

IDENTIFICATION

PRODUCT CODE: AC-E107A-MC
PRODUCT NAME: CZKCgA0 KMC FREE RUNNING TEST
DATE: MAY 1978
MAINTAINER: DIAGNOSTICS
AUTHOR: Ed Badger

COPYRIGHT (C) 1978 BY DIGITAL EQUIPMENT CORPORATION
THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES
NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A
LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE
TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR
THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS
NOT SUPPLIED BY DIGITAL.

1. ABSTRACT

The function of the KMC11 diagnostics is to verify that the option operates according to specifications. The diagnostics verify 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.

CZDMG tests the KMC11 micro-processor Free running tests are performed. A line unit (M8201 or M8202) must be installed. CZDMG can be used as a heat test diagnostic by manufacturing.

Currently there are five 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 five diagnostics are:

1. DZKCA KMC-11 CPU Micro-Diagnostics
2. DZKCC Basic W/R and Micro-Processor Tests
3. DZKCD KMC-11 Low speed jump and memory tests
4. DZKCE DDCMP mode line unit tests
5. DZKCF Bitstuff mode line unit tests

2. REQUIREMENTS

2.1 EQUIPMENT

Any PDP11 family CPU (except an LSI-11) with minimum 8k memory
ASR 33 (or equivalent)
KMC11 or
M8201 or M8202 line unit

2.2 STORAGE

Program will use all 8K of memory except where ABL and BOOTSTRAP LOADER reside. Locations 1500 thru 1640; 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 PROCEDURE

- 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 DMC11 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 DMC11 STATUS

PC	CSR	STAT1	STAT2	STAT3
001500	160010	145310	177777	000000
001510	160020	145320	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. 1500 in the program. In this example the table contains the information and status of two DMC11'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

DOES MICRO-PROCESSOR HAVE CRAM? (Y OR N)N

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 carriage 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 'DMACTV' from the switch register. Using this switch(SW00) alters that location;therefore if four DMC11s are in the system ***DO NOT*** set switches greater than SW 03 in the up position. this would be a fatal error. do not select more active DMC11s 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 DMC'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 'SCOP1') 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 enableed; and there is a HARD error (constant); SW08 is best. (SW14=1,0, SW10=0, SW09=0, SW08=1). for intermittent errors; SW14=1 will loop on test regardless 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 possibly 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 'TSTNO' (address 1226) 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 DMC11 was doing at the time of the error.

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

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

KMC(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 DMC11's in system are tested. When the diagnostic has completed a pass the following is an example of the print out to be expected.

```
ENC PASS DZDMG CSR: 175000 VEC: 0300 PASSES: 000001
ERRORS: 000000
```

NOTE: The pass count and error counts are cummulative 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 DMC11 since the diagnostic was started are reflected in PASSES: and ERRORS:.

8.4 KEY LOCATIONS

RETURN (1214) Contains the address where program will return when iteration count is reached or if loop on test is asserted.

NEXT (1216) Contains the address of the next test to be performed.

TSTNO (1226) Contains the number of the test now being performed.

RUN (1316) The bit in 'RUN' always points to the KMC11 currently being tested. EXAMPLE: (RUN) 1302/0000000001000000 Means that KMC11 no.06 is the KMC11 now running.

DMCROO-DMCR17
DMSTOO-DMST17
(1500)-(1640)

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.

DMACTV (1306) Each bit set in this location indicates that the associated KMC11 will be tested in turn. EXAMPLE: (DMACTV) 1276/0000000000011111 means that KMC11 no. 00,01,02,03,04 will be tested. EXAMPLE: (DMACTV) 1276/0000000000010001 Means that DMC11 no. 00,04 will be tested.

DMCSR (1404) Contains the CSR of the current KMC11 under test.

8.4A 'STATUS TABLE' (1500-1640)

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 DMC11'S. the table can contain up to 16 DMC11'S. Following the map is a description of the bits for each map entry

MAP OF KMC11 STATUS

PC	CSR	STAT1	STAT2	STAT3
001500	160010	145310	177777	000000
001510	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, 1500, 1502, 1504, and 1506. The second KMC status is located at 1510, 1512, 1514, and 1516. 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
BIT15=1 MICRO-PROCESSOR HAS CROM
BIT15=0 MICRO-PROCESSOR HAS CROM
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: BIT1=0 KMC11-AR (LOW SPEED)
BIT1=1 KMC11-AL (HIGH SPEED)

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 CROM address is written to a 125252 then it is read back. If it contains a -1 or 125252 or a 626 or 16520 a DMC11 or KMC11 has been found, if not, the address is updated by 10 and the search continues. A -1 indicates a DMC11 with no CROM, a 125252 indicates a KMC11 with CROM, a 626 indicates a DMC11-AL and a 16520 indicates a DMC11-AR. 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 DMC11's in the system will be found by the auto-sizer. If it does not find a DMC11 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 DMC is programmed to interrupt. The PS is lowered by 1 until the DMC interrupts, a delay is made and if no interrupt occurs at PS level 3 (because of a bad DMC11) 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 interrupt occurred; the address to which the DMC11 interrupted 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.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

01200
01300
01400
01500
01600
01700
01800
01900
02000
02100
02200
02300
02400
02500
02600
02700
02800
02900
03000
03100
03200
03300
03400
03500
03600
03700
03800
03900
04000
04100
04200
04300
04400
04500
04600
04700
04800
04900
05000
05100
05200
05300
05400
05500
05600
05700
05800
05900
06000
06100
06200

***** .\$\$SCOPE *****

.\$SCOPE IS USED TO HANDLE SCOPE LOOPS

ARGUMENT:

- 1) NUM ---- IF NON-BLANK DESIRED NUMBER OF ITERATIONS
IF BLANK 2000. ITERATIONS WILL BE MADE
- 2) INSTR -- IF NON-BLANK WILL BE THE FIRST INSTRUCTION OF
THE SCOPE ROUTINE
EXAMPLES OF USE:
1) <<MOV R1,SAVR1 ;:SAVE R1>>
2) <<MOV R1,SAVR1>,<MOV R2,SAVR2>>
3) AS A MACRO I.E. <<PUSH <R0,R1,R2,R3,R4,R5>>>
- 3) NOLOOP - IF BLANK THE FIRST PASS THROUGH THE PROGRAM WILL
INHIBIT ITERATIONS.
IF NON-BLANK ITERATIONS WILL OCCUR ON THE FIRST PASS.
- 4) INSTR2 - IF NON-BLANK WILL REPLACE THE LAST INSTRUCTION (RTI).
REFER TO ARGUMENT 2 (INSTR) FOR EXAMPLE OF USE.
REMEMBER YOU NEED AN RTI (OR RTS) FOR EXITING THE ROUTINE.
- 5) TABLE - IF THIS ARGUMENT IS IDENTICAL TO THE WORD "SW08TBL"
AND THE SWITCH 8 (SW08) SCOPE OPTION IS TO BE USED
A DISPATCH TABLE WILL BE CREATED. IF SW08 IS ON A "1"
THE LOWER BYTE OF THE SWITCH REGISTER WILL BE USED TO
INDEX INTO THE DISPATCH TABLE AND SELECT THE STARTING
ADDRESS OF THE SPECIFIED TEST. THE TABLE IS OF THE FORM:

```
$$SW08TBL:  
    .WORD  TST1+2  
    .WORD  TST2+2  
    .  
    .  
    .WORD  TSTN+2
```

NOTE: THIS ROUTINE IS CONDITIONALLY ASSEMBLE BY \$SWR
FOR SW14,SW11,SW09,\$SW08
SW14=1 LOOP ON TEST
SW11=1 INHIBIT ITERATIONS
SW09=1 LOOP ON ERROR
SW08=1 LOOP ON TEST IN SW<7:0>

52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107

001200

000011
000012
000015
000200
177776

177774
177772
177570
177570

000000
000001
000002

```
.TITLE AC-E107A-MC
;*COPYRIGHT (C) 1978
;*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.
;*

;*AC-E107A-MC CZKCGAO KMC FREE RUNNING TEST
;*COPYRIGHT 1978, 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 'AC-E107A-MC CZKCGAO KMC FREE RUNNING TEST'
;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 ***
STACK= 1200
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

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

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


BASIC DEFINITIONS

