /
(2)   007620  042600  042116  050040  MEPASS: .ASCIZ  <200>/END PASS DZKCD  /
(2)   007642  051200     000  MR:     .ASCIZ  <200>/R/
(2)   007645     200  047516  042040  MERR2:  .ASCIZ  <200>/NO DEVICES PRESENT./
(2)   007672  044600  051516  043125  MERR3:  .ASCIZ  <200>/INSUFFICIENT DATA!/
(2)   007716  046200  041517  020113  MLOCK:  .ASCIZ  <200>/LOCK ON SELECTED TEST/
(2)   007745     103  051123  020072  MCSRX:  .ASCIZ   /CSR: /
(2)   007753     126  041505  020072  MVECX:  .ASCIZ  /VEC: /
(2)   007761     120  051501  042523  MPASSX: .ASCIZ  /PASSES: /
(2)   007772  051105  047522  051522  MERRX:  .ASCIZ  /ERRORS: /
(2)   010003     124  051505  020124  MTSTN:  .ASCIZ  /TEST NO: /
(2)   010015     052     000  MASTEK: .ASCIZ  /*/
(2)   010017     200  042523  020124  MNEW:   .ASCIZ  <200>/SET SWITCH REG TO KMC11'S DESIRED ACTIVE./
(2)   010072  041520  020072     000  MERRPC: .ASCIZ  /PC: /
(2)   010077     200  020040  020040  XHEAD:  .ASCII  <200>/            MAP OF KMC11 STATUS/
(2)   010136  020200  020040  020040          .ASCII  <200>/            --------------------/
(2)   010175     200  020040  041520          .ASCII  <200>/  PC      CSR     STAT1    STAT2    STAT3/
(2)   010247     200  026455  026455          .ASCIZ  <200>/------   ------   ------   ------   ------/
(2)   010323     200  047510  020127  NUM:    .ASCIZ  <200>/HOW MANY KMC11'S TO BE TESTED?/
(2)   010363     200  051503  020122  CSR:    .ASCIZ  <200>/CSR ADDRESS?/
(2)   010401     200  042526  052103  VEC:    .ASCIZ  <200>/VECTOR ADDRESS?/
```

```
   (2)  010422  041200  020122  051120  PRIO:   .ASCIZ  <200>/BR PRIORITY LEVEL? (4,5,6,7)?/
   (2)  010461     200  044127  041511  MODU:   .ASCIZ  <200>/WHICH LINE UNIT? IF NONE TYPE "N", IF M8201 TYPE "1", IF M8202 TYP
   (2)  010573     200  053523  052111  LINE:   .ASCIZ  <200>/SWITCH PAC#1 (DDCMP LINE #)?/
   (2)  010631     200  053523  052111  BM:     .ASCIZ  <200>/SWITCH PAC#2 (BM873 BOOT ADD)?/
   (2)  010671     200  051511  052040  CONN:   .ASCIZ  <200>/IS THE LOOP BACK CONNECTOR ON?/
   (2)  010731     200  047516  042040  NOACT:  .ASCIZ  <200>/NO DEVICES ARE SELECTED/
   (2)  010762  100200  046513  030503  CONERR: .ASCIZ  <200><200>/KMC11 AT NONSTANDARD ADDRESS  PC: /
   (2)  011027     200  054105  042520  CNERR:  .ASCIZ  <200>/EXPECTED  FOUND/
   (2)  011050  024040  046513  024503  KMCM:   .ASCIZ  / (KMC) /
   (2)                                  .EVEN
   (2)  011060  000005                  XSTATQ: 5
1704    011062     006     003                  .BYTE   6,3
1705    011064  001276                          $TMP0
1706    011066     006     003                  .BYTE   6,3
1707    011070  001300                          $TMP1
1708    011072     006     003                  .BYTE   6,3
1709    011074  001302                          $TMP2
1710    011076     006     003                  .BYTE   6,3
1711    011100  001304                          $TMP3
1712    011102     006     002                  .BYTE   6,2
1713    011104  001306                          $TMP4
1714                                 .EVEN
1715
1716                                         ;BUFFERS FOR INPUT-OUTPUT
1717
1718    011106  000000                  TEMP:   0
1719            011150                  .=.+40
1720    011150  000000                  MDATA:  0
1721            011212                  .=.+40
1722
1723
1724                                         ;ROUTINE USED TO CHANGE SOFTWARE SWITCH
1725                                         ;REGISTER USING THE CONSOLE TERMINAL
1726                                         ;-----------------------------------
1727
1728    011212  022737  000176  001240  CKSWR:  CMP     #SWREG,SWR      ;IS THE SOFT SWR BEING USED?
1729    011220  001075                          BNE     CKSWR5          ;BR IF NO
1730    011222  132737  000001  001336          BITB    #1,$ENV         ; IS IT RUNNING UNDER APT?
1731    011230  001071                          BNE     CKSWR5          ; EXIT IF YES.
1732    011232  022777  000007  170006          CMP     #7,@$TKB        ;WAS CTRL G TYPED? (7 BIT ASCII)
1733    011240  001404                          BEQ     1$              ;BR IF YES
1734    011242  022777  000207  167776          CMP     #207,@$TKB      ;WAS CTRL G TYPED? (8 BIT ASCII)
1735    011250  001061                          BNE     CKSWR5          ;BR IF NO
1736    011252  010246                  1$:     MOV     R2,-(SP)        ;STORE R2
1737    011254  010346                          MOV     R3,-(SP)        ;STORE R3
1738    011256  010446                          MOV     R4,-(SP)        ;STORE R4
1739    011260  012737  177777  011416          MOV     #-1,SWFLG       ;SET SOFT TYPE OUT FLAG
1740    011266  005002                  CKSWR1: CLR     R2              ;CLEAR NEW SWR CONTENTS
1741    011270  012704  177777                  MOV     #-1,R4          ;SET FLAG TO ALL ONES
1742    011274  104401  005541                  TYPE    ,$MSWR          ;TYPE "SWR= "
1743    011300  104417                  CKSWR2: CNVRT                   ;TYPE OUT PRESENT CONTENTS
1744    011302  011452                          SOFTSW                  ;OF SOFT SWITCH REGISTER
1745    011304  104401  005552          CKSWR3: TYPE    ,$MNEW          ;TYPE "NEW? "
1746    011310  004737  011420          CKSWR4: JSR     PC,INCHAR       ;GET RESPONSE
1747    011314  022703  000015                  CMP     #15,R3          ;WAS IT A CR?
1748    011320  001424                          BEQ     5$              ;BR IF YES
```

```
1749  011322  022703  000012              CMP     #12,R3          ;WAS IT A LF?
1750  011326  001416                      BEQ     4$              ;BR IF YES
1751  011330  022703  000025              CMP     #25,R3          ;WAS IT CTRL U?
1752  011334  001754                      BEQ     CKSWR1          ;BR IF YES(START OVER)
1753  011336  022703  000007              CMP     #7,R3           ;IF CNTL G GET NEXT CHAR
1754  011342  001762                      BEQ     CKSWR4
1755  011344  005004                      CLR     R4              ;IT MUST BE A DIGIT SO CLR FLAG
1756  011346  042703  177770              BIC     #177770,R3      ;ONLY 0-7 ARE LEGAL SO MASK OFF BITS
1757  011352  006302                      ASL     R2              ;SHIFT R2 3 TIMES
1758  011354  006302                      ASL     R2
1759  011356  006302                      ASL     R2
1760  011360  050302                      BIS     R3,R2           ;ADD LAST DIGIT
1761  011362  000752                      BR      CKSWR4          ;GET NEXT CHARACTER
1762  011364  012766  002402  000006  4$: MOV     #.START,6(SP)   ;LF WAS TYPED SO GO TO START
1763  011372  005704                  5$: TST     R4              ;IS FLAG CLEAR?
1764  011374  001002                      BNE     6$              ;IF NOT DON'T CHANGE SOFT SWR
1765  011376  010277  167636              MOV     R2,@SWR         ;IF YES THEN WRITE NEW CONTENTS TO SOFT SWR
1766  011402  005037  011416          6$: CLR     SWFLG           ;CLEAR TYPEOUT FLAG
1767  011406  012604                      MOV     (SP)+,R4        ;RESTORE R4
1768  011410  012603                      MOV     (SP)+,R3        ;RESTORE R3
1769  011412  012602                      MOV     (SP)+,R2        ;RESTORE R2
1770  011414  000207              CKSWR5: RTS     PC              ;RETURN
1771
1772  011416  000000              SWFLG:  0
1773
1774  011420  105777  167620      INCHAR: TSTB    @#STKS
1775  011424  100375                      BPL     .-4
1776  011426  017703  167614              MOV     @#STKB,R3
1777  011432  105777  167612              TSTB    @#STPS
1778  011436  100375                      BPL     .-4
1779  011440  010377  167606              MOV     R3,@#STPB
1780  011444  042703  000200              BIC     #BIT7,R3
1781  011450  000207                      RTS     PC
1782
1783  011452  000001              SOFTSW: 1
1784  011454     006     002              .BYTE   6,2
1785  011456  000176                      SWREG
```

```
1786
1787                                        ;
1788                                        ;ROUTINE USED TO "CYCLE" THROUGH UP TO 16 KMC11'S
1789                                        ;THIS ROUTINE SETS UP THE CONTROL ADDRESS FOR THE DIAGNOSTIC
1790                                        ;AND RUNS THE SPECIFIED KMC11'S.   THIS ROUTINE *MUST*
1791                                        ;BE RUN FIRST BEFORE ENTERING THE DIAGNOSTIC FOR THE
1792                                        ;SETUP NECESSARY.
1793                                        ;
1794
1795  011460  005737  001470        CYCLE:  TST     KMACTV          ;ARE ANY KMC11'S TO BE TESTED?
1796  011464  001004                        BNE     1$              ;BR IF OK.
1797  011466  104401  010731                TYPE    ,NOACT          ;NO KMC11'S SELECTED!!
1798  011472  000000                        HALT                    ;STOP THE SHOW.
1799  011474  000776                        BR      .-2             ;DISQUALIFY CONT. SW.
1800  011476  000241                1$:     CLC                     ;CLEAR PROC. CARRY BIT.
1801  011500  006137  001500                ROL     RUN             ;UPDATE POINTER
1802  011504  005537  001500                ADC     RUN             ;CATCH CARRY FROM RUN
1803  011510  062737  000004  001504        ADD     #4,MILK         ;UPDATE POINTER
1804  011516  062737  000010  001502        ADD     #10,CREAM       ;UPDATE ADDRESS POINTER.
1805  011524  022737  002300  001502        CMP     #KM.MAP+200,CREAM
1806  011532  001006                        BNE     2$              ;KEEP GOING; NOT ALL TESTED FOR.
1807  011534  012737  002100  001502        MOV     #KM.MAP,CREAM   ;RESET ADDRESS POINTER.
1808  011542  012737  002302  001504        MOV     #CNT.MAP,MILK   ;RESET PASS COUNT POINTER
1809  011550  033737  001500  001470  2$:    BIT     RUN,KMACTV      ;IS THIS ONE ACTIVE?
1810  011556  001747                        BEQ     1$              ;BR IF NO
1811  011560  013700  001502                MOV     CREAM,R0        ;GET ADDRESS POINTER
1812  011564  013702  001504                MOV     MILK,R2         ;GET PASS COUNT POINTER
1813  011570  012037  002066                MOV     (R0)+,KMCSR     ;LOAD SYSTEM CTRL. REG
1814  011574  011037  002056                MOV     (R0),KMRVEC     ;LOAD VECTOR
1815  011600  042737  177000  002056        BIC     #177000,KMRVEC  ;CLEAR UNWANTED BITS
1816  011606  012037  002050                MOV     (R0)+,STAT1     ;LOAD STAT1
1817  011612  012037  002052                MOV     (R0)+,STAT2     ;LOAD STAT2
1818  011616  012037  002054                MOV     (R0)+,STAT3     ;LOAD STAT3
1819  011622  012237  001324                MOV     (R2)+,$PASS     ;LOAD PASS COUNT
1820  011626  012237  001212                MOV     (R2)+,$ERTTL    ;LOAD ERROR COUNT
1821  011632  012700  000002                MOV     #2,R0           ;SAVE CORE THIS WAY!
1822  011636  013737  002066  002070        MOV     KMCSR,KMCSRH
1823  011644  005237  002070                INC     KMCSRH
1824  011650  013737  002070  002072        MOV     KMCSRH,KMCTL
1825  011656  005237  002072                INC     KMCTL
1826  011662  013737  002072  002074        MOV     KMCTL,KMP04
1827  011670  060037  002074                ADD     R0,KMP04
1828  011674  013737  002074  002076        MOV     KMP04,KMP06
1829  011702  060037  002076                ADD     R0,KMP06
1830
1831  011706  013737  002056  002060        MOV     KMRVEC,KMRLVL   ;PTY LVL
1832  011714  060037  002060                ADD     R0,KMRLVL       ;
1833  011720  013737  002060  002062        MOV     KMRLVL,KMTVEC   ;TX VEC
1834  011726  060037  002062                ADD     R0,KMTVEC       ;
1835  011732  013737  002062  002064        MOV     KMTVEC,KMTLVL   ;TX LVL
1836  011740  060037  002064                ADD     R0,KMTLVL
1837
1838  011744  032737  000002  001446        BIT     #SW01,STRTSW    ;IS TEST NO. SELECTED
1839  011752  001447                        BEQ     7$              ;BR IF NO
1840  011754                        4$:
1841  011754  005737  000042                TST     @#42            ;RUNNING IN AUTO MODE?
```

```
1842  011760  001044                        BNE     7$              ;BR IF YES
1843  011762  104401  001313                TYPE    ,$CRLF
1844  011766  104415                        INPUT
1845  011770  010003                        MTSTN
1846  011772  000001                        1
1847  011774  001000                        1000
1848  011776  001202                        $TSTNM
1849  012000     000          .BYTE   0
1850  012001     001          .BYTE   1
1851  012002  012700  013732                MOV     #TST1,R0
1852  012006  022710           5$:   CMP     (PC)+,(R0)      ;CMP FIRST WORD TO 12737
1853  012010  012737                        MOV     (PC)+,@(PC)+
1854  012012  001020                        BNE     6$              ;BR IF NOT SAME
1855  012014  023760  001202  000002        CMP     $TSTNM,2(R0)    ;DOES $TSTNM MATCH?
1856  012022  001014                        BNE     6$              ;BR IF NO
1857  012024  022760  001202  000004        CMP     #$TSTNM,4(R0)   ;IS LAST WORD OK?
1858  012032  001010                        BNE     6$              ;BR IF NO
1859  012034  010037  001206                MOV     R0,$LPADR       ;IT IS A LEGAL TEST SO DO IT
1860  012040  104401  007642                TYPE    ,MR
1861  012044  042737  000002  001446        BIC     #SW01,STRTSW
1862  012052  000412                        BR      8$
1863  012054  005720           6$:   TST     (R0)+           ;POP R0
1864  012056  020027  020634                CMP     R0,#TLAST+10    ;AT END YET?
1865  012062  001351                        BNE     5$              ;BR IF NO
1866  012064  104401  001312                TYPE    ,$QUES          ;YES ILLEGAL TEST NO.
1867  012070  000731                        BR      4$              ;TRY AGAIN
1868
1869  012072  012737  013732  001206  7$:   MOV     #TST1,$LPADR    ;PREPARE $LPADR ADDRESS
1870  012100  013701  002066        8$:   MOV     KMCSR,R1        ;R1 = BASE KMC11 ADDRESS
1871  012104  000177  167076                JMP     @$LPADR         ;GO START TESTING.
1872
1873
1874                                    ;ROUTINE USED TO "AUTO SIZE" THE KMC11
1875                                    ;CSR AND VECTOR.
1876                                    ;NOTE:  THE CSR MAY BE ANY WHERE IN THE FLOATING
1877                                    ;       ADDRESS RANGE (160000:164000)
1878                                    ;       AND THE VECTOR MAY BE ANY WHERE IN THE
1879                                    ;       FLOATING VECTOR RANGE (300:770)
1880                                    ;
1881
1882  012110                    AUTO.SIZE:
1883  012110  000005                        RESET                   ;INSURE A BUS INIT.
1884  012112  012702  002100    CSRMAP:     MOV     #KM.MAP,R2      ;LOAD MAP POINTER.
1885  012116  005022            1$:   CLR     (R2)+           ;ZERO ENTIRE MAP
1886  012120  022702  002300                CMP     #KM.END,R2      ;ALL DONE?
1887  012124  001374                        BNE     1$              ;BR IF NO
1888  012126  005037  001472                CLR     KMNUM           ;SET OCTAL NUMBER OF KMC11'S TO 0
1889  012132  012702  002100                MOV     #KM.MAP,R2      ;R2 POINTS TO KMC MAP
1890  012136  005037  001470                CLR     KMACTV          ;CLEAR ACTIVE
1891  012142  032737  000001  001446        BIT     #SW00,STRTSW    ;QUESTIONS?
1892  012150  001002                        BNE     .+6             ;BR IF YES
1893  012152  000137  012532                JMP     7$              ;IF NO SKIP QUESTIONS
1894  012156  012737  000001  001306        MOV     #1,$TMP4        ;START WITH 1
1895  012164  104415                        INPUT
1896  012166  010323                        NUM
1897  012170  000001                        1
```

```
1898  012172  000020                          16.
1899  012174  001302                          $TMP2
1900  012176     000                          .BYTE   0
1901  012177     001                          .BYTE   1
1902  012200  013737  001302  001472         MOV     $TMP2,KMNUM     ;KMNUM = HOW MANY
1903  012206  104401  001313          12$:    TYPE    ,$CRLF
1904  012212  104416                          CONVRT                  ;TYPE WHICH KMC IS BEING DONE
1905  012214  013164                          WHICH                   ;$TMP4 IS WHICH KMC
1906  012216  005237  001306                  INC     $TMP4
1907  012222  104415                          INPUT
1908  012224  010363                          CSR
1909  012226  160000                          160000
1910  012230  164000                          164000
1911  012232  001304                          $TMP3
1912  012234     000                          .BYTE   0
1913  012235     001                          .BYTE   1
1914  012236  013722  001304                  MOV     $TMP3,(R2)+     ;STORE CSR IN MAP
1915  012242  104415                          INPUT
1916  012244  010401                          VEC
1917  012246  000000                          0
1918  012250  000776                          776
1919  012252  001304                          $TMP3
1920  012254     000                          .BYTE   0
1921  012255     001                          .BYTE   1
1922  012256  013712  001304                  MOV     $TMP3,(R2)      ;STORE VECTOR IN MAP
1923  012262  104401                  10$:    TYPE
1924  012264  010422                          PRIO                    ;ASK WHAT BR LEVEL
1925  012266  004737  013456                  JSR     PC,INTTY        ;GET RESPONSE
1926  012272  022703  000024                  CMP     #24,R3          ;
1927  012276  101014                          BHI     50$             ;BR IF LESS THAN 4
1928  012300  022703  000027                  CMP     #27,R3          ;
1929  012304  103411                          BLO     50$             ;BR IF GREATER THAN 7
1930  012306  012704  000011                  MOV     #11,R4          ;R4 = NUMBER OF SHIFTS
1931  012312  006303                          ASL     R3              ;SHIFT R3 LEFT
1932  012314  005304                          DEC     R4              ;DEC SHIFT COUNT
1933  012316  001375                          BNE     .-4             ;BR IF NOT DONE
1934  012320  042703  170777                  BIC     #170777,R3      ;BIC UNWANTED BITS
1935  012324  050312                          BIS     R3,(R2)         ;PUT BR LEVEL IN STATUS MAP
1936  012326  000403                          BR      8$              ;CONTINUE
1937  012330  104401                  50$:    TYPE
1938  012332  001312                          $QUES                   ;RESPONSE IS OUT OF LIMITS
1939  012334  000752                          BR      10$             ;TRY AGAIN
1940  012336                          8$:
1941  012336                          9$:
1942  012336  104401                  16$:    TYPE
1943  012340  010461                          MODU                    ;ASK WHICH LINE UNIT
1944  012342  004737  013456                  JSR     PC,INTTY        ;GET REPLY
1945  012346  022703  000021                  CMP     #21,R3          ;"1"
1946  012352  001417                          BEQ     30$
1947  012354  022703  000022                  CMP     #22,R3          ;"2"
1948  012360  001412                          BEQ     31$
1949  012362  022703  000116                  CMP     #116,R3         ;"N"
1950  012366  001403                          BEQ     32$
1951  012370  104401                          TYPE
1952  012372  001312                          $QUES                   ;IF NOT A 1,2 OR N TYPE "?"
1953  012374  000760                          BR      16$             ;TRY AGIAN
```