```
108      000003      R3=      %3      ;;GENERAL REGISTER
109      000004      R4=      %4      ;;GENERAL REGISTER
110      000005      R5=      %5      ;;GENERAL REGISTER
111      000006      R6=      %6      ;;GENERAL REGISTER
112      000007      R7=      %7      ;;GENERAL REGISTER
113      000006      SP=      %6      ;;STACK POINTER
114      000007      PC=      %7      ;;PROGRAM COUNTER
115
116      ;*PRIORITY LEVEL DEFINITIONS
117      000000      PR0=      0      ;;PRIORITY LEVEL 0
118      000040      PR1=      40     ;;PRIORITY LEVEL 1
119      000100      PR2=      100    ;;PRIORITY LEVEL 2
120      000140      PR3=      140    ;;PRIORITY LEVEL 3
121      000200      PR4=      200    ;;PRIORITY LEVEL 4
122      000240      PR5=      240    ;;PRIORITY LEVEL 5
123      000300      PR6=      300    ;;PRIORITY LEVEL 6
124      000340      PR7=      340    ;;PRIORITY LEVEL 7
125
126      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
127      100000      SW15=     100000
128      040000      SW14=      40000
129      020000      SW13=      20000
130      010000      SW12=      10000
131      004000      SW11=       4000
132      002000      SW10=       2000
133      001000      SW09=       1000
134      000400      SW08=        400
135      000200      SW07=        200
136      000100      SW06=        100
137      000040      SW05=         40
138      000020      SW04=         20
139      000010      SW03=          10
140      000004      SW02=           4
141      000002      SW01=           2
142      000001      SW00=           1
143      .EQUIV      SW09,SW9
144      .EQUIV      SW08,SW8
145      .EQUIV      SW07,SW7
146      .EQUIV      SW06,SW6
147      .EQUIV      SW05,SW5
148      .EQUIV      SW04,SW4
149      .EQUIV      SW03,SW3
150      .EQUIV      SW02,SW2
151      .EQUIV      SW01,SW1
152      .EQUIV      SW00,SW0
153
154      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
155      100000      BIT15=     100000
156      040000      BIT14=      40000
157      020000      BIT13=      20000
158      010000      BIT12=      10000
159      004000      BIT11=       4000
160      002000      BIT10=       2000
161      001000      BIT09=       1000
162      000400      BIT08=        400
163      000200      BIT07=        200
```


BASIC DEFINITIONS

164 000100
165 000040
166 000020
167 000010
168 000004
169 000002
170 000001

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

181
182
183 000004
184 000010
185 000014
186 000014
187 000014
188 000020
189 000024
190 000030
191 000034
192 000060
193 000064
194 000240
195
196
197
198
199

;*BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14 ;: "T" BIT
TRTVEC= 14 ;:TRACE TRAP
BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24 ;:POWER FAIL
EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34 ;: "TRAP" TRAP
TKVEC= 60 ;:TTY KEYBOARD VECTOR
TPVEC= 64 ;:TTY PRINTER VECTOR
PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR

200
201
202 005746
203 005726
204 010046
205 012600
206 024646
207 022626
208
209
210
211

;; INSTRUCTION DEFINITIONS

PUSH1SP=5746 ;:DECREMENT PROCESSOR STACK 1 WORD
POP1SP=5726 ;:INCREMENT PROCESSOR STACK 1 WORD
PUSHR0=10046 ;:SAVE R0 ON STACK
POP R0=12600 ;:RESTORE R0 FROM STACK
PUSH2SP=24646 ;:DECREMENT STACK TWICE
POP2SP=22626 ;:INCREMENT STACK TWICE
.EQUIV EMT,HLT ;:BASIC DEFINITION OF ERROR CALL

TRAPCATCHER FOR UNEXPECTED INTERRUPTS

212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
(2)
(2)
257
258

000000 000000 000000
000020 004134
000022 000340
000024 007122
000026 000340
000030 006506
000032 000340
000034 006410
000036 000340
000040 000046
000046 004070
000052 000052
000052 000000
000052 000040
000174 000174
000174 000000
000176 000000
000200 000200
000200 000137 002402
001000 001000
001000 005200 041501 042455
001016 055103 041513 040507
177570
177570

```
*****  
-----  
:TRAPCATCAER FOR ILLEGAL INTERRUPTS  
:THE STANDARD 'TRAP CATCHER' IS PLACED  
:BETWEEN ADDRESS 0 TO ADDRESS 776.  
:IT LOOKS LIKE 'PC+2 HALT'.  
-----  
*****  
.=0  
      .WORD 0,0  
:STANDARD INTERRUPT VECTORS  
-----  
.=20  
      $SCOPE      ; SCOPE LOOP HANDLER.  
      PR7         ; SERVICE AT LEVEL 7.  
      $PWRDN      ; POWER FAIL HANDLER  
      PR7         ; SERVICE AT LEVEL 7  
      $ERROR      ; ERROR HANDLER  
      PR7         ; SERVICE AT LEVEL 7  
      $STRAP      ; GENERAL HANDLER DISPATCH SERVICE  
      PR7         ; SERVICE AT LEVEL 7  
      .SBTTL ACT11 HOOKS  
:*****  
:HOOKS REQUIRED BY ACT11  
      $SVPC=.      ; SAVE PC  
      .=46  
      $ENDAD      ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP  
      .=52  
      .WORD 0      ;:2)SET LOC.52 TO ZERO  
      .=$SVPC     ;: RESTORE PC  
      .=174  
      DISPREG:0   ; SOFTWARE DISPLAY REGISTER  
      SWREG: 0    ; SOFTWARE SWITCH REGISTER  
      .=200  
      JMP .START  ; GO TO START OF PROGRAM  
      .=1000  
      MTITLE: .ASCII <200><12>/AC-E107A-MC/<200>  
              .ASCIIZ /CZKCGA0 KMC FREE RUNNING TEST/<200>  
      DSWR = 177570  
      DDISP = 177570
```


COMMON TAGS

259
260
261
262
263
264
265 001200
266 001200 000000
267 001200 000000
268 001202 000
269 001203 000
270 001204 000000
271 001206 000000
272 001210 000000
273 001212 000000
274 001214 000
275 001215 001
276 001216 000000
277 001220 000000
278 001222 000000
279 001224 000000
280 001226 000000
281 001230 000000
282 001232 000000
283 001234 000
284 001235 000
285 001236 000000
286 001240 177570
287 001242 177570
288 001244 177560
289 001246 177562
290 001250 177564
291 001252 177566
292 001254 000
293 001255 002
294 001256 012
295 001257 000
296 001260 000000
297
298 001262 000000
299 001264 000000
300 001266 000000
301 001270 000000
302 001272 000000
303 001274 000000
304 001276 000000
305 001300 000000
306 001302 000000
307 001304 000000
308 001306 000000
309 001310 000000
310 001312 077
311 001313 015
312 001314 000012
313
314

.SBTTL COMMON TAGS

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

.=1200

\$CMTAG:

.WORD 0
\$TSTNM: .BYTE 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
 .WORD 0
 .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
 .WORD 0
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$REGAD: .WORD 0
\$REG0: .WORD 0
\$REG1: .WORD 0
\$REG2: .WORD 0
\$REG3: .WORD 0
\$REG4: .WORD 0
\$REG5: .WORD 0
\$TMP0: .WORD 0
\$TMP1: .WORD 0
\$TMP2: .WORD 0
\$TMP3: .WORD 0
\$TMP4: .WORD 0
\$TIMES: 0
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCII <12>

:::START OF COMMON TAGS

:::CONTAINS THE TEST NUMBER
:::CONTAINS ERROR FLAG
:::CONTAINS SUBTEST ITERATION COUNT
:::CONTAINS SCOPE LOOP ADDRESS
:::CONTAINS SCOPE RETURN FOR ERRORS
:::CONTAINS TOTAL ERRORS DETECTED
:::CONTAINS ITEM CONTROL BYTE
:::CONTAINS MAX. ERRORS PER TEST
:::CONTAINS PC OF LAST ERROR INSTRUCTION
:::CONTAINS ADDRESS OF 'GOOD' DATA
:::CONTAINS ADDRESS OF 'BAD' DATA
:::CONTAINS 'GOOD' DATA
:::CONTAINS 'BAD' DATA
:::RESERVED--NOT TO BE USED

:::AUTOMATIC MODE INDICATOR
:::INTERRUPT MODE INDICATOR

:::ADDRESS OF SWITCH REGISTER
:::ADDRESS OF DISPLAY REGISTER
:::TTY KBD STATUS
:::TTY KBD BUFFER
:::TTY PRINTER STATUS REG. ADDRESS
:::TTY PRINTER BUFFER REG. ADDRESS
:::CONTAINS NULL CHARACTER FOR FILLS
:::CONTAINS # OF FILLER CHARACTERS REQUIRED
:::INSERT FILL CHARS. AFTER A 'LINE FEED'
:::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
:::CONTAINS THE ADDRESS FROM
:::WHICH (\$REG0) WAS OBTAINED
:::CONTAINS ((\$REGAD)+0)
:::CONTAINS ((\$REGAD)+2)
:::CONTAINS ((\$REGAD)+4)
:::CONTAINS ((\$REGAD)+6)
:::CONTAINS ((\$REGAD)+10)
:::CONTAINS ((\$REGAD)+12)
:::USER DEFINED
:::USER DEFINED
:::USER DEFINED
:::USER DEFINED
:::USER DEFINED
:::MAX. NUMBER OF ITERATIONS
:::QUESTION MARK
:::CARRIAGE RETURN
:::LINE FEED

.SBTTL APT MAILBOX-ETABLE

APT MAILBOX-ETABLE

315			*****		
316			.EVEN		
317			\$MAIL:	::APT MAILBOX	
318	001316		\$MSGTY: .WORD	AMSGTY	::MESSAGE TYPE CODE
319	001316	000000	\$FATAL: .WORD	AFATAL	::FATAL ERROR NUMBER
320	001320	000000	\$TESTN: .WORD	ATESTN	::TEST NUMBER
321	001322	000000	\$PASS: .WORD	APASS	::PASS COUNT
322	001324	000000	\$DEVCT: .WORD	ADEVCT	::DEVICE COUNT
323	001326	000000	\$UNIT: .WORD	AUNIT	::I/O UNIT NUMBER
324	001330	000000	\$MSGAD: .WORD	AMSGAD	::MESSAGE ADDRESS
325	001332	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
326	001334	000000	\$ETABLE:	::APT ENVIRONMENT TABLE	
327	001336		\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
328	001336	002	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
329	001337	000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
330	001340	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
331	001342	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE, OPTIONS
332	001344	000000	*		BITS 15-11=CPU TYPE
333			*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
334			*		11/70=06,PDQ=07,q=10
335			*		BIT 10=REAL TIME CLOCK
336			*		BIT 9=FLOATING POINT PROCESSOR
337			*		BIT 8=MEMORY MANAGEMENT
338			\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
339	001346	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
340	001347	000	*		MEM.TYPE BYTE -- (HIGH BYTE)
341			*		900 NSEC CORE=001
342			*		300 NSEC BIPOLAR=002
343			*		500 NSEC MOS=003
344			\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
345	001350	000000	*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABO
346			\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
347	001352	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
348	001353	000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
349	001354	000000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
350	001356	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
351	001357	000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
352	001360	000000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
353	001362	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
354	001363	000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
355	001364	000000	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
356	001366	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
357	001370	000000	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
358	001372	000000	\$DEVCM: .WORD	ADEVCM	::DEVICE MAP
359	001374	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
360	001376	000000	\$CDW2: .WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
361	001400	000000	\$DDW0: .WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
362	001402	000000	\$DDW1: .WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
363	001404	000000	\$DDW2: .WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
364	001406	000000	\$DDW3: .WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
365	001410	000000	\$DDW4: .WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
366	001412	000000	\$DDW5: .WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
367	001414	000000	\$DDW6: .WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
368	001416	000000	\$DDW7: .WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
369	001420	000000	\$DDW8: .WORD	ADDW8	::DEVICE DESCRIPTOR WORD#8
370	001422	000000			

APT MAILBOX-ETABLE

371 001424 000000
 372 001426 000000
 373 001430 000000
 374 001432 000000
 375 001434 000000
 376 001436 000000
 377 001440 000000
 378
 379
 380 001442
 381
 382
 383
 384
 385 001442 000000
 386 001444 000000
 387
 388
 389
 390 001446 000000
 391 001450 000000
 392 001452 000000
 393 001454 000000
 394 001456 000000
 395 001460 000000
 396 001462 000000
 397 001464 000001
 398 001466 000000
 399 001470 000001
 400 001472 000001
 401 001474 000001
 402 001476 000001
 403 001500 000000
 404
 405 001502 002072
 406 001504 002276
 407
 408
 409
 410 001506 000
 411 001510 001510
 412 001510 000
 413 001511 000
 414
 415

\$DDW9: .WORD ADDW9 ::DEVICE DESCRIPTOR WORD#9
 \$DDW10: .WORD ADDW10 ::DEVICE DESCRIPTOR WORD#10
 \$DDW11: .WORD ADDW11 ::DEVICE DESCRIPTOR WORD#11
 \$DDW12: .WORD ADDW12 ::DEVICE DESCRIPTOR WORD#12
 \$DDW13: .WORD ADDW13 ::DEVICE DESCRIPTOR WORD#13
 \$DDW14: .WORD ADDW14 ::DEVICE DESCRIPTOR WORD#14
 \$DDW15: .WORD ADDW15 ::DEVICE DESCRIPTOR WORD#15

\$ETEND:

PROGRAM CONTROL PARAMETERS

 NEXT: .WORD 0 ; ADDRESS OF NEXT TEST TO BE EXECUTED
 LOCK: .WORD 0 ; ADDRESS FOR LOCK CURRENT DATA

PROGRAM VARIABLES

 STRTSW: .WORD 0 ; SWITCHES AT START OF PROGRAM
 STAT: .WORD 0 ; KM STATUS WORD STORAGE
 CLKX: .WORD 0 ;
 MASKX: .WORD 0 ;
 SAVSP: .WORD 0 ; STACK POINTER STORAGE
 SAVPC: .WORD 0 ; PROGRAM COUNTER STORAGE
 ZERO: .WORD 0 ;
 ONE: .WORD 1 ;
 MEMLIM: .WORD 0 ; HIGHEST LOCATION FOR NPR'S
 KMACTV: .BLKW 1 ; KMC11 SELECTED ACTIVE
 KMNUM: .BLKW 1 ; OCTAL NUMBER OF KMC11'S
 SAVACT: .BLKW 1 ; ORIGINAL ACTIVE DEVICES.
 SAVNUM: .BLKW 1 ; WORKABLE NUMBER.
 RUN: .WORD 0 ; POINTER TO RUNNING DEVICES
 .EVEN
 CREAM: .WORD KM.MAP-6 ; TABLE POINTER
 MILK: .WORD CNT.MAP-4 ; TABLE POINTER

PROGRAM CONTROL FLAGS

 INIFLG: .BYTE 0 ; PROGRAM INITIALIZING FLAG
 .EVEN
 LOKFLG: .BYTE 0 ; LOCK ON CURRENT TEST FLAG
 QV.FLG: .BYTE 0 ; QUICK VERIFY FLAG
 .EVEN ; ON FIRST PASS OF EACH KMC11 ITERATIONS WILL BE

ERROR POINTER TABLE

416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430 001512
 431
 432
 433 001512 000000
 434 001514 000000
 435 001516 000000
 436 001520 034020
 437 001522 034263
 438 001524 000000
 439 001526 033654
 440 001530 034167
 441 001532 034526
 442 001534 033677
 443 001536 000000
 444 001540 000000
 445 001542 033725
 446 001544 000000
 447 001546 000000
 448 001550 033747
 449 001552 000000
 450 001554 000000
 451 001556 033725
 452 001560 034167
 453 001562 034366
 454 001564 033747
 455 001566 034167
 456 001570 034354
 457 001572 000000
 458 001574 034135
 459 001576 034400
 460 001600 000000
 461 001602 034135
 462 001604 034416
 463 001606 000000
 464 001610 034167
 465 001612 034354
 466 001614 033774
 467 001616 034135
 468 001620 034434
 469 001622 034020
 470 001624 000000
 471 001626 000000

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

.EVEN
 ;* DF ;; DOES NOT APPLY IN THIS DIAGNOSTIC.

0
 0
 0
 EM12
 DH5 ;ERROR 1
 0
 EM2
 DH2 ;ERROR 2
 DT13
 EM3
 0 ;ERROR 3
 0
 EM4
 0 ;ERROR 4
 0
 EM5
 0 ;ERROR 5
 0
 EM4
 DH2 ;ERROR 6
 DT5
 EM5
 DH2 ;ERROR 7
 DT4
 0
 DH1 ;ERROR 10
 DT6
 0
 DH1 ;ERROR 11
 DT7
 0
 DH2 ;ERROR 12
 DT4
 EM11
 DH1 ;ERROR 13
 DT10
 EM12
 0 ;ERROR 14
 0

ERROR POINTER TABLE

472 001630 034020
473 001632 034167
474 001634 034366
475 001636 034044
476 001640 034210
477 001642 034452
478 001644 034067
479 001646 034231
480 001650 034464
481 002034
482
483
484
485
486
487 002034
488 000024 000024
489 000024 000200
490 000044 000044
491 000044 002034
492 002034
493
494
495
496
497 002034
498 002034 000000
499 002036 001316
500 002040 000132
501 002042 000137
502 002044 000137
503 002046 000052
504

EM12
DH2 ;ERROR 15
DT5
EM13
DH3 ;ERROR 16
DT11
EM14
DH4 ;ERROR 17
DT12

. =2034
.SBTTL APT PARAMETER BLOCK

::*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
. \$X= . ::SAVE CURRENT LOCATION
. =24 ::SET POWER FAIL TO POINT TO START OF PROGRAM
200 ::FOR APT START UP
. =44 ::POINT TO APT INDIRECT ADDRESS PNTR.
\$APTHDR ::POINT TO APT HEADER BLOCK
. =.\$X ::RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.

\$APTHD:
\$HIBTS: .WORD 0 ::TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
\$MBADR: .WORD \$MAIL ::ADDRESS OF APT MAILBOX (BITS 0-15)
\$TSTM: .WORD 90. ::RUN TIM OF LONGEST TEST
\$PASTM: .WORD 95. ::RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 95. ::ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITION
.WORD \$ETEND-\$MAIL/2 ::LENGTH MAILBOX-ETABLE(WORDS)

APT PARAMETER BLOCK