```
1954  012376  052722  010000          32$:    BIS     #BIT12,(R2)+    ;SET BIT 12 IN STAT2 IF NO LU
1955  012402  062222                          CMP     (R2)+,(R2)+     ;POP OVER STAT2 AND STAT3
1956  012404  000445                          BR      33$
1957  012406  052712  020000          31$:    BIS     #BIT13,(R2)     ;SET BIT 13 IN STAT2 IF M8202
1958  012412  104401                  30$:    TYPE
1959  012414  010671                          CONN                    ;ASK IF LOOP-BACK IS ON
1960  012416  004737  013456                  JSR     PC,INTTY        ;GET REPLY
1961  012422  022703  000131                  CMP     #131,R3         ;Y
1962  012426  001406                          BEQ     17$
1963  012430  022703  000116                  CMP     #116,R3         ;N
1964  012434  001406                          BEQ     18$
1965  012436  104401                          TYPE
1966  012440  001312                          SQUES                   ;IF NOT Y OR N TYPE "?"
1967  012442  000763                          BR      30$             ;TRY AGAIN
1968  012444  052722  040000          17$:    BIS     #BIT14,(R2)+    ;TURNAROUND IS CONNECTED
1969  012450  000402                          BR      19$
1970  012452  042722  040000          18$:    BIC     #BIT14,(R2)+    ;NO TURNAROUND
1971  012456                          19$:
1972  012456  104415                          INPUT
1973  012460  010573                          LINE
1974  012462  000000                          0
1975  012464  000377                          377
1976  012466  001304                          $TMP3
1977  012470     000                          .BYTE   0
1978  012471     001                          .BYTE   1
1979  012472  113722  001304                  MOVB    $TMP3,(R2)+     ;STORE SWITCH PAC IN MAP
1980  012476  104415                          INPUT
1981  012500  010631                          BM
1982  012502  000000                          0
1983  012504  000377                          377
1984  012506  001304                          $TMP3
1985  012510     000                          .BYTE   0
1986  012511     001                          .BYTE   1
1987  012512  113722  001304                  MOVB    $TMP3,(R2)+     ;STORE SWITCH PAC IN MAP
1988  012516  005722                          TST     (R2)+           ;POP OVER STAT3
1989  012520  005337  001302          33$:    DEC     $TMP2           ;DEC KMC COUNT
1990  012524  001230                          BNE     12$             ;BR IF MORE TO DO
1991  012526  000137  013064                  JMP     13$             ;CONTINUE
1992  012532  012701  160000          7$:     MOV     #160000,R1      ;SET FOR FIRST ADDRESS TO BE TESTED
1993  012536  012737  013156  000004          MOV     #6$,@#4         ;SET FOR NON-EXISTANT DEVICE TIME OUT
1994  012544  005011                  2$:     CLR     (R1)            ;CLEAR SEL0
1995  012546  005711                          TST     (R1)            ;IF KMC11 KMCSR S/B 0
1996  012550  001135                          BNE     3$              ;IF NO DEV ; TRAP TO 4. IF NO BIT 8 THEN NO KMC11
1997  012552  005061  000006                  CLR     6(R1)           ;CLEAR SEL6
1998  012556  005761  000006                  TST     6(R1)           ;IF KMC11 THEN KMRIC S/B =0!
1999  012562  001130                          BNE     3$              ;BR IF NOT KMC11
2000  012564  012711  002000                  MOV     #BIT10,(R1)     ;SET ROMO
2001  012570  005061  000004                  CLR     4(R1)           ;CLEAR SEL4
2002  012574  012761  125252  000006          MOV     #125252,6(R1)   ;WRITE THIS TO SEL6
2003  012602  052711  020000                  BIS     #BIT13,(R1)     ;WRITE IT!
2004  012606  022761  125252  000004          CMP     #125252,4(R1)   ;WAS IT WRITTEN?
2005  012614  001113                          BNE     3$              ;IF NO IT IS NOT CRAM
2006                          ;AT THIS POINT IT IS ASSUMED THAT R1 HOLDS A KMC11 CSR ADDRESS.
2007  012616                  21$:
2008  012616  010122                  22$:    MOV     R1,(R2)+        ;STORE CSR IN CORE TABLE.
2009  012620  012711  001000          15$:    MOV     #BIT9,(R1)      ;CLEAR LINE UNIT LOOP
```

```
2010  012624  005061  000004          CLR     4(R1)             ;CLEAR PORT4
2011  012630  012761  122113  000006  MOV     #122113,6(R1)     ;LOAD INSTRUCTION (CLR DTR)
2012  012636  052711  000400          BIS     #BIT8,(R1)        ;CLOCK INSTRUCTION
2013  012642  012761  021264  000006  MOV     #021264,6(R1)     ;LOAD INSTRUCTION
2014  012650  052711  000400          BIS     #BIT8,(R1)        ;CLOCK INSTRUCTION
2015  012654  122761  000377  000004  CMPB    #377,4(R1)        ;IS IT ALL ONES?
2016  012662  001003                  BNE     .+10              ;BR IF NO
2017  012664  052712  010000          BIS     #BIT12,(R2)       ;IF YES, NO LINE UNIT, SET STATUS BIT
2018  012670  000436                  BR      20$
2019  012672  032761  000002  000004  BIT     #BIT1,4(R1)       ;IS SWITCH A ONE?
2020  012700  001403                  BEQ     .+10              ;BR IF MR201
2021  012702  052712  060000          BIS     #BIT13!BIT14,(R2) ;MR202 ASSUME CONNECTOR
2022  012706  000427                  BR      20$               ;CONNECTOR ON)
2023  012710  032761  000010  000004  BIT     #BIT3,4(R1)       ;IS MRDY SET
2024  012716  001023                  BNE     20$               ;BR IF MR201 NO CONNECTOR (ON LINE)
2025  012720  012761  000100  000004  MOV     #BIT6,4(R1)       ;LOAD PORT4
2026  012726  012761  122113  000006  MOV     #122113,6(R1)     ;LOAD INSTRUCTION
2027  012734  052711  000400          BIS     #BIT8,(R1)        ;CLOCK INSTRUCTION(SET DTR)
2028  012740  012761  021264  000006  MOV     #021264,6(R1)     ;LOAD INSTRUCTION
2029  012746  052711  000400          BIS     #BIT8,(R1)        ;CLOCK INSTRUCTION(READ MODEM REG)
2030  012752  032761  000010  000004  BIT     #BIT3,4(R1)       ;IS MRDY SET NOW?
2031  012760  001402                  BEQ     20$               ;BR IF NO CONNECTOR
2032  012762  052712  040000          BIS     #BIT14,(R2)       ;SET STATUS BIT FOR CONNECTOR
2033  012766  005722          20$:    TST     (R2)+             ;POP POINTER
2034  012770  012761  021324  000006  MOV     #021324,6(R1)     ;PUT INSTRUCTION IN PORT6
2035  012776  012711  001400          MOV     #BIT9!BIT8,(R1)   ;PORT4=LU 15
2036  013002  156122  000004          BISB    4(R1),(R2)+       ;STORE DDCMP LINE # IN TABLE
2037  013006  012761  021344  000006  MOV     #021344,6(R1)     ;PORT6=INSTRUCTION
2038  013014  012711  001400          MOV     #BIT8!BIT9,(R1)   ;CLOCK INSTR.
2039  013020  156122  000004          BISB    4(R1),(R2)+       ;STORE BM873 ADD IN TABLE
2040  013024  005722                  TST     (R2)+             ;POP OVER STAT3
2041  013026  005011                  CLR     (R1)              ;CLEAR ROMI
2042  013030  005237  001472          INC     KMNUM             ;UPDATE DEVICE COUNTER
2043  013034  022737  000020  001472  CMP     #20,KMNUM         ;ARE MAX. NO. OF DEV FOUND?
2044  013042  001410                  BEQ     13$               ;YES DON'T LOOK FOR ANY MORE.
2045  013044  005011          3$:     CLR     (R1)              ;CLEAR BIT 10
2046  013046  005061  000006          CLR     6(R1)             ;CLEAR SEL 6
2047  013052  062701  000010  14$:    ADD     #10,R1            ;UPDATE CSR POINTER ADDRESS
2048  013056  022701  164000          CMP     #164000,R1
2049  013062  001230                  BNE     2$                ;BR IF MORE ADDRESS TO CHECK.
2050  013064  005037  001470  13$:    CLR     KMACTV
2051  013070  005737  001472          TST     KMNUM             ;WERE ANY KMC11'S FOUND AT ALL?
2052  013074  001423                  BEQ     5$                ;ERROR AUTO SIZER FOUND NO KMC11'S IN THIS SYS.
2053  013076  013701  001472          MOV     KMNUM,R1
2054  013102  010137  001476          MOV     R1,SAVNUM         ;SAVE NUMBER OF DEVICES
2055  013106  000241          4$:     CLC
2056  013110  006137  001470          ROL     KMACTV            ;GENERATE ACTIVE REGISTER OF DEVICES.
2057  013114  005237  001470          INC     KMACTV            ;SET THE BIT
2058  013120  005301                  DEC     R1
2059  013122  001371                  BNE     4$                ;BR IF MORE TO GENERATE
2060  013124  012737  000006  000004  MOV     #6,@#4            ;RESTORE TRAP VECTOR
2061  013132  013737  001470  001474  MOV     KMACTV,SAVACT     ;SAVE ACTIVE REGISTER
2062  013140  000137  013172          JMP     VECMAP            ;GO FIND THE VECTOR NOW.
2063  013144  104401  007645  5$:     TYPE    ,MERR2            ;NOTIFY OPR THAT NO KMC11'S FOUND.
2064  013150  005000                  CLR     R0                ;MAKE DATA LIGHTS ZERO
2065  013152  000000                  HALT                      ;STOP THE SHOW
```

```
2066  013154  000776                          BR      .-2             ;DISABLE CONT. SW.
2067  013156  012716  013052          6$:     MOV     #14$,(SP)       ;ENTERED BY NON-EXISTANT TIME-OUT.
2068  013162  000002                          RTI                     ;RETURN TO MAINSTREAM
2069
2070  013164  000001                  WHICH:  1
2071  013166    002     002                   .BYTE   2,2
2072  013170  001306                          $TMP4
2073
2074  013172  032737  000001  001446  VECMAP: BIT     #SW00,STRTSW
2075  013200  001114                          BNE     5$
2076  013202  012737  000340  000022          MOV     #340,@#22       ;SET IOT TRAP PRIO TO 7
2077  013210  012737  013364  000020          MOV     #4$,@#20        ;SET IOT TRAP VECTOR
2078  013216  012702  002100                  MOV     #KM.MAP,R2      ;SET SOFTWARE POINTER
2079  013222  012700  000300                  MOV     #300,R0         ;FLOATING VECTORS START HERE.
2080  013226  012701  000302                  MOV     #302,R1         ;PC OF IOT INSTR.
2081  013232  010120                  1$:     MOV     R1,(R0)+        ;START FILLING VECTOR AREA
2082  013234  012721  000004                  MOV     #4,(R1)+        ;WITH .+2; IOT
2083  013240  022021                          CMP     (R0)+,(R1)+     ;ADD 2 TO R0 +R1
2084  013242  020127  001000                  CMP     R1,#1000
2085  013246  101771                          BLOS    1$              ;BR IF MORE TO FILL
2086  013250  013737  001470  001276          MOV     KMACTV,$TMP0    ;STORE TEMPORALLY
2087  013256  006037  001276          2$:     ROR     $TMP0           ;BRING OUT A BIT
2088  013262  103063                          BCC     5$              ;BR IF ALL DONE
2089  013264  012704  000012                  MOV     #12,R4          ;R4 IS INDEX REGISTER
2090  013270  016437  013442  177776          MOV     BRLVL(R4),PS    ;SET PS TO 7
2091  013276  011201                          MOV     (R2),R1
2092  013300  012761  000200  000004          MOV     #200,4(R1)
2093  013306  012711  001000                  MOV     #BIT9,(R1)      ;SET ROMI
2094  013312  012761  121111  000006          MOV     #121111,6(R1)   ;PUT INSTRUCTION IN PORT6
2095  013320  012711  001400                  MOV     #BIT9!BIT8,(R1) ;FORCE AN INTERRUPT
2096  013324  105200                  7$:     INCB    R0              ;STALL
2097  013326  001376                          BNE     .-2             ;FOR TIME TO INTERUPT
2098  013330  162704  000002                  SUB     #2,R4           ;GET NEXT LOWEST PS LEVEL
2099  013334  001404                          BEQ     6$              ;BR IF R4 = 0
2100  013336  016437  013442  177776          MOV     BRLVL(R4),PS    ;MOVE NEXT LOWER LEVEL IN PS
2101  013344  000767                          BR      7$              ;BR TO DELAY
2102  013346  052762  005300  000002  6$:     BIS     #5300,2(R2)     ;NO INTERUPT ASSUME 300 AT LEVEL 5 AND FIX KMC11 LATER
2103  013354  005011                  3$:     CLR     (R1)            ;CLEAR ROMI
2104  013356  062702  000010                  ADD     #10,R2          ;POP SOFTWARE POINTER
2105  013362  000735                          BR      2$              ;KEEP GOING
2106  013364  051662  000002          4$:     BIS     (SP),2(R2)      ;GET VECTOR ADDRESS
2107  013370  042762  000007  000002          BIC     #7,2(R2)        ;CLEAR JUNK
2108  013376  016405  013444                  MOV     BRLVL+2(R4),R5  ;GET BR LEVEL OF KMC11
2109  013402  006305                          ASL     R5              ;SHIFT LEVEL 4 PLACES
2110  013404  006305                          ASL     R5              ;TO THE LEFT FOR THE
2111  013406  006305                          ASL     R5              ;STATUS TABLE
2112  013410  006305                          ASL     R5
2113  013412  042705  170777                  BIC     #170777,R5      ;CLEAR UNWANTED BITS
2114  013416  050562  000002                  BIS     R5,2(R2)        ;PUT BR LEVEL IN STATUS TABLE
2115  013422  022626                          CMP     (SP)+,(SP)+     ;POP IOT JUNK OFF STACK
2116  013424  012716  013354                  MOV     #3$,(SP)        ;SET FOR RETURN
2117  013430  000002                          RTI
2118  013432  012737  004134  000020  5$:     MOV     #$SCOPE,@#20    ; RESTORE SCOPE VECTOR
2119  013440  000207                          RTS     PC              ;ALL DONE WITH "AUTO SIZING"
2120
2121  013442  000000                  BRLVL:  PRO     ;LEVEL 0
```

```
2122  013444  000000                        PR0     ;LEVEL 0
2123  013446  000200                        PR4     ;LEVEL 4
2124  013450  000240                        PR5     ;LEVEL 5
2125  013452  000300                        PR6     ;LEVEL 6
2126  013454  000340                        PR7     ;LEVEL 7
2127
2128
2129  013456  105777  165562      INTTY:    TSTB    @#TKS           ;WAIT FOR DONE
2130  013462  100375                        BPL     .-4
2131  013464  017703  165556                MOV     @#TKB,R3        ;PUT CHAR IN R3
2132  013470  105777  165554                TSTB    @#TPS           ;WAIT UNTIL PRINTER IS READY
2133  013474  100375                        BPL     .-4
2134  013476  010377  165550                MOV     R3,@#TPB        ;ECHO CHAR
2135  013502  042703  000240                BIC     #BIT7!BIT5,R3   ;MASK OFF LOWER CASE
2136  013506  000207                        RTS     PC              ;RETURN
2137
2138  013510                        APT.SIZE:
2139  013510  000005                        RESET
2140  013512  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2141  013514  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2142  013516  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2143  013520  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2144  013522  005037  013724                CLR     VECTR           ; CLEAR THE LOCAL VARIABLE
2145  013526  005037  013730                CLR     PRIRTY          ; CLEAN UP LOCAL VARIABLE
2146  013532  013700  001376                MOV     $CDW1,R0        ; GET THE DEVICE COUNT
2147  013536  010037  001476                MOV     R0,SAVNUM       ; SAVE THE NO. OF DEVICES
2148  013542  012701  001346                MOV     #[illegible]S1,R1       ; GET EXTRA INFO, BITS POINTER
2149  013546  013737  001372  013726        MOV     $[illegible]E,BASE      ; GET BASE CSR ADDRESS
2150  013[illegible]54  113737  001366  013724        MOVB    $VECT1,VECTR    ; GET THE VECTOR
2151  013[illegible]62  113737  001367  013730        MOVB    $VECT1+1,PRIRTY ; GET THE PRIORITY
2152  013[illegible]70  013737  001374  001470        MOV     $[illegible]VM,KMACTV   ; SAVE THE KMC'S SELECTED ACTIVE
2153  013576  013737  001470  001474        MOV     KMACTV,SAVACT   ; SAVE THE ACTIVE REGISTER
2154  013604  012702  001402                MOV     #DDHD,R2        ; GET ADDRESS OF FIRST DEVICE DESCRIPTOR WORD
2155  013610  012703  002100                MOV     #KM.MAP,R3      ; GET POINTER TO DEVICE MAP
2156  013614  005023                3$:     CLR     (R3)+           ; CLEAR DEVICE MAP
2157  013616  022703  002300                CMP     #KM.END,R3      ; IS WHOLE DEV.MAP CLEARED?
2158  013622  003374                        BGT     3$              ; NO, THEN GO ON.
2159  013624  012703  002100                MOV     #KM.MAP,R3      ; RESTORE DEV.MAP POINTER.
2160  013630  013723  013726        1$:     MOV     BASE,(R3)+      ; LOAD CSR ADDRESS
2161  013634  112163  000001                MOVB    (R1)+,1(R3)     ; GET EXTRA INFO, BITS
2162  013640  006213                        ASR     (R3)            ; SET IT IN RIGHT POSITION.
2163  013642  006213                        ASR     (R3)            ; SET IT IN RIGHT POSITION.
2164  013644  053713  013730                BIS     PRIRTY,(R3)     ; GET PRIORITY IN STAT1
2165  013650  006313                        ASL     (R3)            ; SET THEM IN RIGHT POSITION
2166  013652  006313                        ASL     (R3)            ;  "   "   "   "   "
2167  013654  006313                        ASL     (R3)            ;  "   "   "   "     "
2168  013656  006313                        ASL     (R3)            ;  "   "   "   "     "
2169  013660  053723  013724                BIS     VECTR,(R3)+     ; GET THE VECTOR IN STAT1.
2170  013664  012223                        MOV     (R2)+,(R3)+     ; GET THE STAT2 FROM DDWXX
2171  013666  005723                        TST     (R3)+           ; SKIP OVER STAT3
2172  013670  005300                        DEC     R0              ; COUNT BY 1
2173  013672  001407                        BEQ     2$              ; ALL DONE?
2174  013674  062737  000010  013726        ADD     #10,BASE        ; INCREMENT BASE CSR ADDRESS BY 10
2175  013702  062737  000010  013724        ADD     #10,VECTR       ; INCREMENT VECTOR ADDRESS BY 10
2176  013710  000747                        BR      1$              ; SET THE NEXT MAP ENTRY
2177  013712                        2$:
```

```
2178  013712  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
2179  013714  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
2180  013716  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
2181  013720  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
2182  013722  000207                  RTS     PC              ; RETURN
2183  013724  000000          VECTR:  .WORD   0
2184  013726  000000          BASE:   .WORD   0
2185  013730  000000          PRIRTY: .WORD   0
2186
2187  013732                  ROMMAP:
```