```

505
506           ;KMC11 CONTROL INDICATORS FOR CURRENT KMC11 UNDER TEST
507           ;-----
508
509 002050 000000  STAT1: 0
510 002052 000000  STAT2: 0
511 002054 000000  STAT3: 0
512
513           ;KMC11 VECTOR AND REGISTER INDIRECT POINTERS
514           ;-----
515
516 002056 000000  KMRVEC: 0           ;POINTER TO KMC11 RECEIVER INTERRUPT VECTOR
517 002060 000000  KMRLVL: 0          ;POINTER TO KMC11 RECEIVER INTERRUPT SERVICE PS
518 002062 000000  KMTVEC: 0          ;POINTER TO KMC11 TRANSMITTER INTERRUPT VECTOR
519 002064 000000  KMTLVL: 0         ;POINTER TO KMC11 TRANSMITTER INTERRUPT SERVICE PS
520 002066 000000  KMCSR: 0           ;POINTER TO KMC11 CONTROL STATUS REGISTER
521 002070 000000  KMCSRH: 0          ;POINTER TO KMC11 CONTROL STATUS REGISTER HIGH BYTE.
522 002072 000000  KMCTL: 0           ;POINTER TO KMC11 CONTROL OUT REGISTER
523 002074 000000  KMP04: 0          ;POINTER TO KMC11 PORT REGISTER(SEL 4)
524 002076 000000  KMP06: 0          ;POINTER TO KMC11 PORT REGISTER(SEL 6)
525
526           ;TEMP STORAGE
527           ;-----
528
529           ;TEMP: 0
530           ;.=.+40
531
532           ;KMC11 STATUS TABLE AND ADDRESS ASSIGNMENTS
533           ;-----
534
535           .=2100
536 002100 002100  KM.MAP:
537 002100 000001  KMCRO0: .BLKW 1           ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 00
538 002102 000001  KMS100: .BLKW 1          ;VECTOR FOR KMC11 NUMBER 00
539 002104 000001  KMS200: .BLKW 1          ;DDCMP LINE# FOR KMC11 NUMBER 00
540 002106 000001  KMS300: .BLKW 1          ;3RD STATUS WORD
541
542 002110 000001  KMCRO1: .BLKW 1           ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 01
543 002112 000001  KMS101: .BLKW 1          ;VECTOR FOR KMC11 NUMBER 01
544 002114 000001  KMS201: .BLKW 1          ;DDCMP LINE# FOR KMC11 NUMBER 01
545 002116 000001  KMS301: .BLKW 1          ;3RD STATUS WORD
546
547 002120 000001  KMCRO2: .BLKW 1           ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 02
548 002122 000001  KMS102: .BLKW 1          ;VECTOR FOR KMC11 NUMBER 02
549 002124 000001  KMS202: .BLKW 1          ;DDCMP LINE# FOR KMC11 NUMBER 02
550 002126 000001  KMS302: .BLKW 1          ;3RD STATUS WORD
551
552 002130 000001  KMCRO3: .BLKW 1           ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 03
553 002132 000001  KMS103: .BLKW 1          ;VECTOR FOR KMC11 NUMBER 03
554 002134 000001  KMS203: .BLKW 1          ;DDCMP LINE# FOR KMC11 NUMBER 03
555 002136 000001  KMS303: .BLKW 1          ;3RD STATUS WORD
556
557 002140 000001  KMCRO4: .BLKW 1           ;CONTROL STATUS REGISTER FOR KMC11 NUMBER 04
558 002142 000001  KMS104: .BLKW 1          ;VECTOR FOR KMC11 NUMBER 04
559 002144 000001  KMS204: .BLKW 1          ;DDCMP LINE# FOR KMC11 NUMBER 04
560 002146 000001  KMS304: .BLKW 1          ;3RD STATUS WORD

```


APT PARAMETER BLOCK

561					
562	002150	000001	KMCR05: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 05
563	002152	000001	KMS105: .BLKW	1	:VECTOR FOR KMC11 NUMBER 05
564	002154	000001	KMS205: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 05
565	002156	000001	KMS305: .BLKW	1	:3RD STATUS WORD
566					
567	002160	000001	KMCR06: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 06
568	002162	000001	KMS106: .BLKW	1	:VECTOR FOR KMC11 NUMBER 06
569	002164	000001	KMS206: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 06
570	002166	000001	KMS306: .BLKW	1	:3RD STATUS WORD
571					
572	002170	000001	KMCR07: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 07
573	002172	000001	KMS107: .BLKW	1	:VECTOR FOR KMC11 NUMBER 07
574	002174	000001	KMS207: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 07
575	002176	000001	KMS307: .BLKW	1	:3RD STATUS WORD
576					
577	002200	000001	KMCR10: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 10
578	002202	000001	KMS110: .BLKW	1	:VECTOR FOR KMC11 NUMBER 10
579	002204	000001	KMS210: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 10
580	002206	000001	KMS310: .BLKW	1	:3RD STATUS WORD
581					
582	002210	000001	KMCR11: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 11
583	002212	000001	KMS111: .BLKW	1	:VECTOR FOR KMC11 NUMBER 11
584	002214	000001	KMS211: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 11
585	002216	000001	KMS311: .BLKW	1	:3RD STATUS WORD
586					
587	002220	000001	KMCR12: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 12
588	002222	000001	KMS112: .BLKW	1	:VECTOR FOR KMC11 NUMBER 12
589	002224	000001	KMS212: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 12
590	002226	000001	KMS312: .BLKW	1	:3RD STATUS WORD
591					
592	002230	000001	KMCR13: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 13
593	002232	000001	KMS113: .BLKW	1	:VECTOR FOR KMC11 NUMBER 13
594	002234	000001	KMS213: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 13
595	002236	000001	KMS313: .BLKW	1	:3RD STATUS WORD
596					
597	002240	000001	KMCR14: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 14
598	002242	000001	KMS114: .BLKW	1	:VECTOR FOR KMC11 NUMBER 14
599	002244	000001	KMS214: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 14
600	002246	000001	KMS314: .BLKW	1	:3RD STATUS WORD
601					
602	002250	000001	KMCR15: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 15
603	002252	000001	KMS115: .BLKW	1	:VECTOR FOR KMC11 NUMBER 15
604	002254	000001	KMS215: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 15
605	002256	000001	KMS315: .BLKW	1	:3RD STATUS WORD
606					
607	002260	000001	KMCR16: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 16
608	002262	000001	KMS116: .BLKW	1	:VECTOR FOR KMC11 NUMBER 16
609	002264	000001	KMS216: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 16
610	002266	000001	KMS316: .BLKW	1	:3RD STATUS WORD
611					
612	002270	000001	KMCR17: .BLKW	1	:CONTROL STATUS REGISTER FOR KMC11 NUMBER 17
613	002272	000001	KMS117: .BLKW	1	:VECTOR FOR KMC11 NUMBER 17
614	002274	000001	KMS217: .BLKW	1	:DDCMP LINE# FOR KMC11 NUMBER 17
615	002276	000001	KMS317: .BLKW	1	:3RD STATUS WORD
616					

08-JUN-78 07:54 PAGE 14
CZKCGA.P11 08-JUN-78 07:53

N 2

PAGE: 0026CZ

APT PARAMETER BLOCK

617 002300 000000

KM.END: 000000

APT PARAMETER BLOCK

618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670

002302
002302 000000
002304 000000
002306 000000
002310 000000
002312 000000
002314 000000
002316 000000
002320 000000
002322 000000
002324 000000
002326 000000
002330 000000
002332 000000
002334 000000
002336 000000
002340 000000
002342 000000
002344 000000
002346 000000
002350 000000
002352 000000
002354 000000
002356 000000
002360 000000
002362 000000
002364 000000
002366 000000
002370 000000
002372 000000
002374 000000
002376 000000
002400 000000

;KMC11 PASS COUNT AND ERROR COUNT TABLE

CNT.MAP:
PACT00: 0
ERCT00: 0
PACT01: 0
ERCT01: 0
PACT02: 0
ERCT02: 0
PACT03: 0
ERCT03: 0
PACT04: 0
ERCT04: 0
PACT05: 0
ERCT05: 0
PACT06: 0
ERCT06: 0
PACT07: 0
ERCT07: 0
PACT10: 0
ERCT10: 0
PACT11: 0
ERCT11: 0
PACT12: 0
ERCT12: 0
PACT13: 0
ERCT13: 0
PACT14: 0
ERCT14: 0
PACT15: 0
ERCT15: 0
PACT16: 0
ERCT16: 0
PACT17: 0
ERCT17: 0

;PASS COUNT FOR KMC11 NUMBER 00
;ERROR COUNT FOR KMC11 NUMBER 00
;PASS COUNT FOR KMC11 NUMBER 01
;ERROR COUNT FOR KMC11 NUMBER 01
;PASS COUNT FOR KMC11 NUMBER 02
;ERROR COUNT FOR KMC11 NUMBER 02
;PASS COUNT FOR KMC11 NUMBER 03
;ERROR COUNT FOR KMC11 NUMBER 03
;PASS COUNT FOR KMC11 NUMBER 04
;ERROR COUNT FOR KMC11 NUMBER 04
;PASS COUNT FOR KMC11 NUMBER 05
;ERROR COUNT FOR KMC11 NUMBER 05
;PASS COUNT FOR KMC11 NUMBER 06
;ERROR COUNT FOR KMC11 NUMBER 06
;PASS COUNT FOR KMC11 NUMBER 07
;ERROR COUNT FOR KMC11 NUMBER 07
;PASS COUNT FOR KMC11 NUMBER 10
;ERROR COUNT FOR KMC11 NUMBER 10
;PASS COUNT FOR KMC11 NUMBER 11
;ERROR COUNT FOR KMC11 NUMBER 11
;PASS COUNT FOR KMC11 NUMBER 12
;ERROR COUNT FOR KMC11 NUMBER 12
;PASS COUNT FOR KMC11 NUMBER 13
;ERROR COUNT FOR KMC11 NUMBER 13
;PASS COUNT FOR KMC11 NUMBER 14
;ERROR COUNT FOR KMC11 NUMBER 14
;PASS COUNT FOR KMC11 NUMBER 15
;ERROR COUNT FOR KMC11 NUMBER 15
;PASS COUNT FOR KMC11 NUMBER 16
;ERROR COUNT FOR KMC11 NUMBER 16
;PASS COUNT FOR KMC11 NUMBER 17
;ERROR COUNT FOR KMC11 NUMBER 17

APT PARAMETER BLOCK

671
 672
 673
 674
 675
 676

FORMAT OF STATUS TABLE

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	CSR
I	C	O	N	T	R	O	L	R	E	G	I	S	T	E	R	I	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	STAT1
I	*	I	*	I	*	I	*	I	*	I	*	I	*	I	*	I	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	STAT2
I	*	B	M	A	D	D	*	I	*	L	I	N	E	#	*	I	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	STAT3
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	

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 AND START UP.

```

725
726      ;PROGRAM INITIALIZATION
727      ;LOCK OUT INTERRUPTS
728      ;SET UP PROCESSOR STACK
729      ;SET UP POWER FAIL VECTOR
730      ;CLEAR PROGRAM CONTROL FLAGS AND COUNTS
731      ;TYPE TITLE MESSAGE
732
733      002402 012737 000340 177776      .START: MOV      #340,PS      ;LOCK OUT INTERRUPTS
734      002410 012706 001200              MOV      #STACK,SP      ;SET UP STACK
735      002414 012737 007122 000024      MOV      #SPWRDN,@#24    ;SET UP POWER FAIL VECTOR
736      002422 013737 001472 001476      MOV      KMNUM,SAVNUM    ;SAVE NUMBER OF DEVICES IN SYSTEM.
737      002430 005037 011432              CLR      SWFLG           ;CLEAR SOFT TIMEOUT FLAG
738      002434 105037 001203              CLRB    $ERFLG          ;CLEAR ERROR FLAG
739      002440 105037 001511              CLRB    QV.FLG          ;ZERO QUICK VERIFY FLAG
740      002444 012737 002070 001502      MOV      #KM.MAP-10,CREAM;GET MAP POINTER.
741      002452 012737 002276 001504      MOV      #CNT.MAP-4,MILK ;GET PASS COUNT MAP POINTER
742      002460 012737 100000 001500      MOV      #BIT15,RUN      ;POINT POINTER TO FIRST DEVICE.
743      002466 012700 002302              MOV      #CNT.MAP,R0     ;PASS COUNT POINTER TO R0
744      002472 005020 002402      23$:  CLR      (R0)+          ;CLEAR TABLE
745      002474 022700 002402              CMP      #CNT.MAP+100,R0 ;DONE YET?
746      002500 001374 001216              BNE     23$             ;KEEP GOING
747      002502 005037 001202              CLR      $ERRPC         ;CLEAR LAST ERROR POINTER
748      002506 012737 000001 001202      MOV      #1,$STSTNM     ;SET UP FOR TEST 1
749      002514 012737 002402 001206      MOV      #.START,$LPADR ;SET UP FOR POWER FAIL BEFORE
750                                  ;TESTING STARTS
751      002522 132737 000001 001336      BITB    #1,$ENV         ; IS IT RUNNING UNDER APT?
752      002530 001404 000176              BEQ     3$              ; IF NOT CHECK FOR TYPE OF SWITCH REGISTER.
753      002532 013737 001340 000176      MOV      $$SWREG,SWREG  ; LOAD SOFTWARE SWITCH REG.
754      002540 000423 000006              BR      6$+2           ; GO SET UP SOFTWARE SWITCH REG.
755      002542 013746 000004              3$:  MOV      @#6,-(SP)    ;SAVE CURRENT VECTORS
756      002546 013746 000004              MCV     @#4,-(SP)
757      002552 012737 002606 000004      MOV      #6$,@#4       ;SET UP FOR TIMEOUT
758      002560 012737 177570 001240      MOV      #177570,SWR    ;SET SWR TO HARD SWR ADDRESS
759      002566 012737 177570 001242      MOV      #177570,DISPLAY;SET DISPLAY TO HARD SWR ADDRESS
760      002574 022777 177777 176436      CMP      #-1,@SWR      ;REFERENCE HARDWARE SWITCH REGISTER
761      002602 001402 000407              BEQ     6$+2           ;IF = -1 USE SOFT SWR ANYWAY
762      002604 000407 000407              BR      7$              ;IF IT EXISTS AND NOT = -1 USE HARD SWR
763      002606 022626 000176 001240      6$:  CMP      (SP)+,(SP)+  ;ADJUST STACK
764      002610 012737 000174 001242      MOV      #SWREG,SWR    ;POINTER TO SOFT SWR
765      002616 012737 000004 001242      MOV      #DISPREG,DISPLAY;POINTER TO SOFT DISPLAY REG
766      002624 012637 000006              7$:  MOV      (SP)+,@#4     ;RESTORE VECTORS
767      002630 012637 000006              MOV      (SP)+,@#6
768      002634 105737 001506              TSTB    INIFLG         ;HAS INITIALIZATION BEEN PERFORMED
769      002640 001006 004070 000042      BNE     20$            ;BR IF YES
770      002642 022737 004070 000042      CMP      #SENDAD,@#42  ;IF ACT-11 AUTOMATIC MODE, DON'T TYPE ID
771      002650 001402 001000              BEQ     20$
772      002652 104401 011226              TYPE    ,MTITLE        ;TYPE TITLE MESSAGE
773      002656 004737 011226              JSR     PC,CKSWR       ;CHECK FOR SOFT SWR
774      002662 017737 176352 001446      MOV      @SWR,STRTSW   ;STORE STARTING SWITCHES
775      002670 005737 000042              TST     @#42           ;IS IT RUNNING IN AUTO MODE?
776      002674 001402 001446              BEQ     .+6            ;BR IF NO
777      002676 005037 001446              CLR     STRTSW         ;IF YES, CLEAR SWITCHES
778      002702 032737 000001 001446      BIT     #SW00,STRTSW   ;IF SW00=1, QUESTIONS ARE ASKED.
779      002710 001012 001446              BNE     17$           ;BR IF SW00=1
780      002712 105737 001446              TSTB    STRTSW        ;BIT7=1??
  
```


PROGRAM INITIALIZATION AND START UP.

```

781 002716 100007          BPL      17$          ;BR IF SW07=0
782 002720 005737 001470  TST      KMACTV      ;ARE ANY DEVICES SELECTED?
783 002724 001027          BNE      16$          ;BR IF YES
784 002726 104401 010725  TYPE,    NOACT        ;NO DEVICES SELECTED.
785 002732 000000          HALT                    ;STOP THE SHOW
786 002734 000776          BR        .-2          ;DISQUALIFY CONTINUE SWITCH
787 002736 105737 001336  17$:    TSTB     $ENV      ; IS IT UNDER APT DUMP MODE?
788 002742 001405          BEQ      27$          ; YES, CHECK IF APT SIZED IT?
789 002744 132737 000001 001336  BITB     #1,$ENV      ; IS IT UNDER Q,V OR RUN MODE?
790 002752 001012          BNE      30$          ; YES, NEEDS ONLY APT SIZING.
791 002754 000406          BR        33$          ; NO, NEEDS REGULAR AUTO.SIZE.
792 002756 105737 001337  27$:    TSTB     $ENVM    ; IS IT SIZED BY APT?
793 002762 100406          BMI      30$          ; YES, NEEDS ONLY APT SIZING.
794 002764 042737 000001 001446  BIC      #SW00,STRTSW ; SIZE ONLY IN AUTO MODE.
795 002772 004737 012124  33$:    JSR      PC,AUTO.SIZE ; GO TO THE AUTO.SIZE.
796 002776 000402          BR        16$          ; GO PRINT THE MAP.
797 003000 004737 013540  30$:    JSR      PC,APT.SIZE ; GO DO THE APT SIZING.
798 003004 105737 001506  16$:    TSTB     INIFLG    ;FIRST TIME?
799 003010 001410          BEQ      21$          ;BR IF YES
800 003012 105737 001446  TSTB     STRTSW      ;IF USING SAME PARAMETERS DONT TYPE MAP
801 003016 100431          BMI      1$          ;
802 003020 032737 000006 001446  BIT      #BIT1!BIT2,STRTSW; IS TEST NO. OR LOCK SELECTED
803 003026 001403          BEQ      24$          ;IF NO THEN TYPE STATUS
804 003030 000424          BR        1$          ;IF YES DO NOT TYPE STATUS
805 003032 105137 001506  21$:    COMB     INIFLG    ;SET FLAG
806 003036 104401 010073  24$:    TYPE     ,XHEAD   ;TYPE HEADER
807 003042 012704 002100          MOV      #KM.MAP,R4  ;SET POINTER
808 003046 010437 001276  5$:    MOV      R4,$TMP0  ;SET ADDRESS
809 003052 012437 001300          MOV      (R4)+,$TMP1 ;SET CSR
810 003056 001411          BEQ      1$          ;ALL DONE IF ZERO
811 003060 012437 001302          MOV      (R4)+,$TMP2 ;SET STAT1
812 003064 012437 001304          MCV      (R4)+,$TMP3 ;SET STAT2
813 003070 012437 001306          MOV      (R4)+,$TMP4 ;SET STAT3
814 003074 104416          CONVRT                    ;TYPE OUT STATUS MAP
815 003076 011074          XSTATQ                    ;
816 003100 000762          BR        5$          ;
817 003102 012700 002100  1$:    MOV      #KM.MAP,R0 ;R0 POINTS TO STATUS TABLE
818
819
820
821
822
823
824
825
826
827
828
829 003106 013746 000004          MOV      @#4,-(SP)    ;SAVE LOC 4
830 003112 013746 000006          MOV      @#6,-(SP)    ;SAVE LOC 6
831 003116 005037 000006          CLR      @#6          ;CLEAR VEC+2
832 003122 005037 001302          CLR      $TMP2       ;CLEAR FLAG
833 003126 011037 002066  AUSTRT: MOV      (R0),KMCSR ;GET NEXT KMC CSR
834 003132 001510          BEQ      AUDONE      ;BR IF DONE
835 003134 012737 003240 000004  2$:    MOV      #NODEV,@#4 ;SET UP FOR TIMEOUT
836 003142 012703 000010  3$:    MOV      #10,R3    ;R3 IS COUNT OF DEVICES BEFORE KMC

```