```
2188
2189
2190
2191                                    ;*************************** TEST 1 ****************************
2192                                    ;*TEST OF BR RIGHT SHIFT
2193                                    ;*VERIFY THAT A DEST OF BR RSH (011) OF A MICRO-INSTRUCTION
2194                                    ;*SHIFTS THE RESULTING BR DATA RIGHT ONCE.
2195                                    ;:************************************************************
2196
2197                                    ;  TEST 1
2198                                    ;-------------
2199                                ;;**************************************************************
2200  013732  000004                TST1:   SCOPE
2201  013734  012737  000001  001202        MOV     #1,$TSTNM               ; LOAD THE NO. OF THIS TEST
2202  013742  012737  014044  001442        MOV     #TST2,NEXT              ; POINT TO THE START OF NEXT TEST.
2203                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2204  013750  104410                       MSTCLR                    ;MASTER CLEAR KMC11
2205  013752  013701  002066               MOV     KMCSR,R1          ;R1 = KMC BASE ADDRESS
2206  013756  005011                       CLR     (R1)              ;CLEAR SELO
2207  013760  012705  052525               MOV     #52525,R5         ;START WITH 125
2208  013764  010561  000004               MOV     R5,4(R1)          ;PORT4←125
2209  013770  104412                       ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2210  013772  120500                       120500                    ;BR ← PORT4
2211  013774  104412                       ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2212  013776  061620                       061620                    ;BR RSH←BR, SHIFT BR RIGHT
2213  014000  104412                       ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2214  014002  061225                       061225                    ;PORT5←BR
2215  014004  006005                       ROR     R5                ;R5 = "EXPECTED"
2216  014006  116104  000005               MOVB    5(R1),R4          ;R4 = "FOUND"
2217  014012  120504                       CMPB    R5,R4             ;DID BR SHIFT RIGHT ONCE?
2218  014014  001401                       BEQ     1$                ;BR IF YES
2219  014016  104012                       ERROR   12                ;BR RIGHT SHIFT ERROR
2220  014020                        1$:
2221  014020  104412                       ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2222  014022  061620                       061620                    ;BR RSH←BR, SHFT BR RIGHT AGAIN
2223  014024  104412                       ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2224  014026  061225                       061225                    ;PORT5←BR
2225  014030  006005                       ROR     R5                ;R5 = "EXPECTED"
2226  014032  116104  000005               MOVB    5(R1),R4          ;R4 = "FOUND"
2227  014036  120504                       CMPB    R5,R4             ;DID BR SHIFT RIGHT?
2228  014040  001401                       BEQ     2$                ;BR IF YES
2229  014042  104012                       ERROR   12                ;BR RIGHT SHIFT ERROR
2230  014044                        2$:
2231
2232
2233                                    ;*************************** TEST 2 ****************************
2234                                    ;*IOP CRAM WRITE/READ TEST
2235                                    ;*FLOAT A 1 THROUGH EACH CRAM LOCATION
2236                                    ;:************************************************************
2237
2238                                    ;  TEST 2
2239                                    ;-------------
2240                                ;;**************************************************************
2241  014044  000004                TST2:   SCOPE
2242  014046  012737  000002  001202        MOV     #2,$TSTNM               ; LOAD THE NO. OF THIS TEST
2243  014054  012737  014150  001442        MOV     #TST3,NEXT              ; POINT TO THE START OF NEXT TEST.
```

```
2244  014062  012737  014076  001444          MOV     #3$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2245                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2246  014070  005000                          CLR     R0              ;R0 = CRAM ADDRESS
2247  014072  012702  000001          1$:     MOV     #1,R2           ;R2 = WRITE DATA
2248  014076                           2$:
2249  014076  012711  002000          3$:     MOV     #BIT10,(R1)     ;SET ROMO
2250  014102  010061  000004                  MOV     R0,4(R1)        ;WRITE ADDRESS TO SEL4
2251  014106  010261  000006                  MOV     R2,6(R1)        ;LOAD SEL6 WITH WRITE DATA
2252  014112  052711  020000                  BIS     #BIT13,(R1)     ;WRITE SEL6 INTO CRAM
2253  014116  016104  000004                  MOV     4(R1),R4        ;READ CRAM INTO "FOUND"
2254  014122  020204                          CMP     R2,R4           ;IS DATA CORRECT?
2255  014124  001401                          BEQ     4$              ;BR IF OK
2256  014126  104001                          ERROR   1               ;ERROR
2257  014130  104405                  4$:     SCOPE
2258  014132  000241                          CLC                     ;CLEAR CARRY
2259  014134  006102                          ROL     R2              ;SHIFT WRITE DATA
2260  014136  001357                          BNE     2$              ;BR IF NOT DONE THIS ADDRESS
2261  014140  005200                          INC     R0              ;BUMP TO NEXT CRAM ADDRESS
2262  014142  022700  002000                  CMP     #2000,R0        ;DONE YET?
2263  014146  001351                          BNE     1$              ;BR IF NO
2264  014150                           5$:
2265
2266
2267                                   ;************************** TEST 3 **************************
2268                                   ;*IOP CRAM WRITE/READ TEST
2269                                   ;*FLOAT A 0 THROUGH EACH CRAM LOCATION
2270                                   ;:*******************************************************
2271
2272                                   ;  TEST 3
2273                                   ;-------------
2274                                   ;:*******************************************************
2275  014150  000004                  TST3:   SCOPE
2276  014152  012737  000003  001202          MOV     #3,$TSTNM               ; LOAD THE NO. OF THIS TEST
2277  014160  012737  014262  001442          MOV     #TST4,NEXT              ; POINT TO THE START OF NEXT TEST.
2278  014166  012737  014206  001444          MOV     #3$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2279                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2280  014174  104410                          MSTCLR                  ;MASTER CLEAR KMC11
2281  014176  005000                          CLR     R0              ;R0 = CRAM ADDRESS
2282  014200  012702  000001          1$:     MOV     #1,R2           ;R2 = WRITE DATA
2283  014204                           2$:
2284  014204  005102                          COM     R2              ;MAKE IT A FLOATING ZERO
2285  014206  012711  002000          3$:     MOV     #BIT10,(R1)     ;SET ROMO
2286  014212  010061  000004                  MOV     R0,4(R1)        ;WRITE ADDRESS TO SEL4
2287  014216  010261  000006                  MOV     R2,6(R1)        ;LOAD SEL6 WITH WRITE DATA
2288  014222  052711  020000                  BIS     #BIT13,(R1)     ;WRITE SEL6 INTO CRAM
2289  014226  016104  000004                  MOV     4(R1),R4        ;READ CRAM INTO "FOUND"
2290  014232  020204                          CMP     R2,R4           ;IS DATA CORRECT?
2291  014234  001401                          BEQ     4$              ;BR IF OK
2292  014236  104001                          ERROR   1               ;ERROR
2293  014240  104405                  4$:     SCOPE
2294  014242  005102                          COM     R2              ;BACK TO FLOATING ONE
2295  014244  000241                          CLC                     ;CLEAR CARRY
2296  014246  006102                          ROL     R2              ;SHIFT WRITE DATA
2297  014250  001355                          BNE     2$              ;BR IF NOT DONE THIS ADDRESS
2298  014252  005200                          INC     R0              ;BUMP TO NEXT CRAM ADDRESS
2299  014254  022700  002000                  CMP     #2000,R0        ;DONE YET?
```

```
2300  014260  001347                        BNE     1$              ;BR IF NO
2301  014262                        5$:
2302
2303
2304                                        ;************************** TEST 4 **************************
2305                                        ;*IOP CRAM DUAL ADDRESSING TEST
2306                                        ;*WRITE EACH ADDRESS INTO ITSELF,READ EACH
2307                                        ;*ADDRESS TO VERIFY CORRECT ADDRESSING
2308                                        ;;*********************************************************
2309
2310                                        ;  TEST 4
2311                                        ;-------------
2312                                        ;;*********************************************************
2313  014262  000004                TST4:   SCOPE
2314  014264  012737  000004  001202        MOV     #4,$TSTNM               ; LOAD THE NO. OF THIS TEST
2315  014272  012737  014432  001442        MOV     #TST5,NEXT              ; POINT TO THE START OF NEXT TEST.
2316  014300  012737  014312  001444        MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2317                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
2318  014306  104410                        MSTCLR                  ;MASTER CLEAR KMC11
2319  014310  005000                        CLR     R0              ;R0 =CRAM ADDRESS
2320  014312  010002                1$:     MOV     R0,R2           ;SAVE R2 FOR TYPEOUT
2321  014314  012711  002000               MOV     #BIT10,(R1)     ;SET ROMO
2322  014320  010061  000004               MOV     R0,4(R1)        ;WRITE ADDRESS TO SEL4
2323  014324  010061  000006               MOV     R0,6(R1)        ;LOAD SEL6 WITH WRITE DATA
2324  014330  052711  020000               BIS     #BIT13,(R1)     ;WRITE CRAM
2325  014334  005061  000006               CLR     6(R1)           ;CLEAR SEL 6
2326  014340  016104  000006               MOV     6(R1),R4        ;SHOULD READ BACK OWN ADDRESS
2327  014344  020004                        CMP     R0,R4           ;IS DATA CORRECT?
2328  014346  001401                        BEQ     2$              ;BR IF YES
2329  014350  104001                        ERROR   1               ;DATA ERROR
2330  014352  104405                2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
2331  014354  005200                        INC     R0              ;BUMP TO NEXT ADDRESS
2332  014356  022700  002000               CMP     #2000,R0        ;DONE WRITING YET?
2333  014362  001353                        BNE     1$              ;BR IF NO
2334  014364  005000                        CLR     R0              ;RESTART AT ADDRESS 0
2335  014366  012737  014374  001444        MOV     #3$,LOCK        ;NEW SCOP1
2336  014374  010002                3$:     MOV     R0,R2           ;SAVE R2 FOR TYPEOUT
2337  014376  012711  002000               MOV     #BIT10,(R1)     ;SET ROMO
2338  014402  010061  000004               MOV     R0,4(R1)        ;SEL4 = CRAM ADDRESS
2339  014406  016104  000006               MOV     6(R1),R4        ;READ CRAM INTO "FOUND"
2340  014412  020004                        CMP     R0,R4           ;IS DATA CORRECT?
2341  014414  001401                        BEQ     4$              ;BR IF YES
2342  014416  104002                        ERROR   2               ;DUAL ADDRESSING ERROR
2343  014420  104405                4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
2344  014422  005200                        INC     R0              ;BUMP TO NEXT ADDRESS
2345  014424  022700  002000               CMP     #2000,R0        ;DONE WRITING YET?
2346  014430  001361                        BNE     3$              ;BR IF NO
2347  014432                        5$:
2348
2349
2350                                        ;************************** TEST 5 **************************
2351                                        ;*IOP CRAM READ TEST
2352                                        ;*THIS TEST WRITES THE CRAM WITH THE CROM MICRO-CODE MAP
2353                                        ;*THEN READS IT BACK AND COMPARES EACH ADDRESS WITH THE
2354                                        ;*DUPLICATE OF THE CROM MICRO-CODE.
2355                                        ;;*********************************************************
```

```
2356
2357                                        ;  TEST 5
2358                                        ;-------------
2359                                    ;;*****************************************************************
2360  014432  000004                TST5:   SCOPE
2361  014434  012737  000005  001202        MOV     #5,$TSTNM               ; LOAD THE NO. OF THIS TEST
2362  014442  012737  014542  001442        MOV     #TST6,NEXT              ; POINT TO THE START OF NEXT TEST.
2363  014450  012737  014474  001444        MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2364                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2365  014456  104410                       MSTCLR                  ;MASTER CLEAR KMC11
2366  014460  005011                       CLR     (R1)            ;CLEAR RUN
2367  014462  004737  021140               JSR     PC,WROM         ;WRITE CRAM WITH MAP
2368  014466  012700  013732               MOV     #ROMMAP,R0      ;SOFTWARE POINTER TO CROM DUPLICATE
2369  014472  005002                       CLR     R2              ;R2 = CROM ADDRESS
2370  014474  010261  000004          1$:  MOV     R2,4(R1)        ;WRITE CROM ADDRESS TO SEL4
2371  014500  012711  002000               MOV     #BIT10,(R1)     ;SET CROM0
2372  014504  011005                       MOV     (R0),R5         ;PUT "EXPECTED" IN R5
2373  014506  016104  000006               MOV     6(R1),R4        ;PUT "FOUND" IN R4
2374  014512  020504                       CMP     R5,R4           ;COMPARE HARD ROM TO SOFT DUPLICATE
2375  014514  001401                       BEQ     2$              ;BR IF OK
2376  014516  104003                       ERROR   3               ;CRAM READ ERROR!
2377  014520  005011                  2$:  CLR     (R1)            ;CLR  BIT10
2378  014522  005061  000006               CLR     6(R1)           ;CLEAR SEL6
2379  014526  104405                       SCOP1                   ;LOOP TO 1$ IF SW09=1
2380  014530  005202                       INC     R2              ;INC TO NEXT CROM ADDRESS
2381  014532  005720                       TST     (R0)+           ;POP R0 BY 2
2382  014534  022702  002000               CMP     #2000,R2        ;DONE 1K YET?
2383  014540  001355                       BNE     1$              ;BR IF NO
2384  014542                          3$:
2385
2386
2387                                    ;************************** TEST 6 **************************
2388                                    ;*IOP MAIN MEMORY TEST
2389                                    ;*FLOAT A 1 THROUGH ALL MAIN MEMORY LOCATIONS
2390                                    ;:*****************************************************************
2391
2392                                        ;  TEST 6
2393                                        ;-------------
2394                                    ;;*****************************************************************
2395  014542  000004                TST6:   SCOPE
2396  014544  012737  000006  001202        MOV     #6,$TSTNM               ; LOAD THE NO. OF THIS TEST
2397  014552  012737  014720  001442        MOV     #TST7,NEXT              ; POINT TO THE START OF NEXT TEST.
2398  014560  012737  014600  001444        MOV     #65$,LOCK               ; ADDRESS FOR LOCK ON DATA.
2399                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2400  014566  104410                       MSTCLR                  ;MASTER CLEAR KMC11
2401  014570  005037  021012               CLR     FLAG            ;START WITH ADDRESS 0
2402  014574  012700  000001          1$:  MOV     #1,R0           ;START WITH BIT 0
2403  014600  042737  000377  014632  65$:  BIC     #377,66$                ;CLEAR ADDRESS FIELD OF INSTRUCTION
2404  014606  042737  000003  014636        BIC     #3,68$          ;CLEAR ADDRESS FIELD OF INSTRUCTION
2405  014614  153737  021012  014632        BISB    FLAG,66$                ;ADD ADDRESS TO INSTRUCTION
2406  014622  153737  021013  014636        BISB    FLAG+1,68$      ;ADD ADDRESS TO INSTRUCTION
2407  014630  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2408  014632  010000                  66$: 010000                  ;LOAD MAR LO WITH ADDRESS IN FLAG
2409  014634  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2410  014636  004000                  68$: 004000                  ;LOAD MAR HI
2411  014640  010061  000004               MOV     R0,4(R1)        ;WRITE PATTERN IN PORT4
```

```
2412  014644  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2413  014646  122500                          122500                  ;MOVE PORT4 TO MEMORY
2414  014650  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2415  014652  040620                          040620                  ;MOVE MEMORY TO BR
2416  014654  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2417  014656  061225                          61225                   ;MOVE BR TO PORT5
2418  014660  010005                          MOV     R0,R5           ;PUT "EXPECTED" IN R5
2419  014662  116104  000005                  MOVB    5(R1),R4        ;PUT "FOUND" IN R4
2420  014666  120504                          CMPB    R5,R4           ;DATA CORRECT?
2421  014670  001401                          BEQ     67$             ;BR IF YES
2422  014672  104010                          ERROR   10              ;DATA ERROR
2423  014674  104405                  67$:    SCOPE!                  ;SW09=1?
2424  014676  000241                          CLC                     ;CLEAR CARRY
2425  014700  106100                          ROLB    R0              ;SHIFT BIT IN R0
2426  014702  001336                          BNE     65$             ;DONE IF R0=0
2427  014704  005237  021012                  INC     FLAG            ;NEXT ADDRESS
2428  014710  022737  002000  021012          CMP     #2000,FLAG      ;LAST ADDRESS?
2429  014716  001326                          BNE     1$              ;BR IF NO
2430  014720                          2$:
2431
2432
2433                                          ;************************** TEST 7 **************************
2434                                          ;*IOP MAIN MEMORY TEST
2435                                          ;*FLOAT A 0 THROUGH ALL MAIN MEMORY LOCATIONS
2436                                          ;:*************************************************************
2437
2438                                          ;  TEST 7
2439                                          ;-------------
2440                                  ;;**********************************************************************
2441  014720  000004                  TST7:   SCOPE
2442  014722  012737  000007  001202          MOV     #7,$TSTNM               ; LOAD THE NO. OF THIS TEST
2443  014730  012737  015102  001442          MOV     #TST10,NEXT             ; POINT TO THE START OF NEXT TEST.
2444  014736  012737  014760  001444          MOV     #65$,LOCK               ; ADDRESS FOR LOCK ON DATA.
2445                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2446  014744  104410                          MSTCLR                  ;MASTER CLEAR KMC11
2447  014746  005037  021012                  CLR     FLAG            ;START WITH ADDRESS 0
2448  014752  012700  000001          1$:     MOV     #1,R0           ;START WITH BIT 0
2449  014756  005100                  64$:    COM     R0              ;CHANGE TO FLOATING 0
2450  014760  042737  000377  015012  65$:    BIC     #377,66$                ;CLEAR ADDRESS FIELD OF INSTRUCTION
2451  014766  042737  000003  015016          BIC     #3,68$          ;CLEAR ADDRESS FIELD OF INSTRUCTION
2452  014774  153737  021012  015012          BISB    FLAG,66$                ;ADD ADDRESS TO INSTRUCTION
2453  015002  153737  021013  015016          BISB    FLAG+1,68$      ;ADD ADDRESS TO INSTRUCTION
2454  015010  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2455  015012  010000                  66$:    010000                  ;LOAD MAR LO WITH ADDRESS IN FLAG
2456  015014  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2457  015016  004000                  68$:    004000                  ;LOAD MAR HI
2458  015020  010061  000004                  MOV     R0,4(R1)        ;WRITE PATTERN IN PORT4
2459  015024  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2460  015026  122500                          122500                  ;MOVE PORT4 TO MEMORY
2461  015030  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2462  015032  040620                          040620                  ;MOVE MEMORY TO BR
2463  015034  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2464  015036  061225                          61225                   ;MOVE BR TO PORT5
2465  015040  010005                          MOV     R0,R5           ;PUT "EXPECTED" IN R5
2466  015042  116104  000005                  MOVB    5(R1),R4        ;PUT "FOUND" IN R4
2467  015046  120504                          CMPB    R5,R4           ;DATA CORRECT?
```