```

*****
;*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.
*****

```


PROGRAM INITIALIZATION AND START UP.

837	003146	012702	003342		4\$:	MOV	#DEVTAB,R2		:R2 IS DEVICE TABLE PONTER
838	003152	012701	160010			MOV	#160010,R1		:START WITH ADDRESS 160010
839	003156	005711			FLOAT:	TST	(R1)		:CHECK ADDRESS IN R1
840	003160	111204				MOVB	(R2),R4		:IF NO TIMEOUT, GET NEXT ADDRESS
841	003162	060401				ADD	R4,R1		:IN R1
842	003164	005201				INC	R1		:
843	003166	040401				BIC	R4,R1		:
844	003170	005703				TST	R3		:ANY MORE DEVICES TO CHECK FOR?
845	003172	001371				BNE	FLOAT		:BR IF YES
846	003174	012737	003244	000004		MOV	#ERR,@#4		:OK ONLY KMC'S ARE LEFT, SET UP FOR TIMEOUT
847	003202	005711			FY:	TST	(R1)		:CHECK KMC ADDRESS
848	003204	020137	002066			CMP	R1,KMCSR		:DOES IT MATCH
849	003210	001403				BEQ	OK		:BR IF YES
850	003212	062701	000010			ADD	#10,R1		:GET NEXT KMC ADDRESS
851	003216	000771				BR	FY		:DO IT AGAIN
852	003220	062700	000010		OK:	ADD	#10,R0		:SKIP TO NEXT KMC CSR
853	003224	062701	000010			ADD	#10,R1		:GET NEXT KMC ADDRESS
854	003230	011037	002066			MOV	(R0),KMCSR		:GET NEXT KMC CSR
855	003234	001447				BEQ	AUDONE		:BRANCH IF ALL DONE.
856	003236	000761				BR	FY		:DO IT AGAIN.
857	003240	122243			NODEV:	CMPB	(R2)+,-(R3)		:ON TIMEOUT, INC R2, DEC R3
858	003242	000002				RTI			:\$LPADR
859	003244	005737	001302		ERR:	TST	\$TMP2		:CHECK FLAG IF = 0 TYPE HEADER
860	003250	001014				BNE	1\$:SKIP HEADER
861	003252	104401				TYPE			:TYPEOUT HEADER MESSAGE
862	003254	010756				CONERR			:CONFIGURATION ERROR!!!!
863	003256	012737	003244	001460		MOV	#ERR,SAVPC		:SAVE PC FOR TYPEOUT
864	003264	104417				CNVRT			:TYPE OUT ERROR PC
865	003266	003322				ERRPC			:
866	003270	104401				TYPE			:TYPE REST OF HEADER
867	003272	011023				CNERR			:
868	003274	012737	177777	001302		MCV	#-1,\$TMP2		:SET FLAG SO IT ONLY GETS TYPED ONCE
869	003302	010137	001264		1\$:	MOV	R1,\$REG1		:SAVE R1 FOR TYPEOUT
870	003306	104416				CONVRT			:
871	003310	003330				CONTAB			:TYPE CSR VALUES
872	003312	104401			3\$:	TYPE			:
873	003314	011044				KMCM			:
874	003316	022626			4\$:	CMP	(SP)+,(SP)+		:ADJUST STACK
875	003320	000737				BR	OK		:BR TO GET OUT
876	003322	000001			ERRPC:	1			:
877	003324	006	002			.BYTE	6,2		:
878	003326	001460				SAVPC			:
879	003330	000002			CONTAB:	2			:
880	003332	006	004			.BYTE	6,4		:
881	003334	001264				\$REG1			:
882	003336	006	002			.BYTE	6,2		:
883	003340	002066				KMCSR			:
884	003342	007			DEVTAB:	.BYTE	7		:DJ
885	003343	017				.BYTE	17		:DH
886	003344	007				.BYTE	7		:DQ
887	003345	007				.BYTE	7		:DU
888	003346	007				.BYTE	7		:DUP
889	003347	007				.BYTE	7		:LK
890	003350	007				.BYTE	7		:DMC
891	003351	007				.BYTE	7		:DZ
892	0033_2	007				.BYTE	7		:KMC

PROGRAM INITIALIZATION AND START UP.

893		003354			.EVEN			
894	003354				AUDONE:			
895	003354	012637	000006		1\$:	MOV	(SP)+, @#6	:RESTORE LOC 6
896	003360	012637	000004			MOV	(SP)+, @#4	:RESTORE LOC 4
897	003364	032737	000010	0C1446		BIT	#SW03, STRTSW	:SELECT SPECIFIC DEVICES??
898	003372	001422				BEQ	3\$:BR IF NO.
899	003374	104401	010013			TYPE	, MNEW	:TYPE THE MESSAGE.
900	003400	005000				CLR	R0	:ZERO DATA LIGHTS
901	003402	000000				HALT		:WAIT FOR USER TO TELL WHAT DEVICES TO RUN
902	003404	027737	175630	001474		CMP	@SWR, SAVACT	:IS THE NUMBER VALID?
903	003412	101404				BLOS	2\$:BR IF NUMBER IS OK.
904	003414	104401	007666			TYPE	, MERR3	:TELL USER OF INVALID NUMBER.
905	003420	000000				HALT		:STOP EVERY THING.
906	003422	000776				BR	.-2	:RESTART THE PROGRAM AGAIN.
907	003424	017737	175610	001470	2\$:	MOV	@SWR, KMACTV	:GET NEW DEVICE PATTERN
908	003432	013700	001470			MOV	KMACTV, R0	:SHOW THE USER WHAT HE SELECTED.
909	003436	000000				HALT		:CONTINUE DYNAMIC SWITCHES.
910	003440	012700	000300		3\$:	MOV	#300, R0	:PREPARE TO CLEAR THE FLOATING
911	003444	012701	000302			MOV	#302, R1	:VECTOR AREA. 300-776
912	003450	010120			4\$:	MOV	R1, (R0)+	:START PUTTING 'PC+2 - HALT'
913	003452	005021				CLR	(R1)+	:IN VECTOR AREA.
914	003454	022021				CMP	(R0)+, (R1)+	:POP POINTERS
915	003456	022700	001000			CMP	#1000, R0	:ALL DONE??
916	003462	001372				BNE	4\$:BR IF NO.
917								
918								
919								
920								
921	003464	012706	001200		.BEGIN:	MOV	#STACK, SP	:SET UP STACK
922	003470	013746	000006			MOV	@#6, -(SP)	:SAVE LOC 6
923	003474	013746	000004			MOV	@#4, -(SP)	:SAVE LOC 4
924	003500	005000				CLR	R0	:START AT 0
925	003502	012737	003546	000004		MOV	#2\$, @#4	:SET UP FOR TIME OUT
926	003510	005037	000006			CLR	@#6	:TO AUTOSIZE MEMORY
927	003514	005720			6\$:	TST	(R0)+	:CHECK ADDRESS IN R0
928	003516	022700	157776			CMP	#157776, R0	:IS IT AT LEAST 28K
929	003522	001374				BNE	6\$:BR IF NO
930	003524	162700	007776			SUB	#7776, R0	:SAVE 2K FOR MONITORS
931	003530	010037	001466		7\$:	MOV	R0, MEMLIM	:STORE MEMORY LIMIT
932	003534	012637	000004			MOV	(SP)+, @#4	:RESTORE LOC 4
933	003540	012637	000006			MOV	(SP)+, @#6	:RESTORE LOC 6
934	003544	000413				BR	10\$:CONTINUE
935	003546	022626			2\$:	CMP	(SP)+, (SP)+	:ADJUST STACK
936	003550	162700	000004			SUB	#4, R0	:GET LAST GOOD ADDRESS
937	003554	162700	007776			SUB	#7776, R0	:SAVE 2K FOR MONITORS
938	003560	022700	030000			CMP	#30000, R0	:IS IT 8K?
939	003564	001361				BNE	7\$:BR IF NO
940	003566	012700	037400			MOV	#37400, R0	:IF 8K DON'T SAVE 2K
941	003572	000756				BR	7\$:
942	003574	012737	000340	177776	10\$:	MOV	#340, PS	:LOCK OUT INTERRUPTS
943	003602	032737	000004	001446		BIT	#BIT2, STRTSW	:CHECK FOR LOCK ON TEST
944	003610	001406				BEQ	1\$:BR IF NO LOCK DESIRED.
945	003612	104401	007712			TYPE	, MLOCK	:TYPE LOCK SELECTED.
946	003616	012737	000240	004146		MOV	#NOP, TTST	:SET UP TO LOCK
947	003624	000403				BR	3\$:CONTINUE ALONG.
948	003626	013737	004354	004146	1\$:	MOV	BRW, TTST	:PREPARE NORMAL SCOPE ROUTINE

:TEST START AND RESTART

PROGRAM INITIALIZATION AND START UP.

949	003634	012737	011474	001206	3\$:	MOV	#CYCLE,\$LPADR	:START AT 'CYCLE' FIND WHICH DEVICE TO TEST
950	003642	032737	000002	001446	4\$:	BIT	#SW01,STRTSW	:IS TEST NO. SELECTED?
951	003650	001002				BNE	5\$:BR IF YES
952	003652	104401	007636			TYPE	,MR	:TYPE R
953	003656	000177	175324		5\$:	JMP	@\$LPADR	:START TESTING

END OF PASS ROUTINE

```

:END OF PASS
:TYPE NAME OF TEST
:UPDATE PASS COUNT
:CHECK FOR EXIT TO ACT-11
:RESTART TEST
    
```

.SBTTL END OF PASS ROUTINE

```

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

\$EOP:

```

954
955
956
957
958
959
960
961
962
963
964
965
966
967 003662
968 003662 000005
969 003664 005237 001324
970 003670 105037 001203
971 003674 104401 007614
972 003700 104401 007741
973 003704 104417 004104
974 003710 104401 007747
975 003714 104417 004112
976 003720 104401 007755
977 003724 104417 004120
978 003730 104401 007766
979 003734 104417 004126
980 003740 013700 001504
981 003744 013720 001324
982 003750 013720 001212
983 003754 013777 002060 176074
984 003762 005077 176072
985 003766 013777 002064 176066
986 003774 005077 176064
987 004000 005337 001476
988 004004 001035
989 004006 112737 000377 001511
990 004014 013737 001472 001476
991 004022 005037 001216
992 004026 005037 001310
993 004032 005237 001324
994 004036 042737 100000 001324
995 004044 005327
996 004046 000001
997 004050 003013
998 004052 012737
999 004054 000001
1000 004056 004046
1001 004060 013700 000042
1002 004064 001405
1003 004066 000005
1004 004070 004710
1005 004072 000240
1006 004074 000240
1007 004076 000240
1008 004100
1009 004100 000137
    
```

```

RESET
INC $PASS ; INCREMENT THE PASS COUNT
CLRB $ERFLG ; CLEAR ERROR FLAG
TYPE ,MEPASS ; TYPE END PASS.
TYPE ,MCSRX ; TYPE "CSR"
CNVRT ,XCSR ; SHOW IT.
TYPE ,MVECX ; TYPE VECTOR.
CNVRT ,XVEC ; SHOW IT.
TYPE ,MPASSX ; TYPE " PASSES "
CNVRT ,XPASS ; SHOW IT.
TYPE ,MERRX ; TYPE " ERRORS "
CNVRT ,XERR ; SHOW IT.
MOV MILK,RO ; SET POINTER TO PASSCNT.
MOV $PASS,(RO)+ ; SAVE THE PASS COUNT.
MOV $ERTTL,(RO)+ ; SAVE ERROR COUNT
MOV KMRLVL,@KMRVEC ; RESTORE THE RECEIVER INTERRUPT VECTOR.
CLR @KMRLVL ; RESTORE RECEIVER LEVEL
MOV KMTLVL,@KMTVEC ; RESTORE THE TRANSMIT INTERRUPT VECTOR.
CLR @KMTLVL ; RESTORE TRANSMITTER LEVEL
DEC SAVNUM ; ALL DEVICE TESTED?
BNE $DOAGN ; BRANCH IF NO.
MOVB #377,QV.FLG ; SET QUICK VERIFY FLAG.
MOV KMMNUM,SAVNUM ; RESTORE DEVICE COUNT.
CLR $ERRPC ; CLEAR LAST ERROR PC
CLR $TIMES ; ZERO THE NUMBER OF ITERATIONS
INC $PASS ; INCREMENT THE PASS NUMBER
BIC #100000,$PASS ; DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ; LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ; YES
MOV (PC)+,@(PC)+ ; RESTORE COUNTER
$ENDCT: .WORD 1
$EOPCT
$GET42: MOV @#42,RO ; GET MONITOR ADDRESS
BEQ $DOAGN ; BRANCH IF NO MONITOR
RESET ; CLEAR THE WORLD
$ENDAD: JSR PC,(RO) ; GO TO MONITOR
NOP ; SAVE ROOM
NOP ; FOR
NOP ; ACT11
$DOAGN: JMP @ (PC)+ ; RETURN
    
```