```
2468  015050  001401                        BEQ     67$             ;BR IF YES
2469  015052  104010                        ERROR   10              ;DATA ERROR
2470  015054  104405              67$:      SCOPE1                  ;SW09=1?
2471  015056  005100                        COM     R0              ;CHANGE TO FLOATING 1
2472  015060  000241                        CLC                     ;CLEAR CARRY
2473  015062  106100                        ROLB    R0              ;SHIFT BIT IN R0
2474  015064  001334                        BNE     64$             ;DONE IF R0=0
2475  015066  005237  021012                INC     FLAG            ;NEXT ADDRESS
2476  015072  022737  002000  021012        CMP     #2000,FLAG      ;LAST ADDRESS?
2477  015100  001324                        BNE     1$              ;BR IF NO
2478  015102                        2$:
2479
2480
2481                                        ;*************************** TEST 10 ****************************
2482                                        ;*IOP MAIN MEMORY DUAL ADDRESSING TEST
2483                                        ;*LOAD EACH MEMORY LOCATION WITH ITS OWN ADDRESS
2484                                        ;*READ BACK EACH LOCATION TO VERIFY CORRECT ADDRESSING
2485                                        ;:************************************************************
2486
2487                                        ;  TEST 10
2488                                        ;-------------
2489                                        ;:************************************************************
2490  015102  000004              TST10:    SCOPE
2491  015104  012737  000010  001202        MOV     #10,$TSTNM              ; LOAD THE NO. OF THIS TEST
2492  015112  012737  015372  001442        MOV     #TST11,NEXT             ; POINT TO THE START OF NEXT TEST.
2493  015120  012737  015134  001444        MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2494                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
2495  015126  104410                        MSTCLR                  ;MASTER CLEAR KMC11
2496  015130  005037  021012                CLR     FLAG            ;START AT ADDRESS 0
2497  015134  013702  021012        1$:     MOV     FLAG,R2         ;PUT DATA IN R2
2498  015140  042737  000377  015172        BIC     #377,2$         ;CLEAR ADDRESS FIELD OF INSTRUCTION
2499  015146  042737  000003  015176        BIC     #3,7$           ;CLEAR ADDRESS FIELD OF INSTRUCTION
2500  015154  153737  021012  015172        BISB    FLAG,2$         ;ADD ADDRESS TO INSTRUCTION
2501  015162  153737  021013  015176        BISB    FLAG+1,7$       ;ADD ADDRESS TO INSTRUCTION
2502  015170  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2503  015172  010000                2$:     010000                  ;LOAD MAR LO
2504  015174  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2505  015176  004000                7$:     004000                  ;LOAD MAR HI
2506  015200  010261  000004              MOV     R2,4(R1)
2507  015204  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2508  015206  122500                        122500                  ;MOVE PORT4 TO MEMORY
2509  015210  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2510  015212  040620                        040620                  ;MOVE MEMORY TO THE BR
2511  015214  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2512  015216  061225                        61225                   ;MOV BR TO PORT5
2513  015220  010205                        MOV     R2,R5           ;PUT "EXPECTED" IN R5
2514  015222  116104  000005              MOVB    5(R1),R4        ;PUT "FOUND" IN R4
2515  015226  120504                        CMPB    R5,R4           ;DATA CORRECT?
2516  015230  001401                        BEQ     3$              ;BR IF YES
2517  015232  104010                        ERROR   10              ;DATA ERROR
2518  015234  104405              3$:       SCOPE1                  ;SW09=1?
2519  015236  005237  021012                INC     FLAG            ;NEXT ADDRESS
2520  015242  022737  002000  021012        CMP     #2000,FLAG      ;LAST ADDRESS
2521  015250  001331                        BNE     1$              ;BR IF NO
2522  015252  012737  015264  001444        MOV     #4$,LOCK        ;NEW SCOPE 1
2523  015260  005037  021012                CLR     FLAG            ;RESTART AT ADDRESS 0
```

```
2524  015264  013702  021012          4$:   MOV     FLAG,R2         ;PUT DATA IN R2
2525  015270  042737  000377  015322        BIC     #377,5$         ;CLEAR ADDRESS FIELD OF INSTRUCTION
2526  015276  042737  000003  015326        BIC     #3,8$           ;CLEAR ADDRESS FIELD OF INSTRUCTION
2527  015304  153737  021012  015322        BISB    FLAG,5$         ;ADD ADDRESS TO INSTRUCTION
2528  015312  153737  021013  015326        BISB    FLAG+1,8$       ;ADD ADDRESS TO INSTRUCTION
2529  015320  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2530  015322  010000                  5$:   010000                  ;LOAD THE MAR LO
2531  015324  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2532  015326  004000                  8$:   004000                  ;LOAD MAR HI
2533  015330  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2534  015332  040620                        040620                  ;MOVE MEMORY TO THE BR
2535  015334  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2536  015336  061225                        61225                   ;MOV BR TO PORT5
2537  015340  010205                        MOV     R2,R5           ;PUT "EXPECTED" IN R5
2538  015342  116104  000005                MOVB    5(R1),R4        ;PUT "FOUND" IN R4
2539  015346  120504                        CMPB    R5,R4           ;DATA CORRECT?
2540  015350  001401                        BEQ     6$              ;BR IF YES
2541  015352  104010                        ERROR   10              ;ADDRESSING ERROR
2542  015354  104405                  6$:   SCOP1                   ;SW09=1?
2543  015356  005237  021012                INC     FLAG            ;NEXT ADDRESS
2544  015362  022737  002000  021012        CMP     #2000,FLAG      ;IS IT THE LAST
2545  015370  001335                        BNE     4$              ;BR IF NO
2546  015372                            9$:
2547
2548
2549                                        ;*************************** TEST 11 ****************************
2550                                        ;*IOP MAR TEST
2551                                        ;*PERFORM DUAL ADDRESSING TEST
2552                                        ;*USING MAR AUTO-INC FEATURE
2553                                        ;;************************************************************************
2554
2555                                        ;  TEST 11
2556                                        ;--------------
2557                                        ;;************************************************************************
2558  015372  000004                  TST11:  SCOPE
2559  015374  012737  000011  001202        MOV     #11,$TSTNM              ; LOAD THE NO. OF THIS TEST
2560  015402  012737  015476  001442        MOV     #TST12,NEXT             ; POINT TO THE START OF NEXT TEST.
2561                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2562  015410  104410                        MSTCLR                  ;MASTER CLEAR KMC11
2563  015412  005002                        CLR     R2              ;START WITH A ZERO
2564  015414  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2565  015416  010000                        010000                  ;LOAD MAR WITH A ZERO
2566  015420  010261  000004          1$:   MOV     R2,4(R1)        ;WRITE DATA TO PORT4
2567  015424  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2568  015426  136500                        136500                  ;MEM<-PORT4, AUTO-INC MAR
2569  015430  005202                        INC     R2              ;INCREMENT DATA
2570  015432  022702  002000                CMP     #2000,R2        ;DONE YET?
2571  015436  001370                        BNE     1$              ;BR IF NO
2572  015440  005002                        CLR     R2              ;RESTART WITH A ZERO
2573  015442  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2574  015444  010000                        010000                  ;LOAD MAR WITH A ZERO
2575  015446                            2$:
2576  015446  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2577  015450  055224                        055224                  ;MOVE MEM TO PORT4
2578  015452  010205                        MOV     R2,R5           ;PUT "EXPECTED" IN R5
2579  015454  016104  000004                MOV     4(R1),R4        ;PUT "FOUND" IN R4
```

```
2580  015460  120504                        CMPB    R5,R4           ;DATA CORRECT?
2581  015462  001401                        BEQ     3$              ;BR IF YES
2582  015464  104011                        ERROR   11              ;MAR ERROR
2583  015466  005202                3$:     INC     R2              ;NEXT ADDRESS
2584  015470  022702  002000                CMP     #2000,R2        ;DONE YET?
2585  015474  001364                        BNE     2$              ;BR IF NO
2586  015476                        4$:
2587
2588
2589                                        ;*************************** TEST 12 ****************************
2590                                        ;*IOP (CRAM) OUT BITS TEST
2591                                        ;*LOAD MAR WITH A 0 INC MAR UNTIL IT OVERFLOWS (2000 TIMES)
2592                                        ;*VERIFY THAT IBUS* 10 BITS IS SET ONLY WHEN MAR BIT 8 IS A ONE
2593                                        ;*AND THAT IBUS* 10 BIT6 IS SET ON MAR OVERFLOW(2000)
2594                                        ;:************************************************************
2595
2596                                        ;  TEST 12
2597                                        ;-------------
2598                                    ;;**********************************************************
2599  015476  000004                TST12:  SCOPE
2600  015500  012737  000012  001202        MOV     #12,$TSTNM              ; LOAD THE NO. OF THIS TEST
2601  015506  012737  015674  001442        MOV     #TST13,NEXT             ; POINT TO THE START OF NEXT TEST.
2602  015514  012737  015532  001444        MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2603                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
2604  015522  104410                        MSTCLR                  ;MASTER CLEAR KMC11
2605  015524  005002                        CLR     R2              ;R2=SAME AS MAR CONTENTS
2606  015526  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2607  015530  010000                        010000                  ;MAR←0
2608  015532                        1$:
2609  015532  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2610  015534  121204                        121204                  ;PORT4=IBUS* 10
2611  015536  005005                        CLR     R5              ;R5="EXPECTED"
2612  015540  032702  000400                BIT     #BIT8,R2        ;IS BIT8 SET IN MAR?
2613  015544  001402                        BEQ     .+6             ;BR IF NO
2614  015546  012705  000040                MOV     #BIT5,R5        ;IF YES THEN SET BITS
2615  015552  016104  000004                MOV     4(R1),R4        ;R4="FOUND"
2616  015556  042704  177637                BIC     #177637,R4      ;CLEAR UNWANTED BITS
2617  015562  020504                        CMP     R5,R4           ;BITS 5&6 SHOULD BE CLEAR
2618  015564  001401                        BEQ     .+4             ;BR IF OK
2619  015566  104007                        ERROR   7               ;ERROR BITS 5&6 NOT CLEAR
2620  015570  104405                        SCOP1                   ;LOOP TO 11$ IF SW09=1
2621  015572  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2622  015574  014000                        014000                  ;INC MAR
2623  015576  005202                        INC     R2              ;BUMP MEM ADDRESS
2624  015600  022702  002000                CMP     #2000,R2        ;OVERFLOWED YET?
2625  015604  001352                        BNE     1$              ;BR IF NO
2626  015606  005037  001444                CLR     LOCK            ;NO MORE SCOP1
2627  015612  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2628  015614  121204                        121204                  ;PART4←IBUS* 10
2629  015616  012705  000100                MOV     #BIT6,R5        ;R5="EXPECTED"
2630  015622  016104  000004                MOV     4(R1),R4        ;R4="FOUND"
2631  015626  042704  177637                BIC     #177637,R4      ;CLEAR UNWANTED BITS
2632  015632  020504                        CMP     R5,R4           ;BIT6 SHOULD BE SET
2633  015634  001401                        BEQ     .+4             ;BR IF OK
2634  015636  104007                        ERROR   7               ;ERROR, BIT6 NOT SET
2635  015640  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
```

```
2636  015642  010000                          010000                  ;MAR←0
2637  015644  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2638  015646  004000                          004000                  ;MAR HI←0
2639  015650  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2640  015652  121204                          121204                  ;PORT4←IBUS# 10
2641  015654  005005                          CLR     R5              ;R5="EXPECTED"
2642  015656  016104  000004                  MOV     4(R1),R4        ;R4="FOUND"
2643  015662  042704  177637                  BIC     #177637,R4      ;CLEAR UNWANTED BITS
2644  015666  020504                          CMP     R5,R4           ;BITS 5&6 SHOULD BE CLEAR
2645  015670  001401                          BEQ     .+4             ;BR IF OK
2646  015672  104007                          ERROR   7               ;ERROR 5&6 NOT BOTH CLEAR
2647  015674                          2$:
2648
2649
2650                                          ;************************** TEST 13 **************************
2651                                          ;*CRAM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
2652                                          ;*PERFORM THE JUMP INSTRUCTION
2653                                          ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
2654                                          ;*IN THE LOCATION IT IS AT, THIS INSTRUCTION LOADS THE
2655                                          ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
2656                                          ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
2657                                          ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
2658                                          ;*THEN PORT4 CONTAINS A 37
2659                                          ;;*********************************************************
2660
2661                                          ;  TEST 13
2662                                          ;--------------
2663                                  ;;*********************************************************
2664  015674  000004                  TST13:  SCOPE
2665  015676  012737  000013  001202          MOV     #13,$TSTNM              ; LOAD THE NO. OF THIS TEST
2666  015704  012737  016060  001442          MOV     #TST14,NEXT             ; POINT TO THE START OF NEXT TEST.
2667  015712  012737  015726  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2668                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2669  015720  104410                          MSTCLR                  ;MASTER CLEAR KMC11
2670  015722  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
2671  015726                          1$:
2672  015726  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
2673  015732  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2674  015734  100400                          100400                  ;START AT ROM PC=0
2675  015736  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2676  015740  114377                          114377!<400*0>          ;JUMP TO ROM PC OF 1777
2677  015742  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2678  015746  000001                          1                       ;EXPECTED DATA
2679  015750  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2680  015752  001401                          BEQ     2$              ;BR IF YES
2681  015754  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2682  015756  104405                  2$:     SCOPI                   ;LOOP TO 1$ IF SW09=1
2683  015760  012737  015766  001444          MOV     #3$,LOCK        ;NEW SCOPI
2684  015766                          3$:
2685  015766  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
2686  015772  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2687  015774  100403                          100403                  ;START AT ROM PC=3
2688  015776  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2689  016000  100000                          100000!<400*0>  ;JUMP TO ROM PC OF 0
2690  016002  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2691  016006  000004                          4                       ;EXPECTED DATA
```

```
2692  016010  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2693  016012  001401                          BEQ     4$              ;BR IF YES
2694  016014  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2695  016016  104405                  4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
2696  016020  012737  016026  001444          MOV     #5$,LOCK        ;NEW SCOP1
2697  016026                          5$:
2698  016026  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
2699  016032  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2700  016034  100406                          100406                  ;START AT ROM PC=6
2701  016036  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2702  016040  104125                          104125!<400*0>          ;JUMP TO ROM PC OF 525
2703  016042  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2704  016046  000007                          7                       ;EXPECTED DATA
2705  016050  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2706  016052  001401                          BEQ     6$              ;BR IF YES
2707  016054  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2708  016056  104405                  6$:     SCOP1                   ;LOOP TO 5$ IF SW59=1
2709
2710
2711                                          ;************************** TEST 14 **************************
2712                                          ;*CRAM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
2713                                          ;*PERFORM THE JUMP INSTRUCTION
2714                                          ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
2715                                          ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
2716                                          ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
2717                                          ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
2718                                          ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
2719                                          ;*THEN PORT4 WILL CONTAIN A 37
2720                                          ;;**********************************************************************
2721
2722                                          ;  TEST 14
2723                                          ;--------------
2724                                  ;;**********************************************************************
2725  016060  000004                  TST14:  SCOPE
2726  016062  012737  000014  001202          MOV     #14,$ISTNM              ; LOAD THE NO. OF THIS TEST.
2727  016070  012737  016230  001442          MOV     #TST15,NEXT             ; POINT TO THE START OF NEXT TEST.
2728  016076  012737  016112  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2729                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2730  016104  104410                          MSTCLR                  ;MASTER CLEAR KMC11
2731  016106  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
2732  016112                          1$:
2733  016112  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2734  016114  100400                          100400                  ;START AT ROM PC=0
2735  016116  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2736  016120  114777                          114377!<400*1>          ;JUMP TO ROM PC OF 1777
2737  016122  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2738  016126  000377                          377                     ;EXPECTED DATA
2739  016130  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2740  016132  001401                          BEQ     2$              ;BR IF YES
2741  016134  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2742  016136  104405                  2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
2743  016140  012737  016146  001444          MOV     #3$,LOCK        ;NEW SCOP1
2744  016146                          3$:
2745  016146  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2746  016150  100403                          100403                  ;START AT ROM PC=3
2747  016152  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
```

```
                                        100000!(400*1)  ;JUMP TO ROM PC OF 0
                                        JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
                                        0                       ;EXPECTED DATA
                                        CMPB    R5,R4           ;IS ROM PC CORRECT?
                                        BEQ     4$              ;BR IF YES
                                        ERROR   5               ;ERROR, CRAM PC IS WRONG
                                        SCOP1                   ;LOOP TO 3$ IF SW09=1
                                4$:     MOV     #5$,LOCK        ;NEW SCOP1
                                5$:
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        100406                  ;START AT ROM PC=6
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        104125!(400*1)          ;JUMP TO ROM PC OF 525
                                        JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
                                        125                     ;EXPECTED DATA
                                        CMPB    R5,R4           ;IS ROM PC CORRECT?
                                        BEQ     6$              ;BR IF YES
                                        ERROR   5               ;ERROR, CRAM PC IS WRONG
                                6$:     SCOP1                   ;LOOP TO 5$ IF SW59=1


                                ;************************** TEST 15 **************************
                                ;*CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
                                ;*SET THE C BIT, PERFORM THE JUMP INSTRUCTION,
                                ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
                                ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
                                ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
                                ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
                                ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
                                ;*THEN PORT4 WILL CONTAIN A 37
                                ;:*********************************************************


                                ;  TEST 15
                                ;-------------

                                ;;*********************************************************
                        TST15:  SCOPE
016230  000004
016232  012737  000015  001202          MOV     #15,$TSTNM              ; LOAD THE NO. OF THIS TEST
016240  012737  016414  001442          MOV     #TST16,NEXT             ; POINT TO THE START OF NEXT TEST.
016246  012737  016262  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
016254  104410                          MSTCLR                  ;MASTER CLEAR KMC11
016256  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
016262                          1$:
016262  004737  021062                  JSR     PC,SETC ;SET THE C BIT'
016266  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
016270  100400                          100400                  ;START AT ROM PC=0
016272  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
016274  115377                          114377!(400*2)          ;JUMP TO ROM PC OF 1777
016276  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
016302  000377                          377                     ;EXPECTED DATA
016304  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
016306  001401                          BEQ     2$              ;BR IF YES
016310  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
016312  104405                  2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
016314  012737  016322  001444          MOV     #3$,LOCK        ;NEW SCOP1
016322                          3$:
```

```
2804  016322  004737  021062                  JSR     PC,SETC ;SET THE C BIT'
2805  016326  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2806  016330  100403                          100403                  ;START AT ROM PC=3
2807  016332  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2808  016334  101000                          100000!<400*2>  ;JUMP TO ROM PC OF 0
2809  016336  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2810  016342  000000                          0                       ;EXPECTED DATA
2811  016344  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2812  016346  001401                          BEQ     4$              ;BR IF YES
2813  016350  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2814  016352  104405                  4$:     SCOPI                   ;LOOP TO 3$ IF SW09=1
2815  016354  012737  016362  001444          MOV     #5$,LOCK        ;NEW SCOPI
2816  016362                          5$:
2817  016366  004737  021062                  JSR     PC,SETC ;SET THE C BIT'
2818  016372  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2819  016374  100406                          100406                  ;START AT ROM PC=6
2820  016376  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2821  016400  105125                          104125!<400*2>          ;JUMP TO ROM PC OF 525
2822  016402  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2823  016406  000125                          125                     ;EXPECTED DATA
2824  016410  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2825  016412  001401                          BEQ     6$              ;BR IF YES
2826  016414  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2827  016416  104405                  6$:     SCOPI                   ;LOOP TO 5$ IF SW59=1
2828
2829
2830                                          ;*************************** TEST 16 ****************************
2831                                          ;*CRAM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
2832                                          ;*SET THE Z BIT, PERFORM THE JUMP INSTRUCTION,
2833                                          ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
2834                                          ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
2835                                          ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
2836                                          ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
2837                                          ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
2838                                          ;*THEN PORT4 WILL CONTAIN A 37
2839                                          ;;*************************************************************
2840
2841                                          ;  TEST 16
2842                                          ;-------------
2843                                  ;;*************************************************************
2844  016420  000004                  TST16:  SCOPE
2845  016422  012737  000016  001202          MOV     #16,STSTNM              ; LOAD THE NO. OF THIS TEST
2846  016430  012737  016600  001442          MOV     #TST17,NEXT             ; POINT TO THE START OF NEXT TEST.
2847  016436  012737  016446  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2848                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2849  016444  104410                          MSTCLR                  ;MASTER CLEAR KMC11
2850  016446  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
2851  016452                          1$:
2852  016452  004737  021100                  JSR     PC,SETZ ;SET THE Z BIT'
2853  016456  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2854  016460  100400                          100400                  ;START AT ROM PC=0
2855  016462  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2856  016464  115777                          114377!<400*3>          ;JUMP TO ROM PC OF 1777
2857  016466  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2858  016472  000377                          377                     ;EXPECTED DATA
2859  016474  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
```

```
2860  016472  001401                          BEQ     2$              ;BR IF YES
2861  016474  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2862  016476  104405                  2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
2863  016500  012737  016506  001444          MOV     #3$,LOCK        ;NEW SCOP1
2864  016506                          3$:
2865  016506  004737  021100                  JSR     PC,SETZ ;SET THE Z BIT'
2866  016512  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2867  016514  100403                          100403                  ;START AT ROM PC=3
2868  016516  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2869  016520  101400                          100000!<400*3>  ;JUMP TO ROM PC OF 0
2870  016522  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2871  016526  000000                          0                       ;EXPECTED DATA
2872  016530  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2873  016532  001401                          BEQ     4$              ;BR IF YES
2874  016534  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2875  016536  104405                  4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
2876  016540  012737  016546  001444          MOV     #5$,LOCK        ;NEW SCOP1
2877  016546                          5$:
2878  016546  004737  021100                  JSR     PC,SETZ ;SET THE Z BIT'
2879  016552  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2880  016554  100406                          100406                  ;START AT ROM PC=6
2881  016556  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2882  016560  105525                          104125!<400*3>          ;JUMP TO ROM PC OF 525
2883  016562  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2884  016566  000125                          125                     ;EXPECTED DATA
2885  016570  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
2886  016572  001401                          BEQ     6$              ;BR IF YES
2887  016574  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
2888  016576  104405                  6$:     SCOP1                   ;LOOP TO 5$ IF SW09=1
2889
2890
2891                                          ;************************** TEST 17 **************************
2892                                          ;*CRAM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
2893                                          ;*SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
2894                                          ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
2895                                          ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
2896                                          ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
2897                                          ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
2898                                          ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
2899                                          ;*THEN PORT4 WILL CONTAIN A 37
2900                                          ;;*************************************************************
2901
2902                                          ;  TEST 17
2903                                          ;-------------
2904                                  ;;*************************************************************
2905  016600  000004                  TST17:  SCOPE
2906  016602  012737  000017  001202          MOV     #17,$TSTNM              ; LOAD THE NO. OF THIS TEST
2907  016610  012737  016764  001442          MOV     #TST20,NEXT             ; POINT TO THE START OF NEXT TEST.
2908  016616  012737  016632  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2909                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
2910  016624  104410                          MSTCLR                  ;MASTER CLEAR KMC11
2911  016626  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
2912  016632                          1$:
2913  016632  004737  021032                  JSR     PC,SETBRO       ;SET THE BRO BIT'
2914  016636  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2915  016640  100400                          100400                  ;START AT ROM PC=0
```

```
2916  016642  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2917  016644  116377                        114377!<400*4>          ;JUMP TO ROM PC OF 1777
2918  016646  004737  021106                JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2919  016652  000377                        377                     ;EXPECTED DATA
2920  016654  120504                        CMPB    R5,R4           ;IS ROM PC CORRECT?
2921  016656  001401                        BEQ     2$              ;BR IF YES
2922  016660  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
2923  016662  104405              2$:       SCOPE                   ;LOOP TO 1$ IF SW09=1
2924  016664  012737  016672  001444        MOV     #3$,LOCK        ;NEW SCOPE
2925  016672                        3$:
2926  016672  004737  021032                JSR     PC,SETBRO       ;SET THE BRO BIT'
2927  016676  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2928  016700  100403                        100403                  ;START AT ROM PC=3
2929  016702  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2930  016704  102000                        100000!<400*4>  ;JUMP TO ROM PC OF 0
2931  016706  004737  021106                JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2932  016712  000000                        0                       ;EXPECTED DATA
2933  016714  120504                        CMPB    R5,R4           ;IS ROM PC CORRECT?
2934  016716  001401                        BEQ     4$              ;BR IF YES
2935  016720  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
2936  016722  104405              4$:       SCOPE                   ;LOOP TO 3$ IF SW09=1
2937  016724  012737  016732  001444        MOV     #5$,LOCK        ;NEW SCOPE
2938  016732                        5$:
2939  016732  004737  021032                JSR     PC,SETBRO       ;SET THE BRO BIT'
2940  016736  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2941  016740  100406                        100406                  ;START AT ROM PC=6
2942  016742  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2943  016744  106125                        104125!<400*4>          ;JUMP TO ROM PC OF 525
2944  016746  004737  021106                JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
2945  016752  000125                        125                     ;EXPECTED DATA
2946  016754  120504                        CMPB    R5,R4           ;IS ROM PC CORRECT?
2947  016756  001401                        BEQ     6$              ;BR IF YES
2948  016760  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
2949  016762  104405              6$:       SCOPE                   ;LOOP TO 5$ IF SW09=1
2950
2951
2952                                        ;*************************** TEST 20 ****************************
2953                                        ;*CRAM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
2954                                        ;*SET THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
2955                                        ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
2956                                        ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
2957                                        ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
2958                                        ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
2959                                        ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
2960                                        ;*THEN PORT4 WILL CONTAIN A 37
2961                                        ;;*****************************************************************
2962
2963                                        ;  TEST 20
2964                                        ;--------------
2965                                        ;;*****************************************************************
2966  016764  000004              TST20:    SCOPE
2967  016766  012737  000020  001202        MOV     #20,$TSTNM              ; LOAD THE NO. OF THIS TEST
2968  016774  012737  017150  001442        MOV     #TST21,NEXT             ; POINT TO THE START OF NEXT TEST.
2969  017002  012737  017016  001444        MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
2970                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
2971  017010  104410                        MSTCLR                  ;MASTER CLEAR KMC11
```

```
                                        JSR     PC,MEMSET       ;SET MEM AND RAM
                                1$:
                                        JSR     PC,SETBR1       ;SET THE BR1 BIT'
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        100400                  ;START AT ROM PC=0
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        114377!(400*5)          ;JUMP TO ROM PC OF 1777
                                        JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
                                        377                     ;EXPECTED DATA
                                        CMPB    R5,R4           ;IS ROM PC CORRECT?
                                        BEQ     2$              ;BR IF YES
                                        ERROR   5               ;ERROR, CRAM PC IS WRONG
                                2$:     SCOPE                   ;LOOP TO 1$ IF SW09=1
                                        MOV     #3$,LOCK        ;NEW SCOPE
                                3$:
                                        JSR     PC,SETBR1       ;SET THE BR1 BIT'
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        100403                  ;START AT ROM PC=3
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        100000!(400*5)  ;JUMP TO ROM PC OF 0
                                        JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
                                        0                       ;EXPECTED DATA
                                        CMPB    R5,R4           ;IS ROM PC CORRECT?
                                        BEQ     4$              ;BR IF YES
                                        ERROR   5               ;ERROR, CRAM PC IS WRONG
                                4$:     SCOPE                   ;LOOP TO 3$ IF SW09=1
                                        MOV     #5$,LOCK        ;NEW SCOPE
                                5$:
                                        JSR     PC,SETBR1       ;SET THE BR1 BIT'
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        100406                  ;START AT ROM PC=6
                                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                                        104125!(400*5)          ;JUMP TO ROM PC OF 525
                                        JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
                                        125                     ;EXPECTED DATA
                                        CMPB    R5,R4           ;IS ROM PC CORRECT?
                                        BEQ     6$              ;BR IF YES
                                        ERROR   5               ;ERROR, CRAM PC IS WRONG
                                6$:     SCOPE                   ;LOOP TO 5$ IF SW09=1


                                ;*************************** TEST 21 ************************
                                ;*CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR      TION.
                                ;*SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTIO
                                ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE IN
                                ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
                                ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
                                ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
                                ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
                                ;*THEN PORT4 WILL CONTAIN A 37
                                ;:*********************************************************


                                ;  TEST 21
                                ;-------------
                                ;;*********************************************************
017150  000004                  TST21:  SCOPE
```

```
3028  017152  012737  000021  001202          MOV     #21,$TSTNM              ; LOAD THE NO. OF THIS TEST
3029  017160  012737  017334  001442          MOV     #TST22,NEXT             ; POINT TO THE START OF NEXT TEST.
3030  017166  012737  017202  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
3031                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
3032  017174  104410                          MSTCLR                  ;MASTER CLEAR KMC11
3033  017176  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
3034  017202                          1$:
3035  017202  004737  021046                  JSR     PC,SETBR4       ;SET THE BR4 BIT'
3036  017206  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3037  017210  100400                          100400                  ;START AT ROM PC=0
3038  017212  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3039  017214  117377                          117377!<400*6>          ;JUMP TO ROM PC OF 1777
3040  017216  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3041  017222  000377                          377                     ;EXPECTED DATA
3042  017224  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
3043  017226  001401                          BEQ     2$              ;BR IF YES
3044  017230  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
3045  017232  104405                  2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
3046  017234  012737  017242  001444          MOV     #3$,LOCK        ;NEW SCOP1
3047  017242                          3$:
3048  017242  004737  021046                  JSR     PC,SETBR4       ;SET THE BR4 BIT'
3049  017246  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3050  017250  100403                          100403                  ;START AT ROM PC=3
3051  017252  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3052  017254  103000                          100000!<400*6>  ;JUMP TO ROM PC OF 0
3053  017256  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3054  017262  000000                          0                       ;EXPECTED DATA
3055  017264  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
3056  017266  001401                          BEQ     4$              ;BR IF YES
3057  017270  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
3058  017272  104405                  4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
3059  017274  012737  017302  001444          MOV     #5$,LOCK        ;NEW SCOP1
3060  017302                          5$:
3061  017302  004737  021046                  JSR     PC,SETBR4       ;SET THE BR4 BIT'
3062  017306  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3063  017310  100406                          100406                  ;START AT ROM PC=6
3064  017312  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3065  017314  107125                          107125!<400*6>          ;JUMP TO ROM PC OF 525
3066  017316  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3067  017322  000125                          125                     ;EXPECTED DATA
3068  017324  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
3069  017326  001401                          BEQ     6$              ;BR IF YES
3070  017330  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
3071  017332  104405                  6$:     SCOP1                   ;LOOP TO 5$ IF SW59=1
3072
3073
3074                                          ;************************** TEST 22 **************************
3075                                          ;*CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
3076                                          ;*SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
3077                                          ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSRUCTION
3078                                          ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3079                                          ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3080                                          ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
3081                                          ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
3082                                          ;*THEN PORT4 WILL CONTAIN A 37
3083                                          ;;**********************************************************
```

```
3084
3085                                        ;  TEST 22
3086                                        ;-------------
3087                                        ;;*************************************************************
3088  017334  000004                        TST22:  SCOPE
3089  017336  012737  000022  001202                MOV     #22,$TSTNM              ; LOAD THE NO. OF THIS TEST
3090  017344  012737  017520  001442                MOV     #TST23,NEXT             ; POINT TO THE START OF NEXT TEST.
3091  017352  012737  017366  001444                MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
3092                                                                        ;R1 CONTAINS BASE KMC11 ADDRESS
3093  017360  104410                               MSTCLR                  ;MASTER CLEAR KMC11
3094  017362  004737  021202                       JSR     PC,MEMSET       ;SET MEM AND RAM
3095  017366                              1$:
3096  017366  004737  021054                       JSR     PC,SETBR7       ;SET THE BR7 BIT'
3097  017372  104412                               ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3098  017374  100400                               100400                  ;START AT ROM PC=0
3099  017376  104412                               ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3100  017400  117777                               114377!(400*7)          ;JUMP TO ROM PC OF 1777
3101  017402  004737  021106                       JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3102  017406  000377                               377                     ;EXPECTED DATA
3103  017410  120504                               CMPB    R5,R4           ;IS ROM PC CORRECT?
3104  017412  001401                               BEQ     2$              ;BR IF YES
3105  017414  104005                               ERROR   5               ;ERROR, CRAM PC IS WRONG
3106  017416  104405                      2$:      SCOP1                   ;LOOP TO 1$ IF SW09=1
3107  017420  012737  017426  001444                MOV     #3$,LOCK        ;NEW SCOP1
3108  017426                              3$:
3109  017426  004737  021054                       JSR     PC,SETBR7       ;SET THE BR7 BIT'
3110  017432  104412                               ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3111  017434  100403                               100403                  ;START AT ROM PC=3
3112  017436  104412                               ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3113  017440  103400                               100000!(400*7)  ;JUMP TO ROM PC OF 0
3114  017442  004737  021106                       JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3115  017446  000000                               0                       ;EXPECTED DATA
3116  017450  120504                               CMPB    R5,R4           ;IS ROM PC CORRECT?
3117  017452  001401                               BEQ     4$              ;BR IF YES
3118  017454  104005                               ERROR   5               ;ERROR, CRAM PC IS WRONG
3119  017456  104405                      4$:      SCOP1                   ;LOOP TO 3$ IF SW09=1
3120  017460  012737  017466  001444                MOV     #5$,LOCK        ;NEW SCOP1
3121  017466                              5$:
3122  017466  004737  021054                       JSR     PC,SETBR7       ;SET THE BR7 BIT'
3123  017472  104412                               ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3124  017474  100406                               100406                  ;START AT ROM PC=6
3125  017476  104412                               ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3126  017500  107525                               104125!(400*7)          ;JUMP TO ROM PC OF 525
3127  017502  004737  021106                       JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3128  017506  000125                               125                     ;EXPECTED DATA
3129  017510  120504                               CMPB    R5,R4           ;IS ROM PC CORRECT?
3130  017512  001401                               BEQ     6$              ;BR IF YES
3131  017514  104005                               ERROR   5               ;ERROR, CRAM PC IS WRONG
3132  017516  104405                      6$:      SCOP1                   ;LOOP TO 5$ IF SW09=1
3133
3134
3135                                        ;*************************** TEST 23 ***************************
3136                                        ;*CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
3137                                        ;*CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION,
3138                                        ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3139                                        ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
```

```
3140                                        ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3141                                        ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3142                                        ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
3143                                        ;*THEN PORT4 CONTAINS A 37
3144                                        ;;*****************************************************************
3145
3146                                        ;  TEST 23
3147                                        ;-------------
3148                                    ;;*****************************************************************
3149  017520  000004                TST23:  SCOPE
3150  017522  012737  000023  001202        MOV     #23,$TSTNM              ; LOAD THE NO. OF THIS TEST
3151  017530  012737  017704  001442        MOV     #TST24,NEXT             ; POINT TO THE START OF NEXT TEST.
3152  017536  012737  017552  001444        MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
3153                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
3154  017544  104410                       MSTCLR                  ;MASTER CLEAR KMC11
3155  017546  004737  021202               JSR     PC,MEMSET       ;SET MEM AND RAM
3156  017552                        1$:
3157  017552  004737  021014               JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3158  017556  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3159  017560  100400                       100400                  ;START AT ROM PC=0
3160  017562  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3161  017564  115377                       114377!(400*2)          ;JUMP TO ROM PC OF 1777
3162  017566  004737  021106               JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3163  017572  000001                       1                       ;EXPECTED DATA
3164  017574  120504                       CMPB    R5,R4           ;IS ROM PC CORRECT?
3165  017576  001401                       BEQ     2$              ;BR IF YES
3166  017600  104005                       ERROR   5               ;ERROR, CRAM PC IS WRONG
3167  017602  104405                2$:    SCOP1                   ;LOOP TO 1$ IF SW09=1
3168  017604  012737  017612  001444       MOV     #3$,LOCK        ;NEW SCOP1
3169  017612                        3$:
3170  017612  004737  021014               JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3171  017616  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3172  017620  100403                       100403                  ;START AT ROM PC=3
3173  017622  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3174  017624  101000                       100000!(400*2)  ;JUMP TO ROM PC OF 0
3175  017626  004737  021106               JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3176  017632  000004                       4                       ;EXPECTED DATA
3177  017634  120504                       CMPB    R5,R4           ;IS ROM PC CORRECT?
3178  017636  001401                       BEQ     4$              ;BR IF YES
3179  017640  104005                       ERROR   5               ;ERROR, CRAM PC IS WRONG
3180  017642  104405                4$:    SCOP1                   ;LOOP TO 3$ IF SW09=1
3181  017644  012737  017652  001444       MOV     #5$,LOCK        ;NEW SCOP1
3182  017652                        5$:
3183  017652  004737  021014               JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3184  017656  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3185  017660  100406                       100406                  ;START AT ROM PC=6
3186  017662  104412                       ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3187  017664  105125                       104125!(400*2)          ;JUMP TO ROM PC OF 525
3188  017666  004737  021106               JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3189  017672  000007                       7                       ;EXPECTED DATA
3190  017674  120504                       CMPB    R5,R4           ;IS ROM PC CORRECT?
3191  017676  001401                       BEQ     6$              ;BR IF YES
3192  017700  104005                       ERROR   5               ;ERROR, CRAM PC IS WRONG
3193  017702  104405                6$:    SCOP1                   ;LOOP TO 5$ IF SW59=1
3194
3195
```

```
                                        ;*************************** TEST 24 ****************************
                                        ;*CRAM TEST OF JUMP(J) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
                                        ;*CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
                                        ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
                                        ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
                                        ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
                                        ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
                                        ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
                                        ;*THEN PORT4 CONTAINS A 37
                                        ;:************************************************************

                                        ;  TEST 24
                                        ;--------------
                                        ;:************************************************************
017704  000004                  TST24:  SCOPE
017706  012737  000024  001202          MOV     #24,$ISTNM              ; LOAD THE NO. OF THIS TEST
017714  012737  020070  001442          MOV     #TST25,NEXT             ; POINT TO THE START OF NEXT TEST.
017722  012737  017736  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
017730  104410                          MSTCLR                  ;MASTER CLEAR KMC11
017732  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
017736                          1$:
017736  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
017742  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
017744  100400                          100400                  ;START AT ROM PC=0
017746  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
017750  115777                          114377!<400*3>          ;JUMP TO ROM PC OF 1777
017752  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
017756  000001                          1                       ;EXPECTED DATA
017760  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
017762  001401                          BEQ     2$              ;BR IF YES
017764  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
017766  104405                  2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
017770  012737  017776  001444          MOV     #3$,LOCK        ;NEW SCOP1
017776                          3$:
017776  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
020002  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
020004  100403                          100403                  ;START AT ROM PC=3
020006  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
020010  101400                          100000!<400*3>  ;JUMP TO ROM PC OF 0
020012  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
020016  000004                          4                       ;EXPECTED DATA
020020  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
020022  001401                          BEQ     4$              ;BR IF YES
020024  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
020026  104405                  4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
020030  012737  020036  001444          MOV     #5$,LOCK        ;NEW SCOP1
020036                          5$:
020036  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
020042  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
020044  100406                          100406                  ;START AT ROM PC=6
020046  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
020050  105525                          104125!<400*3>          ;JUMP TO ROM PC OF 525
020052  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
020056  000007                          7                       ;EXPECTED DATA
020060  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
```

```
3252  020062  001401                        BEQ     6$              ;BR IF YES
3253  020064  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
3254  020066  104405              6$:       SCOP1                   ;LOOP TO 5$ IF SW5=1
3255
3256
3257                                        ;*************************** TEST 25 ***************************
3258                                        ;*CRAM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
3259                                        ;*CLEAR THE BRO BIT, PERFORM THE JUMP INSTRUCTION,
3260                                        ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3261                                        ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3262                                        ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3263                                        ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3264                                        ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
3265                                        ;*THEN PORT4 CONTAINS A 37
3266                                        ;;*************************************************************
3267
3268                                        ;  TEST 25
3269                                        ;--------------
3270                                    ;;*************************************************************
3271  020070  000004              TST25:  SCOPE
3272  020072  012737  000025  001202        MOV     #25,$TSTNM              ; LOAD THE NO. OF THIS TEST
3273  020100  012737  020254  001442        MOV     #TST26,NEXT             ; POINT TO THE START OF NEXT TEST.
3274  020106  012737  020122  001444        MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
3275                                                                ;R1 CONTAINS BASE KMC11 ADDRESS
3276  020114  104410                        MSTCLR                  ;MASTER CLEAR KMC11
3277  020116  004737  021202                JSR     PC,MEMSET       ;SET MEM AND RAM
3278  020122                        1$:
3279  020122  004737  021014                JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3280  020126  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3281  020130  100400                        100400                  ;START AT ROM PC=0
3282  020132  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3283  020134  116377                        114377!<400*4>          ;JUMP TO ROM PC OF 1777
3284  020136  004737  021106                JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3285  020142  000001                        1                       ;EXPECTED DATA
3286  020144  120504                        CMPB    R5,R4           ;IS ROM PC CORRECT?
3287  020146  001401                        BEQ     2$              ;BR IF YES
3288  020150  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
3289  020152  104405              2$:       SCOP1                   ;LOOP TO 1$ IF SW9=1
3290  020154  012737  020162  001444        MOV     #3$,LOCK        ;NEW SCOP1
3291  020162                        3$:
3292  020162  004737  021014                JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3293  020166  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3294  020170  100403                        100403                  ;START AT ROM PC=3
3295  020172  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3296  020174  102000                        100000!<400*4>  ;JUMP TO ROM PC OF 0
3297  020176  004737  021106                JSR     PC,RAMDAT       ;P4=CRAM PC (LSB 8 BITS)
3298  020202  000004                        4                       ;EXPECTED DATA
3299  020204  120504                        CMPB    R5,R4           ;IS ROM PC CORRECT?
3300  020206  001401                        BEQ     4$              ;BR IF YES
3301  020210  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
3302  020212  104405              4$:       SCOP1                   ;LOOP TO 3$ IF SW9=1
3303  020214  012737  020222  001444        MOV     #5$,LOCK        ;NEW SCOP1
3304  020222                        5$:
3305  020222  004737  021014                JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3306  020226  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3307  020230  100406                        100406                  ;START AT ROM PC=6
```