END OF PASS ROUTINE

1010	004102	011474		
1011	004104	000001		
1012	004106	006	002	
1013	004110	002066		
1014	004112	000001		
1015	004114	004	002	
1016	004116	002056		
1017	004120	000001		
1018	004122	006	002	
1019	004124	001324		
1020	004126	000001		
1021	004130	006	002	
1022	004132	001212		
1023				
1024				
1025				
1026				
1027				
1028				
1029				
1030				
1031				
1032				
1033				
1034				
1035				
1036				
1037				
1038				
1039	004134			
1040	004134	005037	001216	
1041	004140	023716	013764	
1042	004144	001413		
1043	004146	000406		
1044	004150	105777	175070	
1045	004154	100065		
1046	004156	017766	175064	177776
1047	004164	032777	040000	175046
1048	004172	001056		
1049				
1050	004174	000416		
1051				
1052	004176	013746	000004	
1053	004202	012737	004222	000004
1054	004210	005737	177060	
1055	004214	012637	000004	
1056	004220	000436		
1057	004222	022626		
1058	004224	012637	000004	
1059	004230	000437		
1060	004232			
1061	004232	105737	001203	
1062	004236	001404		
1063	004240	105037	001203	
1064	004244	005037	001310	
1065	004250	032777	004000	174762

```

$RTNAD: .WORD   CYCLE
XCSR:   1
        .BYTE   6,2
        KMCSR
XVEC:   1
        .BYTE   4,2
        KMRVEC
XPASS:  1
        .BYTE   6,2
        $PASS
XERR:   1
        .BYTE   6,2
        $ERTTL

;SCOPE LOOP AND INTERATION HANDLER
;-----

.SBTTL  SCOPE HANDLER ROUTINE

;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW14=1      LOOP ON TEST
;*SW11=1      INHIBIT ITERATIONS
;*CALL
;*          SCOPE          ;;SCOPE=IOT

$SCOPE:
        CLR     $ERRPC          ; CLEAR LAST ERROR PC
        CMP     TST1+2,(SP)     ; IS THIS TEST #1 ?
        BEQ     $XTSTR         ; IF SO DON'T LOOP.
TTST:   BR      1$
        TSTB   @STKB          ; KEYBOARD DONE ?
        BPL     $OVER          ; IF NO DONT WAIT.
        MOV    @STKB,-2(SP)
1$:     BIT    #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
        BNE    $OVER          ;;YES IF SW14=1
;#####START OF CODE FOR THE XOR TESTER#####
$XTSTR: BR      6$
        MOV    @ERRVEC,-(SP)   ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
        MOV    #55,@ERRVEC    ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
        TST   @#177060        ;;SAVE THE CONTENTS OF THE ERROR VECTOR
        MOV    (SP)+,@ERRVEC   ;;SET FOR TIMEOUT
        BR     $SVLAD         ;;TIME OUT ON XOR?
5$:     CMP    (SP)+,(SP)+     ;;RESTORE THE ERROR VECTOR
        MOV    (SP)+,@ERRVEC   ;;GO TO THE NEXT TEST
        BR     $OVER          ;;CLEAR THE STACK AFTER A TIME OUT
6$:     ;#####END OF CODE FOR THE XOR TESTER#####
        TSTB   $ERFLG        ;;HAS AN ERROR OCCURRED?
        BEQ    3$            ;;BR IF NO
4$:     CLRB   $ERFLG        ;;ZERO THE ERROR FLAG
        CLR    $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
3$:     BIT    #BIT11,@SWR    ;;INHIBIT ITERATIONS?

```


TYPE ROUTINE

1122	004430	122737	000001	001336	CMPB	#APTENV,\$ENV	::RUNNING IN APT MODE
1123	004436	001011			BNE	62\$::NO,GO CHECK FOR APT CONSOLE
1124	004440	132737	000100	001337	BITB	#APTSPOOL,\$ENV	::SPOOL MESSAGE TO APT
1125	004446	001405			BEQ	62\$::NO,GO CHECK FOR CONSOLE
1126	004450	010037	004460		MOV	RO,61\$::SETUP MESSAGE ADDRESS FOR APT
1127	004454	004737	004700		JSR	PC,\$ATY3	::SPOOL MESSAGE TO APT
1128	004460	000000					
1129	004462	132737	000040	001337	61\$:	.WORD	0
1130	004470	001003			62\$:	BITB	#APTCSUP,\$ENV
1131	004472	112046				BNE	60\$
1132	004474	001005			2\$:	MOVB	(RO)+,-(SP)
1133	004476	005726				BNE	4\$
1134	004500	012600				TST	(SP)+
1135	004502	062716	000002		60\$:	MOV	(SP)+,RO
1136	004506	000002			3\$:	ADD	#2,(SP)
1137	004510	122716	000011			RTI	
1138	004514	001430			4\$:	CMPB	#HT,(SP)
1139	004516	122716	000200			BEQ	8\$
1140	004522	001006				CMPB	#CRLF,(SP)
1141	004524	005726				BNE	5\$
1142	004526	104401				TST	(SP)+
1143	004530	001313				TYPE	
1144	004532	105037	004666			\$CRLF	
1145	004536	000755				CLRB	\$CHARCNT
1146	004540	004737	004622			BR	2\$
1147	004544	123726	001256		5\$:	JSR	PC,\$TYPEC
1148	004550	001350			6\$:	CMPB	\$FILLC,(SP)+
1149	004552	013746	001254			BNE	2\$
1150						MOV	\$NULL,-(SP)
1151	004556	105366	000001				
1152	004562	002770			7\$:	DECB	1(SP)
1153	004564	004737	004622			BLT	6\$
1154	004570	105337	004666			JSR	PC,\$TYPEC
1155	004574	000770				DECB	\$CHARCNT
1156						BR	7\$
1157							
1158							
1159	004576	112716	000040				
1160	004602	004737	004622		8\$:	MOVB	#' ,(SP)
1161	004606	132737	000007	004666	9\$:	JSR	PC,\$TYPEC
1162	004614	001372				BITB	#7,\$CHARCNT
1163	004616	005726				BNE	9\$
1164	004620	000724				TST	(SP)+
1165	004622	105777	174422			BR	2\$
1166	004626	100375			\$TYPEC.	TSTB	@\$TPS
1167	004630	116677	000002	174414		BPL	\$TYPEC
1168	004636	122766	000015	000002		MOVB	2(SP),@\$TPB
1169	004644	001003				CMPB	#CR,2(SP)
1170	004646	105037	004666			BNE	1\$
1171	004652	000406				CLRB	\$CHARCNT
1172	004654	122766	000012	000002		BR	\$TYPEX
1173	004662	001402			1\$:	CMPB	#LF,2(SP)
1174	004664	105227				BEQ	\$TYPEX
1175	004666	000000				INCB	(PC)+
1176	004670	000207			\$CHARCNT:	.WORD	0
1177					\$TYPEX:	RTS	PC