```
3308  020232  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3309  020234  106125                  104125!<400*4>          ;JUMP TO ROM PC OF 525
3310  020236  004737  021106          JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3311  020242  000007                  7                       ;EXPECTED DATA
3312  020244  120504                  CMPB    R5,R4           ;IS ROM PC CORRECT?
3313  020246  001401                  BEQ     6$              ;BR IF YES
3314  020250  104005                  ERROR   5               ;ERROR, CRAM PC IS WRONG
3315  020252  104405          6$:     SCOP1                   ;LOOP TO 5$ IF SW59=1
3316
3317
3318                                  ;************************** TEST 26 **************************
3319                                  ;*CRAM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
3320                                  ;*CLEAR THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
3321                                  ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3322                                  ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3323                                  ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3324                                  ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3325                                  ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
3326                                  ;*THEN PORT4 CONTAINS A 37
3327                                  ;;**********************************************************
3328
3329                                  ;  TEST 26
3330                                  ;--------------
3331                                  ;;**********************************************************
3332  020254  000004          ;TST26: SCOPE
3333  020256  012737  000026  001202  MOV     #26.,$ISTNM             ; LOAD THE NO. OF THIS TEST
3334  020264  012737  020440  001442  MOV     #TST27,NEXT             ; POINT TO THE START OF NEXT TEST.
3335  020272  012737  020306  001444  MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
3336                                                          ;R1 CONTAINS BASE KMC11 ADDRESS
3337  020300  104410                  MSTCLR                  ;MASTER CLEAR KMC11
3338  020302  004737  021202          JSR     PC,MEMSET       ;SET MEM AND RAM
3339  020306                  1$:
3340  020306  004737  021014          JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3341  020312  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3342  020314  100400                  100400                  ;START AT ROM PC=0
3343  020316  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3344  020320  116777                  114377!<400*5>          ;JUMP TO ROM PC OF 1777
3345  020322  004737  021106          JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3346  020326  000001                  1                       ;EXPECTED DATA
3347  020330  120504                  CMPB    R5,R4           ;IS ROM PC CORRECT?
3348  020332  001401                  BEQ     2$              ;BR IF YES
3349  020334  104005                  ERROR   5               ;ERROR, CRAM PC IS WRONG
3350  020336  104405          2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
3351  020340  012737  020346  001444  MOV     #3$,LOCK        ;NEW SCOP1
3352  020346                  3$:
3353  020346  004737  021014          JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3354  020352  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3355  020354  100403                  100403                  ;START AT ROM PC=3
3356  020356  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3357  020360  102400                  100000!<400*5>  ;JUMP TO ROM PC OF 0
3358  020362  004737  021106          JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3359  020366  000004                  4                       ;EXPECTED DATA
3360  020370  120504                  CMPB    R5,R4           ;IS ROM PC CORRECT?
3361  020372  001401                  BEQ     4$              ;BR IF YES
3362  020374  104005                  ERROR   5               ;ERROR, CRAM PC IS WRONG
3363  020376  104405          4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
```

```
3364  020400  012737  020406  001444          5$:     MOV     #5$,LOCK        ;NEW SCOP1
3365  020406
3366  020406  004737  021014                          JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3367  020412  104412                                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3368  020414  100406                                  100406                  ;START AT ROM PC=6
3369  020416  104412                                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3370  020420  106525                                  106125!<400*5>          ;JUMP TO ROM PC OF 525
3371  020422  004737  021106                          JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3372  020426  000007                                  7                       ;EXPECTED DATA
3373  020430  120504                                  CMPB    R5,R4           ;IS ROM PC CORRECT?
3374  020432  001401                                  BEQ     6$              ;BR IF YES
3375  020434  104005                                  ERROR   5               ;ERROR, CRAM PC IS WRONG
3376  020436  104405                          6$:     SCOP1                   ;LOOP TO 5$ IF SW9=1
3377
3378
3379                                                  ;************************** TEST 27 **************************
3380                                                  ;*CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
3381                                                  ;*CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
3382                                                  ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3383                                                  ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3384                                                  ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3385                                                  ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3386                                                  ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
3387                                                  ;*THEN PORT4 CONTAINS A 37
3388                                                  ;;**********************************************************
3389
3390                                                  ;  TEST 27
3391                                                  ;-------------
3392                                                  ;;**********************************************************
3393  020440  000004                          TST27:  SCOPE
3394  020442  012737  000027  001202                  MOV     #27,$TSTNM              ; LOAD THE NO. OF THIS TEST
3395  020450  012737  020624  001442                  MOV     #TST30,NEXT             ; POINT TO THE START OF NEXT TEST.
3396  020456  012737  020472  001444                  MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
3397                                                                          ;R1 CONTAINS BASE KMC11 ADDRESS
3398  020464  104410                                  MSTCLR                  ;MASTER CLEAR KMC11
3399  020466  004737  021202                          JSR     PC,MEMSET       ;SET MEM AND RAM
3400  020472                                  1$:
3401  020472  004737  021014                          JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3402  020476  104412                                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3403  020500  100400                                  100400                  ;START AT ROM PC=0
3404  020502  104412                                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3405  020504  117377                                  114377!<400*6>          ;JUMP TO ROM PC OF 1777
3406  020506  004737  021106                          JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3407  020512  000001                                  1                       ;EXPECTED DATA
3408  020514  120504                                  CMPB    R5,R4           ;IS ROM PC CORRECT?
3409  020516  001401                                  BEQ     2$              ;BR IF YES
3410  020520  104005                                  ERROR   5               ;ERROR, CRAM PC IS WRONG
3411  020522  104405                          2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
3412  020524  012737  020532  001444                  MOV     #3$,LOCK        ;NEW SCOP1
3413  020532                                  3$:
3414  020532  004737  021014                          JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3415  020536  104412                                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3416  020540  100403                                  100403                  ;START AT ROM PC=3
3417  020542  104412                                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3418  020544  103000                                  100000!<400*6>  ;JUMP TO ROM PC OF 0
3419  020546  004737  021106                          JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
```

```
3420  020552  000004                          4                       ;EXPECTED DATA
3421  020554  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
3422  020556  001401                          BEQ     4$              ;BR IF YES
3423  020560  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
3424  020562  104405                  4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
3425  020564  012737  020572  001444          MOV     #5$,LOCK        ;NEW SCOP1
3426  020572                          5$:
3427  020572  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3428  020576  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3429  020600  100406                          100406                  ;START AT ROM PC=6
3430  020602  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3431  020604  107125                          104125!(400*6)          ;JUMP TO ROM PC OF 525
3432  020606  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3433  020612  000007                          7                       ;EXPECTED DATA
3434  020614  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
3435  020616  001401                          BEQ     6$              ;BR IF YES
3436  020620  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
3437  020622  104405                  6$:     SCOP1                   ;LOOP TO 5$ IF SW09=1
3438
3439
3440                                          ;*************************** TEST 30 ****************************
3441                                          ;*CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
3442                                          ;*CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
3443                                          ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3444                                          ;*IN THE LOCATION IT IS AT, THIS INSTRUCTION LOADS THE
3445                                          ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3446                                          ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3447                                          ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
3448                                          ;*THEN PORT4 CONTAINS A 37
3449                                          ;;*************************************************************
3450
3451                                          ;  TEST 30
3452                                          ;-------------
3453                                          ;;*************************************************************
3454  020624  000004                  TST30:  SCOPE
3455  020626  012737  000030  001202          MOV     #30,$TSTNM              ; LOAD THE NO. OF THIS TEST
3456  020634  012737  003662  001442          MOV     #$EOP,NEXT              ; POINT TO THE END OF PASS HANDLER.
3457  020642  012737  020656  001444          MOV     #1$,LOCK                ; ADDRESS FOR LOCK ON DATA.
3458                                                                  ;R1 CONTAINS BASE KMC11 ADDRESS
3459  020650  104410                          MSTCLR                  ;MASTER CLEAR KMC11
3460  020652  004737  021202                  JSR     PC,MEMSET       ;SET MEM AND RAM
3461  020656                          1$:
3462  020656  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3463  020662  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3464  020664  100400                          100400                  ;START AT ROM PC=0
3465  020666  104412                          ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3466  020670  117777                          114377!(400*7)          ;JUMP TO ROM PC OF 1777
3467  020672  004737  021106                  JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3468  020676  000001                          1                       ;EXPECTED DATA
3469  020700  120504                          CMPB    R5,R4           ;IS ROM PC CORRECT?
3470  020702  001401                          BEQ     2$              ;BR IF YES
3471  020704  104005                          ERROR   5               ;ERROR, CRAM PC IS WRONG
3472  020706  104405                  2$:     SCOP1                   ;LOOP TO 1$ IF SW09=1
3473  020710  012737  020716  001444          MOV     #3$,LOCK        ;NEW SCOP1
3474  020716                          3$:
3475  020716  004737  021014                  JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
```

```
3476  020722  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3477  020724  100403                        100403                  ;START AT ROM PC=3
3478  020726  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3479  020730  103400                        100000!(400*7)  ;JUMP TO ROM PC OF 0
3480  020732  004737  021106                JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3481  020736  000004                        4                       ;EXPECTED DATA
3482  020740  120504                        CMPB    R5,R4           ;IS ROM PC CORRECT?
3483  020742  001401                        BEQ     4$              ;BR IF YES
3484  020744  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
3485  020746  104405                4$:     SCOP1                   ;LOOP TO 3$ IF SW09=1
3486  020750  012737  020756  001444        MOV     #5$,LOCK        ;NEW SCOP1
3487  020756                        5$:
3488  020756  004737  021014                JSR     PC,CLRALL       ;CLEAR ALL  CONDITIONS
3489  020762  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3490  020764  100406                        100406                  ;START AT ROM PC=6
3491  020766  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3492  020770  107525                        104125!(400*7)          ;JUMP TO ROM PC OF 525
3493  020772  004737  021106                JSR     PC,RAMDAT       ;R4=CRAM PC (LSB 8 BITS)
3494  020776  000007                        7                       ;EXPECTED DATA
3495  021000  120504                        CMPB    R5,R4           ;IS ROM PC CORRECT?
3496  021002  001401                        BEQ     6$              ;BR IF YES
3497  021004  104005                        ERROR   5               ;ERROR, CRAM PC IS WRONG
3498  021006  104405                6$:     SCOP1                   ;LOOP TO 5$ IF SW59=1
3499  021010  104420                        ADVANCE                 ; ADVANCE TO NEXT TEST
3500
3501
3502
3503                                        ;BUFFER AREA
3504                                        ;-----------
3505
3506  021012  000000                FLAG:   0
3507
3508
3509                                        ;SUBROUTINES
3510                                        ;-----------
3511
3512  021014                        CLRALL:
3513                                        ;THIS SUBROUTINE CLEARS THE C&Z BITS AND THE BR
3514
3515  021014  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3516  021016  000400                        000400                  ;BR←0
3517  021020  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3518  021022  063220                        063220                  ;SP(0)←BR
3519  021024  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3520  021026  060400                        060400                  ;BR←SP(0)+BR
3521  021030  000207                        RTS     PC
3522
3523
3524  021032                        SETBR0:
3525                                        ;THIS SUBROUTINE SETS BR0 BIT
3526
3527  021032  104412                        ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3528  021034  000401                        000401                  ;BR←001
3529  021036  000207                        RTS     PC
3530
3531
```

```
3532  021040                          SETBR1:
3533                                          ;THIS SUBROUTINE SETS BR1 BIT
3534
3535  021040  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3536  021042  000402                  000402                  ;BR←002
3537  021044  000207                  RTS     PC
3538
3539
3540  021046                          SETBR4:
3541                                          ;THIS SUBROUTINE SETS BR4 BIT
3542
3543  021046  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3544  021050  000420                  000420                  ;BR←020
3545  021052  000207                  RTS     PC
3546
3547
3548  021054                          SETBR7:
3549                                          ;THIS SUBROUTINE SETS BR7 BIT
3550
3551  021054  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3552  021056  000600                  000600                  ;BR←200
3553  021060  000207                  RTS     PC
3554
3555
3556  021062                          SETC:
3557                                          ;THIS SUBROUTINE SETS THE C BIT
3558
3559  021062  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3560  021064  000777                  000777                  ;BR←377
3561  021066  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3562  021070  063220                  063220                  ;SP(0)←BR
3563  021072  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3564  021074  060400                  060400                  ;BR←SP(0)+BR
3565  021076  000207                  RTS     PC
3566
3567
3568  021100                          SETZ:
3569                                          ;THIS SUBROUTINE SETS THE Z BIT
3570
3571  021100  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3572  021102  000777                  000777                  ;BR←377
3573  021104  000207                  RTS     PC
3574
3575
3576  021106                          RAMDAT:
3577                                          ;THIS SUBROUTINE LOADS R4 WITH THE LOWEST
3578                                          ;8 BITS OF THE CRAM PC.
3579
3580  021106  017605  000000          MOV     @(SP),R5        ;GOOD DATA
3581  021112  062716  000002          ADD     #2,(SP)         ;ADJUST STACK
3582  021116  005011                  CLR     (R1)            ;CLEAR BIT10
3583  021120  052711  000400          BIS     #BIT8,(R1)      ;CLOCK INSTRUCTION IN CRAM THAT WAS
3584                                                          ;JUMPED TO, IT LOADS BR WITH ROM PC
3585  021124  005011                  CLR     (R1)            ;CLR BIT8
3586  021126  104412                  ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3587  021130  061225                  061225                  ;MOV BR TO PORT 5
```

```
3588  021132  116104  000005          MOVB    5(R1),R4        ;PUT "FOUND" IN R4
3589  021136  000207                  RTS     PC              ;RETURN
3590
3591  021140                  WROM:
3592                                  ;THIS SUBROUTINE WRITES THE ROMMAP INTO THE CRAM
3593
3594                          ;       BIT     #BIT15,STAT1    ;BE SURE KMC HAS CRAM
3595                          ;       BEQ     2$              ;SKIP IF NO CRAM
3596  021140  005000                  CLR     R0              ;R0=CRAM ADDRESS
3597  021142  012702  013732          MOV     #ROMMAP,R2      ;R2 POINTS TO ROMMAP
3598  021146  012711  002000  1$:     MOV     #BIT10,(R1)     ;SET ROM0
3599  021152  010061  000004          MOV     R0,4(R1)        ;LOAD CRAM ADDRESS
3600  021156  012261  000006          MOV     (R2)+,6(R1)     ;LOAD WORD TO BE WRITTEN
3601  021162  052711  020000          BIS     #BIT13,(R1)     ;WRITE IT!
3602  021166  005200                  INC     R0              ;NEXT ADDRESS
3603  021170  022700  002000          CMP     #2000,R0        ;DONE YET?
3604  021174  001364                  BNE     1$              ;BR IF NO
3605  021176  005011                  CLR     (R1)            ;CLEAR SEL0
3606  021200  000207          2$:     RTS     PC              ;RETURN
3607
3608
3609  021202                  MEMSET:
3610                                  ;THIS SUBROUTINE LOADS CRAM WITH SPECIAL INSTRUCTIONS
3611                                  ;FOR THE CRAM JUMP TEST. ALL CRAM LOCATIONS ARE LOADED
3612                                  ;WITH INSTRUCTIONS THAT MOVE A 37 TO THE BR, EXCEPT THE
3613                                  ;FOLLOWING CRAM ADDRESSES: 0,1,4,7,525,1777. THESE LOCATIONS
3614                                  ;CONTAIN INTRUCTIONS WHICH LOAD THE BR WITH THE LOWEST
3615                                  ;8 BITS OF THAT CRAM ADDRESS.
3616
3617  021202  005000                  CLR     R0              ;R0 = CRAM ADDRESS
3618  021204  012711  002000  1$:     MOV     #BIT10,(R1)     ;SET ROM0
3619  021210  010061  000004          MOV     R0,4(R1)        ;LOAD CRAM ADDRESS
3620  021214  012761  000437  000006  MOV     #437,6(R1)      ;LOAD INSTRUCTION
3621  021222  052711  020000          BIS     #BIT13,(R1)     ;WRITE INSTRUCTION IN CRAM
3622  021226  005200                  INC     R0              ;NEXT ADDRESS
3623  021230  022700  002000          CMP     #2000,R0        ;DONE YET?
3624  021234  001363                  BNE     1$              ;BR IF NO
3625  021236  005000                  CLR     R0              ;INDEX REGISTER
3626  021240  012711  002000  2$:     MOV     #BIT10,(R1)     ;SET ROM0
3627  021244  016061  021300  000004  MOV     CRAMA(R0),4(R1) ;LOAD CRAM ADDRESS IN SEL4
3628  021252  016061  021314  000006  MOV     INSTU(R0),6(R1) ;LOAD INSTRUCTIIN TO BE WRITTEN
3629  021260  052711  020000          BIS     #BIT13,(R1)     ;WRITE CRAM!
3630  021264  005720                  TST     (R0)+           ;NEXT
3631  021266  022700  000014          CMP     #14,R0          ;DONE YET?
3632  021272  001362                  BNE     2$              ;BR IF NO
3633  021274  005011                  CLR     (R1)            ;CLEAR ALL BITS
3634  021276  000207                  RTS     PC              ;RETURN
3635
3636  021300  000000  000001  000004  CRAMA:  .WORD   0,1,4,7,1777,525
3637  021306  000007  001777  000525
3638  021314  000400          INSTU:  000400          ;BR+0
3639  021316  000401                  000401          ;BR+1
3640  021320  000404                  000404          ;BR+4
3641  021322  000407                  000407          ;BR+7
3642  021324  000777                  000777          ;BR+377
3643  021326  000525                  000525          ;BR+125
```

```
3644
3645
3646

021330  041600  040522  020115  EM1:    .ASCIZ  <200>/CRAM DATA ERROR/
021351     200  051103  046501  EM2:    .ASCIZ  <200>/CRAM DUAL ADDRESSING ERROR/
021405     200  052512  050115  EM3:    .ASCIZ  <200>/JUMP ERROR/
021421     200  042117  020124  EM4:    .ASCIZ  <200>/OOT ERROR IN IBUS* REG10/
021453     200  047511  020120  EM5:    .ASCIZ  <200>/IOP MAIN MEMORY TEST/
021501     200  047511  020120  EM6:    .ASCIZ  <200>/IOP MAR TEST/
021517     200  051102  051040  EM7:    .ASCIZ  <200>/BR RIGHT SHIFT TEST/

021544  042600  050130  041505  DH1:    .ASCIZ  <200>/EXPECTED  FOUND  ADDRESS/
021576  042600  050130  041505  DH2:    .ASCIZ  <200>/EXPECTED  FOUND/
        021620                  .EVEN

021620  000003                  DT1:    3
021622     006      004                 .BYTE   6,4
021624  001266                          $REG2
021626     006      004                 .BYTE   6,4
021630  001272                          $REG4
021632     004      002                 .BYTE   4,2
021634  001262                          $REG0
021636  000003                  DT2:    3
021640     006      004                 .BYTE   6,4
021642  001274                          $REG5
021644     006      004                 .BYTE   6,4
021646  001272                          $REG4
021650     004      002                 .BYTE   4,2
021652  001266                          $REG2
021654  000002                  DT3:    2
021656     003      007                 .BYTE   3,7
021660  001274                          $REG5
021662     003      002                 .BYTE   3,2
021664  001272                          $REG4
021666  000003                  DT4:    3
021670     003      010                 .BYTE   3,10
021672  001274                          $REG5
021674     003      004                 .BYTE   3,4
021676  001272                          $REG4
021700     004      002                 .BYTE   4,2
021702  021012                          FLAG
021704  000003                  DT5:    3
021706     003      010                 .BYTE   3,10
021710  001274                          $REG5
021712     003      004                 .BYTE   3,4
021714  001272                          $REG4
021716     004      002                 .BYTE   4,2
021720  001266                          $REG2


021722                          CORMAX:
        000001                  .END
```

```
ABASE = 000000            268     309
ACOM1 = 000000            268     311
ACOM2 = 000000            268     312
ACPUOP= 000000            268     283
ADCM0 = 000000            268     313
ADCM1 = 000000            268     314
ADCM10= 000000            268     323
ADCM11= 000000            268     324
ADCM12= 000000            268     325
ADCM13= 000000            268     326
ADCM14= 000000            268     327
ADCM15= 000000            268     328
ADCM2 = 000000            268     315
ADCM3 = 000000            268     316
ADCM4 = 000000            268     317
ADCM5 = 000000            268     318
ADCM6 = 000000            268     319
ADCM7 = 000000            268     320
ADCM8 = 000000            268     321
ADCM9 = 000000            268     322
ADEVCT= 000000            268     274
ADEVM = 000000            268     310
ADRCNT  006057           1334*   1349*   1358*
ADVANC= 104420           1503*   3499
AENV  = 000002              1*    268     279
AENVM = 000000            268     280
AFATAL= 000000            268     271
AMADR1= 000000            268     296
AMADR2= 000000            268     300
AMADR3= 000000            268     303
AMADR4= 000000            268     306
AMAMS1= 000000            268     290
AMAMS2= 000000            268     298
AMAMS3= 000000            268     301
AMAMS4= 000000            268     304
AMSCAD= 000000            268     276
AMSCLC= 000000            268     277
AMSCTY= 000000            268     270
AMTYP1= 000000            268     291
AMTYP2= 000000            268     299
AMTYP3= 000000            268     302
AMTYP4= 000000            268     305
APASS = 000000            268     273
APRIOR= 000000            268
APTCSU= 000040           1059    1164*
APTENV= 000001           1052    1120    1162*   1564
APTSIZ= 000200           1161*
APTSPO= 000100           1054    1122    1163*
APT.SI  013510            727    2138*
ASWREG= 000000            268     281
ATESTN= 000000            268     272
AUDONE  003354            764     785     824*
AUNIT = 000000            268     275
AUSTRT  003126            763*
AUSWR = 000000            268     282
AUTO.S  012110            725    1882*
```

```
AVECT1= 000000            268     307
AVECT2= 000000            268     308
BASE    013726           2149*    2160    2174*   2184*
BINBRD  006412           1406*    1409*   1410    1447*
BIT0   = 000001            129*    1671    1672
BIT00  = 000001            119*     129
BIT01  = 000002            118*     128
BIT02  = 000004            117*     127
BIT03  = 000010            116*     126
BIT04  = 000020            115*     125
BIT05  = 000040            114*     124
BIT06  = 000100            113*     123
BIT07  = 000200            112*     122
BIT08  = 000400            111*     121
BIT09  = 001000            110*     120
BIT1   = 000002            128*     732    1665    1671    1672    2019
BIT10  = 002000            109*    2000    2249    2285    2321    2337    2371    3598    3618    3626
BIT11  = 004000            108*     995
BIT12  = 010000            107*    1954    2017
BIT13  = 020000            106*    1957    2003    2021    2252    2288    2324    3601    3621    3629
BIT14  = 040000            105*     193     977    1968    1970    2021    2032
BIT15  = 100000            104*     672
BIT2   = 000004            127*     732     873    1672
BIT3   = 000010            126*    2023    2030
BIT4   = 000020            125*    1655    1678    1680
BIT5   = 000040            124*    2135    2614
BIT6   = 000100            123*    1660    1661    2025    2629
BIT7   = 000200            122*    1661    1780    2135
BIT8   = 000400            121*    2012    2014    2027    2029    2035    2038    2095    2612    3583
BIT9   = 001000            120*    2009    2035    2038    2093    2095
BM      010631            1703*    1981
BPTVEC= 000014             136*
BRLVL   013442            2090     2100    2108    2121*
BRW     004360             878     1012*
CHRCNT  006410            1404*    1407    1411    1427*   1445*   1446
CKSWR   011212             703     1018    1510    1728*
CKSWR1  011266            1740*    1752
CKSWR2  011300            1743*
CKSWR3  011304            1745*
CKSWR4  011310            1746*    1754    1761
CKSWR5  011414            1729     1731    1735    1770*
CLKX    001452             343*
CLRALL  021014            2672     2685    2698    3157    3170    3183    3218    3231    3244    3279    3292    3305    3340
                          3353     3366    3401    3414    3427    3462    3475    3488    3512*
CMERR   011027             797    1703*
CNT.MA  002302             357     552*    671     673     675    1808
CNVRT  = 104417            794     903     905     907     909    1502*   1545    1547    1622    1743
CONERR  010762             792    1703*
CONN    010671            1703*   1959
CONTAB  003330             801     80?*
CONVRT= 104416             744     800    1501*   1561    1904
CORMAX  021722            3646*
CR     = 000015              44*   1098    1108
CRAMA   021300            3627    3636*
CREAM   001502             356*    670*   1804*   1805    1807*   1811
CRLF   = 000200              45*   1069    1108
```

```
CSR     010363          1703#   1908
CSRMAP  013113          1884#
CYCLP   011460           879     940    1795#
DATABP  006764          1534*   1537    1559    1562#
DATACL= 104413          1498#
DATAHD  006752          1533*   1555    1558#
DDISP = 177570            51#    209#    238
DELAY = 104411          1496#
DEVTAB  003342           767     814#
DH1     021544           388     391     394     403     406    3646#
DH2     021576           397     400     409    3646#
DISPLA  001242           238#    689*    695*   1007*
DISPRE  000174           199#    695
DSWR  = 177570            50#    208#    237
DT1     021620           389     392    3646#
DT2     021636           395    3646#
DT3     021654           398     401     410    3646#
DT4     021666           404    3646#
DT5     021704           407    3646#
DZDME = ****** U         384
DZDMG = 000000           384
EMTVEC= 000030           139#
EM1     021330           387     393    3646#
EM2     021351           390    3646#
EM3     021405           396    3646#
EM4     021421           399    3646#
EM5     021453           402    3646#
EM6     021501           405    3646#
EM7     021517           408    3646#
ERCT00  002304           554#
ERCT01  002310           557#
ERCT02  002314           560#
ERCT03  002320           563#
ERCT04  002324           566#
ERCT05  002330           569#
ERCT06  002334           572#
ERCT07  002340           575#
ERCT10  002344           578#
ERCT11  002350           581#
ERCT12  002354           584#
ERCT13  002360           587#
ERCT14  002364           590#
ERCT15  002370           593#
ERCT16  002374           596#
ERCT17  002400           599#
ERR     003244           776     789#    793
ERRMSG  006740          1532*   1550    1553#
ERRPC   003322           795     806#
ERRVEC= 000004           132#    982     983*    985*    988*
ERTABO  007112          1547    1587#
EXIT  = 000205           159#
EXITER  007046          1573    1578#
FLAG    021012          2401*   2405    2406    2427*   2428    2447*   2452    2453    2475*   2476    2496*   2497    2500
                        2501    2519*   2520    2523*   2524    2527    2528    2543*   2544    3506#   3646
FLOAT   003156           769#    775
FY      003202           777#    781     786
```

```
GNS    = ****** U        1486    1489    1490    1491    1492    1493    1494    1495    1496    1497    1498    1499    1500
                         1501    1502    1503
HALTS    006770          1517    1564#
HILIM    006052          1331#   1339    1355#
HT     = 000011            42#   1067    1108
INCHAR   011420          1746    1774#
INIFLG   001506           361#    698     728     735#
INLPI    005766          1336#   1345
INPUT  = 104415          1500#   1844    1895    1907    1915    1972    1980
INSTU    021314          3628    3638#
INTTY    013456          1925    1944    1960    2129#
IOTVEC=  000020           137#
KMACTV   001470           350#    712     837#    838     1795    1809    1890#   2050#   2056#   2057#   2061    2086    2152#
                         2153
KMCM     011050           803    1703#
KMCR00   002100           467#
KMCR01   002110           472#
KMCR02   002120           477#
KMCR03   002130           482#
KMCR04   002140           487#
KMCR05   002150           492#
KMCR06   002160           497#
KMCR07   002170           502#
KMCR10   002200           507#
KMCR11   002210           512#
KMCR12   002220           517#
KMCR13   002230           522#
KMCR14   002240           527#
KMCR15   002250           532#
KMCR16   002260           537#
KMCR17   002270           542#
KMCSR    002066           450#    763#    778     784#    813     943    1010    1625    1679    1813#   1822    1870    2205
KMCSRH   002070           451#   1660#   1661#   1665#   1671#   1672#   1678#   1680#   1822#   1823#   1824
KMCTL    002072           452#   1824#   1825#   1826
KMNUM    001472           351#    666     920    1888#   1902#   2042#   2043    2051    2053
KMP04    002074           453#   1649#   1655    1692    1697    1826#   1827#   1828
KMP06    002076           454#   1666#   1828#   1829#
KMRLVL   002060           447#    913     914#   1831#   1832#   1833
KMRVEC   002056           446#    913#    946    1814#   1815#   1831
KMS100   002102           468#
KMS101   002112           473#
KMS102   002122           478#
KMS103   002132           483#
KMS104   002142           488#
KMS105   002152           493#
KMS106   002162           498#
KMS107   002172           503#
KMS110   002202           508#
KMS111   002212           513#
KMS112   002222           518#
KMS113   002232           523#
KMS114   002242           528#
KMS115   002252           533#
KMS116   002262           538#
KMS117   002272           543#
KMS200   002104           469#
```

```
KMS201  002114            474#
KMS202  002124            479#
KMS203  002134            484#
KMS204  002144            489#
KMS205  002154            494#
KMS206  002164            499#
KMS207  002174            504#
KMS210  002204            509#
KMS211  002214            514#
KMS212  002224            519#
KMS213  002234            524#
KMS214  002244            529#
KMS215  002254            534#
KMS216  002264            539#
KMS217  002274            544#
KMS300  002106            470#
KMS301  002116            475#
KMS302  002126            480#
KMS303  002136            485#
KMS304  002146            490#
KMS305  002156            495#
KMS306  002166            500#
KMS307  002176            505#
KMS310  002206            510#
KMS311  002216            515#
KMS312  002226            520#
KMS313  002236            525#
KMS314  002246            530#
KMS315  002256            535#
KMS316  002266            540#
KMS317  002276            545#
KMTLVL  002064            449#   915     916*   1835*  1836*
KMTVEC  002062            448#   915*   1833*  1834*  1835
KM.END  002300            547#  1886    2157
KM.MAP  002100            356    466#    670    737    747   1805   1807   1884   1889   2078   2155   2159
LF    = 000012             43#  1102    1108
LINE    010573           1703#  1973
LOBITS  006056           1333#  1357#
LOCK    001444            337#  1009#   1021   1023   1364#  1541   2244#  2278#  2316#  2335#  2363#  2398#  2444#
                         2493#  2522#   2602#  2626#  2667#  2683#  2696#  2728#  2743#  2755#  2786#  2802#  2815#
                         2847#  2863#   2876#  2908#  2924#  2937#  2969#  2985#  2998#  3030#  3046#  3059#  3091#
                         3107#  3120#   3152#  3168#  3181#  3213#  3229#  3242#  3274#  3290#  3303#  3335#  3351#
                         3364#  3396#   3412#  3425#  3457#  3473#  3486#
LOKFLG  001510            363#
LOLIM   006050           1330#  1341    1354#
MASKX   001454            344#
MASTEK  010015           1543   1703#
MCSRX   007745            902   1703#
MDATA   011150           1425   1435    1720#
MEMLIM  001466            349#   861#
MEMSET  021202           2670   2731    2789   2850   2911   2972   3033   3094   3155   3216   3277   3338   3399
                         3460   3609#
MEPASS  007620            901   1703#
MERRPC  010072           1546   1703#
MERRX   007772            908   1703#
MERR2   007645           1703#  2063
```

```
MERR3   007672            834    1703#
MILK    001504            357#    671*    910    1803*  1808*  1812
MLOCK   007716            875    1703#
MNEW    010017            829    1703#
MODU    010461           1703#   1943
MPASSX  007761            906    1703#
MPFAIL  007562           1621    1638    1703#
MR      007642            882    1703#   1860
MRESET= 004000            159#
MSTCLR= 104410           1495#   1627    2204   2280   2318   2365   2400   2446   2495   2562   2604   2669   2730
                         2788    2849    2910   2971   3032   3093   3154   3215   3276   3337   3398   3459
MTITLE  001000            207#    702
MTSTN   010003           1544    1703#   1845
MVECX   007753            904    1703#
NEXT    001442            336#   1363    1583   2202*  2243*  2277*  2315*  2362*  2397*  2443*  2492*  2560*  2601*
                         2666*   2727*   2785*  2846*  2907*  2968*  3029*  3090*  3151*  3212*  3273*  3334*  3395*
                         3456*
NOXCT   010731            714    1703#   1797
NODEV   003240            765     787#
NUM     010323           1703#   1896
OK      003220            779     782#    805
ONE     001464            348#
PACT00  002302            553#
PACT01  002306            556#
PACT02  002312            559#
PACT03  002316            562#
PACT04  002322            565#
PACT05  002326            568#
PACT06  002332            571#
PACT07  002336            574#
PACT10  002342            577#
PACT11  002346            580#
PACT12  002352            583#
PACT13  002356            586#
PACT14  002362            589#
PACT15  002366            592#
PACT16  002372            595#
PACT17  002376            598#
PARBIT= 040000            159#
PERFOR= 004537            159#
PFTAB   007324           1622    1644#
PIRQ  = 177772             49#
PIRQVE= 000240            143#
POPRO = 012600            154#   1577
POP1SP= 005726            152#
POP2SP= 022626            156#
PRIO    010422           1703#   1924
PRIRTY  013730           2145*   2151*   2164   2185#
PRO   = 000000             66#   2121    2122
PR1   = 000040             67#
PR2   = 000100             68#
PR3   = 000140             69#
PR4   = 000200             70#   2123
PR5   = 000240             71#   2124
PR6   = 000300             72#   2125
PR7   = 000340             73#    178     180    182    184    2126
```

```
PS     = 177776          46#    47     663#   872#   2090#  2100#
PSW    = 177776          47#
PUSHRO= 010046           153#   1574
PUSH1S= 005746           151#
PUSH2S= 024646           155#
PWRVEC= 000024           138#   1600#  1601#  1610#  1616#  1635#  1636#
QV.FLG  001511           364#    669#   919#
ROMDAT  021106          2677    2690   2703   2737   2749   2761   2796   2809   2822   2857   2870   2883   2918
                        2931    2944   2979   2992   3005   3040   3053   3066   3101   3114   3127   3162   3175
                        3188    3223   3236   3249   3284   3297   3310   3345   3358   3371   3406   3419   3432
                        3467    3480   3493   3576#
ROCHR  = 104402         1216    1489#
ROLIN  = 104403         1289    1490#
ROOCT  = 104404         1338    1491#
RESREG  006766          1560    1563#
RESVEC= 000010           133#
RESOS  = 104407         1494#   1563
ROMCLK= 104412          1497#   1650   1653   1690   1695   2209   2211   2213   2221   2223   2407   2409   2412
                        2414    2416   2454   2456   2459   2461   2463   2502   2504   2507   2509   2511   2529
                        2531    2533   2535   2564   2567   2573   2576   2606   2609   2621   2627   2635   2637
                        2639    2673   2675   2686   2688   2699   2701   2733   2735   2745   2747   2757   2759
                        2792    2794   2805   2807   2818   2820   2853   2855   2866   2868   2879   2881   2914
                        2916    2927   2929   2940   2942   2975   2977   2988   2990   3001   3003   3036   3038
                        3049    3051   3062   3064   3097   3099   3110   3112   3123   3125   3158   3160   3171
                        3173    3184   3186   3219   3221   3232   3234   3245   3247   3280   3282   3293   3295
                        3306    3308   3341   3343   3354   3356   3367   3369   3402   3404   3415   3417   3428
                        3430    3463   3465   3476   3478   3489   3491   3515   3517   3519   3527   3535   3543
                        3551    3559   3561   3563   3571   3586
ROMMAP  013732          2187#   2368   3597
RUN     001500           354#    672#  1801#  1802#  1809
SAVACT  001474           352#    832   2061#  2153#
SAVNUM  001476           353#    666#   917#   920#  2054#  2147#
SAVPC   001460           346#    793#   808   1370#  1589
SAVSP   001456           345#
SAVOS  = 104406         1493#   1522
SCOPI  = 104405         1492#   2257   2293   2330   2343   2379   2423   2470   2518   2542   2620   2682   2695
                        2708    2742   2754   2766   2801   2814   2827   2862   2875   2888   2923   2936   2949
                        2984    2997   3010   3045   3058   3071   3106   3119   3132   3167   3180   3193   3228
                        3241    3254   3289   3302   3315   3350   3363   3376   3411   3424   3437   3472   3485
                        3498
SETBR0  021032          2913    2926   2939   3524#
SETBR1  021040          2974    2987   3000   3532#
SETBR4  021046          3035    3048   3061   3540#
SETBR7  021054          3096    3109   3122   3548#
SETC    021062          2791    2804   2817   3556#
SETZ    021100          2852    2865   2878   3568#
SOFTSW  011452          1744    1783#
SPACNT= 006411          1405*   1429   1432*  1446#
STACK  = 001200           37#    664    851   1584
STAT    001450           342#
STAT1   002050           439#   1816*
STAT2   002052           440#   1817*
STAT3   002054           441#   1818*
STKLMT= 177774            48#
STRTSW  001446           341#    704*   707*   708    710    724*   730    732    827    873    880   1838   1861*
                        1891    2074
```

```
SVOS    006100          1374#
SWFLG   011416           667*  1739*  1766*  1772#
SWR     001240           237#   688*   690    694*   704    832    837    977    995   1019   1511   1516   1572
                        1579   1581   1608   1628*  1668   1728   1765*
SWREG   000176           200#   683*   694   1728   1785
SW0   = 000001           101#
SW00  = 000001            91#   101    708    724   1891   2074
SW01  = 000002            90#   100    880   1838   1861
SW02  = 000004            89#    99
SW03  = 000010            88#    98    827
SW04  = 000020            87#    97
SW05  = 000040            86#    96
SW06  = 000100            85#    95   1668
SW07  = 000200            84#    94
SW08  = 000400            83#    93   1579
SW09  = 001000            82#    92   1019
SW1   = 000002           100#
SW10  = 002000            81#  1581
SW11  = 004000            80#
SW12  = 010000            79#  1511
SW13  = 020000            78#  1516
SW14  = 040000            77#
SW15  = 100000            76#
SW2   = 000004            99#
SW3   = 000010            98#
SW4   = 000020            97#
SW5   = 000040            96#
SW6   = 000100            95#
SW7   = 000200            94#
SW8   = 000400            93#
SW9   = 001000            92#
TBITVE= 000014           134#
TEMP    011106          1412   1676*  1681*  1687*  1699*  1718#
TIMER = 104414          1499#
TKVEC = 000060           141#
TLAST = 020624          1864   3507#
TPVEC = 000064           142#
TRAPVE= 000034           140#
TRTVEC= 000014           135#
TST1    013732           971   1851   1869   2200#
TST10   015102          2443   2490#
TST11   015372          2492   2558#
TST12   015476          2560   2599#
TST13   015674          2601   2664#
TST14   016060          2666   2725#
TST15   016230          2727   2783#
TST16   016414          2785   2844#
TST17   016600          2846   2905#
TST2    014044          2202   2241#
TST20   016764          2907   2966#
TST21   017150          2968   3027#
TST22   017334          3029   3088#
TST23   017520          3090   3149#
TST24   017704          3151   3210#
TST25   020070          3212   3271#
TST26   020254          3273   3332#
```

```
TST27   020440            3334    3393#
TST3    014150            2243    2275#
TST30   020624            3395    3454#   3507
TST31 = ****** U          3456
TST4    014262            2277    2313#
TST5    014432            2315    2360#
TST6    014542            2362    2395#
TST7    014720            2397    2441#
TTST    004146             876#    878#    973#
TWOSYM= 010000             159#
TYPDAT  006754            1538    1556    1559#
TYPE   = 104401            702     714     736     791     796     802     829     834     875     882     901     902     904
                           906     908    1072    1223    1229    1234    1238    1243    1244    1246    1249    1253    1318
                          1320    1336    1343    1344    1395    1435    1486#   1539    1540    1543    1544    1546    1548
                          1552    1557    1621    1637    1742    1745    1797    1843    1860    1866    1902    1923    1937
                          1942    1951    1958    1965    2063
TYPMSG  006654            1536    1539#
VEC     010401            1703#   1916
VECMAP  013172            2062    2074#
VECTR   013724            2144#   2150#   2169    2175#   2183#
WHAT    005770            1329#   1337#
WHERE   006054            1332#   1346    1356#
WHICH   013164            1905    2070#
WRDCNT  006406            1403#   1436#   1444#
WRKO.F  006742            1551    1554#
WROM    021140            2367    3591#
XBX     006542            1512    1514    1516#
XCSR    004104             903     941#
XERR    004126             909     950#
XHEAD   010077             736    1703#
XPASS   004120             907     947#
XSTATO  011060             745    1703#
XTSTN   007120            1545    1590#
XVEC    004112             905     944#
ZERO    001462             347#
$APTHD  002034             421     427#
$ASTAT= ****** U          1142    1157
$ATYC   004722            1113    1115#
$ATY1   004676            1111#
$ATY3   004704            1057    1112#
$ATY4   004714            1114#   1567
$AUTOB  001234             234#
$BAUE   001372             309#   2149
$BDFOR  001222             229#
$BDOAT  001226             231#
$CDW1   001376             311#   2146
$CDW2   001400             312#
$CHARC  004672            1074#   1084#   1091    1100#   1105#
$CKSWR= ****** U          1489
$CMTAG  001200             217#
$CM1   = 000006            249#    250#    251#    252#    253#    254#    255#
$CM2   = 000014            249#    250#    251#    252#    253#    254#    255#
$CM3   = 000006            247#    249
$CM4   = 000005            255#    256#    257#    258#    259#    260#
$CNTLG  005534            1264#
$CNTLU  005527            1238    1263#
```

```
$COD  = ****** U          1
$CPUOP  001344          283#
$CRAP = 177777            1#  2189#  2192   2195#  2231#  2234   2236#  2265#  2268   2270#  2302#  2305   2308#
                       2348#  2351   2355#  2385#  2388   2390#  2431#  2434   2436#  2479#  2482   2485#  2547#
                       2550   2553#  2587#  2590   2594#  2648#  2651   2659#  2709#  2712   2720#  2767#  2770
                       2778#  2828#  2831   2839#  2889#  2892   2900#  2950#  2953   2961#  3011#  3014   3022#
                       3072#  3075   3083#  3133#  3136   3144#  3194#  3197   3205#  3255#  3258   3266#  3316#
                       3319   3327#  3377#  3380   3388#  3438#  3441   3449#
$CRLF   001313          262#  1073   1108   1243   1263   1323   1344   1395   1539   1540   1548   1843   1903
$DDW0   001402          313#  2154
$DDW1   001404          314#
$DDW10  001426          323#
$DDW11  001430          324#
$DDW12  001432          325#
$DDW13  001434          326#
$DDW14  001436          327#
$DDW15  001440          328#
$DDW2   001406          315#
$DDW3   001410          316#
$DDW4   001412          317#
$DDW5   001414          318#
$DDW6   001416          319#
$DDW7   001420          320#
$DDW8   001422          321#
$DDW9   001424          322#
$DEVCT  001326          274#
$DEVM   001374          310#  2152
$DOAGN  004100          918    927    932    938#
$ENDAD  004070          191    700    934#  1570
$ENDCT  004054          929#
$ENV    001336          279#   681    717    719   1052   1120   1144   1564   1730
$ENVM   001337          280#   722   1054   1059   1122
$EOP    003662          897#  3456
$EOPCT  004046          926#   930
$ERFLG  001203          220#   668*   900*   962    991    993*   1014   1521*  1535   1549*  1623*
$ERMAX  001215          226#  1014
$ERROR  006512          181   1510#
$ERRPC  001216          227#   677*   921*   970*  1518   1520*  1624*
$ERRTB  001512          381#  1531
$ERTTL  001212          224#   912    952   1578*  1820*
$ETABL  001336          278#
$ETEND  001442          331#   433
$FATAL  001320          271#  1148*
$FFLG   005142         1111*  1114*  1142   1151*  1159#
$FILLC  001256          245#  1077   1108
$FILLS  001255          244#  1108
$GOADR  001220          228#
$GOOAT  001224          230#
$GET42  004060          931#
$GTSWR= ****** U        1488
$HD   = 000000            11
$HIBTS  002034          428#
$HIOCT  005722         1311*  1322#
$ICNT   001204          221#   999*  1000   1002*  1013
$ILLUP  007316         1600   1616   1640#
$INPUT  005724         1327#  1500
```

```
$INTAG  001235          235#
$ITEMB  001314          225#  1526*  1566
$LF     001314          263#  1108   1253   1263   1323
$LFLG   005141         1152*  1158#
$LPADR  001206          222#   679*   879*   883   1006*  1008   1013   1583*  1585   1859*  1869*  1871
$LPERR  001210          223#
$MADR1  001350          296#
$MADR2  001354          300#
$MADR3  001360          303#
$MADR4  001364          306#
$MAIL   001316          269#   429    433    1005   1052
$MAMS1  001346          290#  2148
$MAMS2  001352          298#
$MAMS3  001356          301#
$MAMS4  001362          304#
$MBADR  002036          429#
$MFLG   005140         1112*  1118   1153*  1157#
$MNEW   005552         1267#  1745
$MSGAD  001332          276#  1128*  1131
$MSGLG  001334          277#  1133*
$MSGTY  001316          270#  1126   1134*  1146   1150*
$MSWR   005541         1265#  1742
$MTYP1  001347          291#
$MTYP2  001353          299#
$MTYP3  001357          302#
$MTYP4  001363          305#
$MXCNT  004362         1003    1013#
$N    = 000030             1#  2189   2195   2197   2204#  2231   2236   2238   2246#  2265   2270   2272   2280
                        2281#  2302   2308   2310   2318   2319#  2348   2355   2357   2365   2366#  2385   2390
                        2392   2400   2401#  2431   2436   2438   2446   2447#  2479   2485   2487   2495   2496#
                        2547   2553   2555   2562   2563#  2587   2594   2596   2604   2605#  2648   2659   2661
                        2669   2670#  2709   2720   2722   2730   2731#  2767   2778   2780   2788   2789#  2828
                        2839   2841   2849   2850#  2889   2900   2902   2910   2911#  2950   2961   2963   2971
                        2972#  3011   3022   3024   3032   3033#  3072   3083   3085   3093   3094#  3133   3144
                        3146   3154   3155#  3194   3205   3207   3215   3216#  3255   3266   3268   3276   3277#
                        3316   3327   3329   3337   3338#  3377   3388   3390   3398   3399#  3438   3449   3451
                        3459   3460#  3507#
$NULL   001254          243#  1079   1108
$NWTST= 000000         2199#  2240#  2274#  2312#  2359#  2394#  2440#  2489#  2557#  2598#  2663#  2724#  2782#
                       2843#  2904#  2965#  3026#  3087#  3148#  3209#  3270#  3331#  3392#  3453#
$OVER   004334          975    978    989   1001   1007#
$PASS   001324          273#   899*   911    923*   924*   941    949    997   1014   1819*
$PASTM  002042          431#
$PWRON  007126          179    665   1600#  1635
$PWRMG  007312         1638#
$PWRUP  007200         1610    1616#
$QUES   001312          261#  1108   1246   1263   1320   1323   1343   1866   1938   1952   1966
$ROCHR  005144         1183#  1489
$ROOEC= ****** U       1492
$ROLIN  005264         1211#  1490
$ROOCT  005564         1284#  1491
$ROSZ = 000007         1204#
$REGAD  001260          247#
$REGO   001262          249#  1379*  1384   3646
$REG1   001264          250#   799*   811   1378*  1385
$REG2   001266          251#  1377*  1386   3646
```

```
$REG3   001270          252#  1376*  1387
$REG4   001272          253#  1375*  1388   3646
$REG5   001274          254#  1374*  1389   3646
$RTNAD  004102          940#
$R2A  = ****** U       1492
$S    = 000032            1#  2202   2204#  2243   2246#  2277   2281#  2315   2319#  2362   2366#  2397   2401#
                       2443   2447#  2492   2496#  2560   2563#  2601   2605#  2666   2670#  2727   2731#  2785
                       2789#  2846   2850#  2907   2911#  2968   2972#  3029   3033#  3090   3094#  3151   3155#
                       3212   3216#  3273   3277#  3334   3338#  3395   3399#  3456   3460#
$SAVRE= ****** U       1492
$SAVR6  007322         1609*  1617   1618*  1619*  1642#
$SCOPE  004134          177    969#  2118
$SETUP= 000000          922    970   1172   1269
$SVLAD  004316          986   1004#
$SVPC = 000040          189#   194
$SWR  = 164000            1#    11    260    261    894    922    933    939    941    963    964    965    966
                        977    989    991    992    993    994    995   1007   1013   1639   2201   2242   2276
                       2314   2361   2396   2442   2491   2559   2600   2665   2726   2784   2845   2906   2967
                       3028   3089   3150   3211   3272   3333   3394   3455
$SWREG  001340          281#   683
$SWRMK= 000000          966
$TESTN  001322          272#  1005*
$TIMES  001310          260#   922*   994*  1000   1003*  1013
$TKB    001246          240#   976   1170   1187   1193   1732   1734   1776   2131
$TKS    001244          239#   974   1170   1185   1191   1774   2129
$TMP0   001276          255#   738*  1705   2086*  2087*
$TMP1   001300          256#   739*  1707
$TMP2   001302          257#   741*   762*   789    798*  1709   1899   1902   1989*
$TMP3   001304          258#   742*  1711   1911   1914   1919   1922   1976   1979   1984   1987
$TMP4   001306          259#   743*  1713   1894*  1906*  2072
$TN   = 000031            1#    11   2199   2201#  2240   2242#  2274   2276#  2312   2314#  2359   2361#  2394
                       2396#  2440   2442#  2489   2491#  2557   2559#  2598   2600#  2663   2665#  2724   2726#
                       2782   2784#  2843   2845#  2904   2906#  2965   2967#  3026   3028#  3087   3089#  3148
                       3150#  3209   3211#  3270   3272#  3331   3333#  3392   3394#  3453   3455#
$TPB    001252          242#  1097*  1108   1515*  1779*  2134*
$TPFLG  001257          246#  1046   1108
$TPS    001250          241#  1095   1108   1513   1777   2132
$TRAP   006414          183   1463#
$TRAP2  006436         1474#  1485
$TRP  = 000021         1478#  1487#  1489   1490#  1491#  1492#  1493#  1494#  1495#  1496#  1497#  1498#  1499#
                       1500#  1501#  1502#  1503#  1504#
$TRPAD  006450         1468   1485#
$TSTM   002040          430#
$TSTNM  001202          219#   678*   962   1004*  1005   1007   1014   1592   1646   1848   1855   1857   2201#
                       2242#  2276#  2314#  2361#  2396#  2442#  2491#  2559#  2600#  2665#  2726#  2784#  2845#
                       2906#  2967#  3028#  3089#  3150#  3211#  3272#  3333#  3394#  3455#
$TTYIN  005520         1213   1214   1226   1244   1258   1262#
$TYPBN= ****** U       1487
$TYPOS= ****** U       1487
$TYPE   004414         1046#  1139   1478   1486
$TYPEC  004626         1076   1083   1090   1095#  1096
$TYPEX  004674         1101   1103   1106#
$TYPOC= ****** U       1487
$UNIT   001330          275#
$UNITM  002044          432#
$USWR   001342          282#
```

```
$VECT1  001366            307#   2150    2151
$VECT2  001370            308#
$XTSTR  004174            972     980#
$Y    = 000000              1#    434#
$$GET4= 000000            933#
$40CAT= ****** U          977
.     = 021722            171#    173     176#    189     190#    192#    194#    198#    202#    206#    216#    264     350#
                          351#    352#    353#    362#    411#    417     418#    420#    422#    465#    467#    468#    469#
                          470#    472#    473#    474#    475#    477#    478#    479#    480#    482#    483#    484#    485#
                          487#    488#    489#    490#    492#    493#    494#    495#    497#    498#    499#    500#    502#
                          503#    504#    505#    507#    508#    509#    510#    512#    513#    514#    515#    517#    518#
                          519#    520#    522#    523#    524#    525#    527#    528#    529#    530#    532#    533#    534#
                          535#    537#    538#    539#    540#    542#    543#    544#    545#    706     716     823#    836
                          941    1013    1014    1108    1160#   1170    1262#   1263    1269#   1323    1612    1641    1684#
                         1719#   1721#   1775    1778    1799    1892    1933    2016    2020    2066    2097    2130    2133
                         2613    2618    2633    2645    3646#
.ADVAN  006060           1363#   1503
.BEGIN  003464            851#
.CNVRT  006170           1396#   1502
.CONVR  006164           1395#   1501
.DATAC  007446           1498    1675#
.DELAY  007332           1496    1648#
.MSTCL  007362           1495    1659#
.RESOS  006132           1384#   1494
.ROMCL  007400           1497    1664#
.SAVOS  006072           1370#   1493
.SCOPI  004364           1018#   1492
.START  002402            203     663#    679     1762
.TIMER  007512           1499    1686#
.$ASTA= ****** U         1112    1115
.$X   = 002034            417#    422
```

```
COMMEN  1448
ENDCOM  1448
ERROR     38#  2219  2229  2256  2292  2329  2342  2376  2422  2469  2517  2541  2582  2619  2634
          2646  2681  2694  2707  2741  2753  2765  2800  2813  2826  2861  2874  2887  2922  2935
          2948  2983  2996  3009  3044  3057  3070  3105  3118  3131  3166  3179  3192  3227  3240
          3253  3288  3301  3314  3349  3362  3375  3410  3423  3436  3471  3484  3497
ESCAPE  1448
GETPRI  1448
GETSWR  1448
HLT     1578
KMEND      1#   884
KMFRNT     1#
MULT    1448
NEWTST  1448#  2199  2240  2274  2312  2359  2394  2440  2489  2557  2598  2663  2724  2782  2843
          2904  2965  3026  3087  3148  3209  3270  3331  3392  3453
POP     1448#  1154  1155  1312  1628  1629  2177
PUSH    1448#  1115  1117  1138  1286  1602  1608  2140
REPORT     1#   1448
SCOPE     39#  2200  2241  2275  2313  2360  2395  2441  2490  2558  2599  2664  2725  2783  2844
          2905  2966  3027  3088  3149  3210  3271  3332  3393  3454
SETPRI  1448
SETTRA  1478#  1489  1490  1491  1492  1493  1494  1495  1496  1497  1498  1499  1500  1501  1502
          1503
SETUP   1448
SKIP    1448
SLASH   1448
SPACE   1448
STARS   1448#   187   212   264   267   414   416   423   892   959  1031  1110  1169  1175  1204
          1272  1457  1598  1614  2199  2240  2274  2312  2359  2394  2440  2489  2557  2598  2663
          2724  2782  2843  2904  2965  3026  3087  3148  3209  3270  3331  3392  3453
SWRSU   1448
TRMTRP  1478#
TYPBIN  1448
TYPDEC  1448
TYPNAM  1448
TYPNUM  1448
TYPOCS  1448
TYPOCT  1448
TYPTXT  1448
$AUTO      1#   748
$BRRSH     1#  2189
$BUFFE     1#  1715
$BYTE      1#
$CKDAT     1#
$COMP      1#
$CRAM      1#  2231  2265
$CRAMD     1#  2302
$CYCLE     1#  1786
$DATAF     1#
$EOP       1#   884
$ERTBL     1#   383
$EXER      1#
$FD        1#
$FINI      1#  3507
$GETPA     1#
$HALF      1#
```

```
$HD          1#
$HEADE       1#    11
$IOPOD       1#  2587
$JUMP        1#  2648    2709    2767    2828    2889    2950    3011    3072    3133    3194    3255    3316    3377    3438
$LSTDA       1#
$MARHI       1#
$MEMFL       1#  2401    2447
$MEMO        1#  2431
$MEM1        1#  2385
$MEM2        1#  2479
$MEM3        1#  2547
$MOCK        1#
$MSG         1#  1703
$NONEX       1#
$ORUN        1#
$PASEN       1#   898
$PFAIL       1#  1593
$PROC        1#
$PROC1       1#
$QUEST       1#  1895    1907    1915    1972    1980
$RAMCL       1#  1647
$RCLK        1#  1650    1653    1690    1695    2209    2211    2213    2220    2223    2407    2409    2412    2414    2416
             2454    2456    2459    2461    2463    2502    2504    2507    2509    2511    2529    2531    2533    2535    2564
             2567    2573    2576    2606    2609    2621    2627    2635    2637    2639    2673    2675    2686    2688    2699
             2701    2733    2735    2745    2747    2757    2759    2792    2794    2805    2807    2818    2820    2853    2855
             2866    2868    2879    2881    2914    2916    2927    2929    2940    2942    2975    2977    2988    2990    3001
             3003    3036    3038    3049    3051    3062    3064    3097    3099    3110    3112    3123    3125    3158    3160
             3171    3173    3184    3186    3219    3221    3232    3234    3245    3247    3280    3282    3293    3295    3306
             3308    3341    3343    3354    3356    3367    3369    3402    3404    3415    3417    3428    3430    3463    3465
             3476    3478    3489    3491    3515    3517    3519    3527    3535    3543    3551    3559    3561    3563    3571
             3586
$ROROM       1#  2348
$ROMRD       1#
$ROVAR       1#   333
$SCA00       1#   970
$SCA01       1#  1009
$SETUP       #
$SIMBC       .#
$SKIPT       .#
$SOFTC       1#  1723
$TSTN        1#  2197    2238    2272    2310    2357    2392    2438    2487    2555    2596    2661    2722    2780    2841
             2902    2963    3024    3085    3146    3207    3268    3329    3390    3451
$UPA00       1#  1621
$VARIA       1#   205
$XZ          1#  2189    2195    2231    2236    2265    2270    2302    2308    2348    2355    2385    2390    2431    2436
             2479    2485    2547    2553    2587    2594    2648    2659    2709    2720    2767    2778    2828    2839    2889
             2900    2950    2961    3011    3022    3072    3083    3133    3144    3194    3205    3255    3266    3316    3327
             3377    3388    3438    3449
$$CMRE     210#   249     250     251     252     253     254
$$CMTM     210#   255     256     257     258     259
$$ESCA     144#
$$NEXT     144#  2199    2240    2274    2312    2359    2394    2440    2489    2557    2598    2663    2724    2782    2843
             2904    2965    3026    3087    3148    3209    3270    3331    3392    3453
$$SCOP       1#   953
$$SET      1478#  1489    1490    1491    1492    1493    1494    1495    1496    1497    1498    1499    1500    1501    1502
             1503
```

```
$$SKIP    144#
.EQUAT      1#      34
.HEADE      1#
.SETUP      1#
.$ACT1      1#     185
.$APTB      1#     265#
.$APTH      1#     412
.$APTY      1#    1108
.$CATC      1#
.$CMTA      1#     210
.$EOP       1#     890
.$ERRO      1#
.$ERRT      1#
.$POWE      1#    1596
.$ROOC      1#    1270
.$READ      1#    1167
.$SCOP      1#     957
.$TRAP      1#    1455
.$TYPE      1#    1029
.$TYPO      1#

. ABS.  021722     000

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DZKCD,DZKCD/SOL/CRF=DZKCD.MAC,DZKCD.P11/EQ:DZDMG
 RUN-TIME: 25 19 1 SECONDS
 RUN-TIME RATIO: 82/46=1.7
 CORE USED:  51k  (102 PAGES)

EOF1DZKCDASEQ          00C100C0      770720          PDP10 411
```