;HORIZONTAL TAB PROCESSOR

APT COMMUNICATIONS ROUTINE

.SBTTL APT COMMUNICATIONS ROUTINE

::*****

1178									
1179									
1180									
1181	004672	112737	000001	005136		\$ATY1:	MOVB	#1,\$FFLG	::TO REPORT FATAL ERROR
1182	004700	112737	000001	005134		\$ATY3:	MOVB	#1,\$MFLG	::TO TYPE A MESSAGE
1183	004706	000403					BR	\$ATYC	
1184	004710	112737	000001	005136		\$ATY4:	MOVB	#1,\$FFLG	::TO ONLY REPORT FATAL ERROR
1185	004716					\$ATYC:			
1186	004716	010046					MOV	R0,-(SP)	::PUSH R0 ON STACK
1187	004720	010146					MOV	R1,-(SP)	::PUSH R1 ON STACK
1188	004722	105737	005134				TSTB	\$MFLG	::SHOULD TYPE A MESSAGE?
1189	004726	001450					BEQ	5\$::IF NOT: BR
1190	004730	122737	000001	001336			CMPB	#APTENV,\$ENV	::OPERATING UNDER APT?
1191	004736	001031					BNE	3\$::IF NOT: BR
1192	004740	132737	000100	001337			BITB	#APTPOOL,\$ENVM	::SHOULD SPOOL MESSAGES?
1193	004746	001425					BEQ	3\$::IF NOT: BR
1194	004750	017600	000004				MOV	@4(SP),R0	::GET MESSAGE ADDR.
1195	004754	062766	000002	000004			ADD	#2,4(SP)	::BUMP RETURN ADDR.
1196	004762	005737	001316			1\$:	TST	\$MSGTYPE	::SEE IF DONE W/ LAST XMISSION?
1197	004766	001375					BNE	1\$::IF NOT: WAIT
1198	004770	010037	001332				MOV	R0,\$MSGAD	::PUT ADDR IN MAILBOX
1199	004774	105720				2\$:	TSTB	(R0)+	::FIND END OF MESSAGE
1200	004776	001376					BNE	2\$	
1201	005000	163700	001332				SUB	\$MSGAD,R0	::SUB START OF MESSAGE
1202	005004	006200					ASR	R0	::GET MESSAGE LNTH IN WORDS
1203	005006	010037	001334				MOV	R0,\$MSGGLT	::PUT LENGTH IN MAILBOX
1204	005012	012737	000004	001316			MOV	#4,\$MSGTYPE	::TELL APT TO TAKE MSG.
1205	005020	000413					BR	5\$	
1206	005022	017637	000004	005046		3\$:	MOV	@4(SP),4\$::PUT MSG ADDR IN JSR LINKAGE
1207	005030	062766	000002	000004			ADD	#2,4(SP)	::BUMP RETURN ADDRESS
1208	005036	013746	177776				MOV	177776,-(SP)	::PUSH 177776 ON STACK
1209	005042	004737	004410				JSR	PC,\$TYPE	::CALL TYPE MACRO
1210	005046	000000				4\$:	.WORD	0	
1211	005050					5\$:			
1212	005050	105737	005136			10\$:	TSTB	\$FFLG	::SHOULD REPORT FATAL ERROR?
1213	005054	001416					BEQ	12\$::IF NOT: BR
1214	005056	005737	001336				TST	\$ENV	::RUNNING UNDER APT?
1215	005062	001413					BEQ	12\$::IF NOT: BR
1216	005064	005737	001316			11\$:	TST	\$MSGTYPE	::FINISHED LAST MESSAGE?
1217	005070	001375					BNE	11\$::IF NOT: WAIT
1218	005072	017637	000004	001320			MOV	@4(SP),\$FATAL	::GET ERROR #
1219	005100	062766	000002	000004			ADD	#2,4(SP)	::BUMP RETURN ADDR.
1220	005106	005237	001316				INC	\$MSGTYPE	::TELL APT TO TAKE ERROR
1221	005112	105037	005136			12\$:	CLRB	\$FFLG	::CLEAR FATAL FLAG
1222	005116	105037	005135				CLRB	\$LFLG	::CLEAR LOG FLAG
1223	005122	105037	005134				CLRB	\$MFLG	::CLEAR MESSAGE FLAG
1224	005126	012601					MOV	(SP)+,R1	::POP STACK INTO R1
1225	005130	012600					MOV	(SP)+,R0	::POP STACK INTO R0
1226	005132	000207					RTS	PC	::RETURN
1227	005134	000				\$MFLG:	.BYTE	0	::MESSG. FLAG
1228	005135	000				\$LFLG:	.BYTE	0	::LOG FLAG
1229	005136	000				\$FFLG:	.BYTE	0	::FATAL FLAG
1230		005140					.EVEN		
1231		000200				APTSIZE=	200		
1232		000001				APTENV=	001		
1233		000100				APTPOOL=	100		

APT COMMUNICATIONS ROUTINE

```

1234          000040
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253 005140 011646
1254 005142 016666 000004 000002
1255 005150 105777 174070
1256 005154 100375
1257 005156 117766 174064 000004
1258 005164 042766 177600 000004
1259 005172 026627 000004 000023
1260 005200 001013
1261 005202 105777 174036
1262 005206 100375
1263 005210 117746 174032
1264 005214 042716 177600
1265 005220 022627 000021
1266 005224 001366
1267 005226 000750
1268 005230 026627 000004 000140
1269 005236 002407
1270 005240 026627 000004 000175
1271 005246 003003
1272 005250 042766 000040 000004
1273 005256 000002
1274
1275
1276
1277
1278
1279
1280
1281 005260 010346
1282 005262 005046
1283 005264 012703 005514
1284 005270 022703 005523
1285 005274 101456
1286 005276 104402
1287 005300 112613
1288 005302 122713 000177
1289 005306 001022
  
```

```

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
:

$RDCHR: MOV      (SP),-(SP)      :: PUSH DOWN THE PC
        MOV      4(SP),2(SP)    :: SAVE THE PS
1$:     TSTB     @STKS          :: WAIT FOR
        BPL      1$             :: A CHARACTER
        MOVB     @STKB,4(SP)    :: READ THE TTY
        BIC      #^C<177>,4(SP) :: GET RID OF JUNK IF ANY
        CMP      4(SP),#23     :: IS IT A CONTROL-S?
        BNE      3$            :: BRANCH IF NO
2$:     TSTB     @STKS          :: WAIT FOR A CHARACTER
        BPL      2$             :: LOOP UNTIL ITS THERE
        MOVB     @STKB,-(SP)    :: GET CHARACTER
        BIC      #^C177,(SP)   :: MAKE IT 7-BIT ASCII
        CMP      (SP)+,#21     :: IS IT A CONTROL-Q?
        BNE      2$            :: IF NOT DISCARD IT
        BR       1$            :: YES, RESUME
3$:     CMP      4(SP),#140     :: IS IT UPPER CASE?
        BLT      4$            :: BRANCH IF YES
        CMP      4(SP),#175    :: IS IT A SPECIAL CHAR?
        BGT      4$            :: BRANCH IF YES
        BIC      #40,4(SP)     :: MAKE IT UPPER CASE
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 STAC
: *                 :: TERMINATOR WILL BE A BYTE OF ALL 0'S
:

$RDLIN: MOV      R3,-(SP)      :: SAVE R3
        CLR      -(SP)         :: CLEAR THE RUBOUT KEY
1$:     MOV      #$TTYIN,R3    :: GET ADDRESS
2$:     CMP      #$TTYIN+7,R3  :: BUFFER FULL?
        BLOS     4$            :: BR IF YES
        RDCHR    :: GO READ ONE CHARACTER FROM THE TTY
        MOVB     (SP)+,(R3)    :: GET CHARACTER
10$:    CMPB     #177,(R3)     :: IS IT A RUBOUT
        BNE     5$            :: BR IF NO
  
```

TTY INPUT ROUTINE

1290	005310	005716			TST	(SP)	:: IS THIS THE FIRST RUBOUT?
1291	005312	001007			BNE	6\$:: BR IF NO
1292	005314	112737	000134	005512	MOVB	#'\,9\$:: TYPE A BACK SLASH
1293	005322	104401	005512		TYPE	,9\$	
1294	005326	012716	177777		MOV	#-1,(SP)	:: SET THE RUBOUT KEY
1295	005332	005303			6\$: DEC	R3	:: BACKUP BY ONE
1296	005334	020327	005514		CMP	R3,#\$TTYIN	:: STACK EMPTY?
1297	005340	103434			BLO	4\$:: BR IF YES
1298	005342	111337	005512		MOVB	(R3),9\$:: SETUP TO TYPEOUT THE DELETED CHAR.
1299	005346	104401	005512		TYPE	,9\$:: GO TYPE
1300	005352	000746			BR	2\$:: GO READ ANOTHER CHAR.
1301	005354	005716			5\$: TST	(SP)	:: RUBOUT KEY SET?
1302	005356	001406			BEQ	7\$:: BR IF NO
1303	005360	112737	000134	005512	MOVB	#'\,9\$:: TYPE A BACK SLASH
1304	005366	104401	005512		TYPE	,9\$	
1305	005372	005016			CLR	(SP)	:: CLEAR THE RUBOUT KEY
1306	005374	122713	000025		7\$: CMPB	#25,(R3)	:: IS CHARACTER A CTRL U?
1307	005400	001003			BNE	8\$:: BR IF NO
1308	005402	104401	005523		TYPE	,\$CNTLU	:: TYPE A CONTROL 'U'
1309	005406	000726			BR	1\$:: GO START OVER
1310	005410	122713	000022		8\$: CMPB	#22,(R3)	:: IS CHARACTER A '^R'?
1311	005414	001011			BNE	3\$:: BRANCH IF NO
1312	005416	105013			CLRB	(R3)	:: CLEAR THE CHARACTER
1313	005420	104401	001313		TYPE	,\$CRLF	:: TYPE A 'CR' & 'LF'
1314	005424	104401	005514		TYPE	,\$TTYIN	:: TYPE THE INPUT STRING
1315	005430	000717			BR	2\$:: GO PICKUP ANOTHER CHACTER
1316	005432	104401	001312		4\$: TYPE	,\$QUES	:: TYPE A '?'
1317	005436	000712			BR	1\$:: CLEAR THE BUFFER AND LOOP
1318	005440	111337	005512		3\$: MOVB	(R3),9\$:: ECHO THE CHARACTER
1319	005444	104401	005512		TYPE	,9\$	
1320	005450	122723	000015		CMPB	#15,(R3)+	:: CHECK FOR RETURN
1321	005454	001305			BNE	2\$:: LOOP IF NOT RETURN
1322	005456	105063	177777		CLRB	-1(R3)	:: CLEAR RETURN (THE 15)
1323	005462	104401	001314		TYPE	,\$LF	:: TYPE A LINE FEED
1324	005466	005726			TST	(SP)+	:: CLEAN RUBOUT KEY FROM THE STACK
1325	005470	012603			MOV	(SP)+,R3	:: RESTORE R3
1326	005472	011646			MOV	(SP),-(SP)	:: ADJUST THE STACK AND PUT ADDRESS OF THE
1327	005474	016666	000004	000002	MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
1328	005502	012766	005514	000004	MOV	#\$TTYIN,4(SP)	
1329	005510	000002			RTI		:: RETURN
1330	005512	000			9\$: .BYTE	0	:: STORAGE FOR ASCII CHAR. TO TYPE
1331	005513	000			.BYTE	0	:: TERMINATOR
1332	005514	000007			\$TTYIN: .BLKB	7	:: RESERVE 7 BYTES FOR TTY INPUT
1333	005523	136	006525	000012	\$CNTLU: .ASCIZ	/^U/<15><12>	:: CONTROL 'U'
1334	005530	043536	005015	000	\$CNTLG: .ASCIZ	/^G/<15><12>	:: CONTROL 'G'
1335	005535	015	051412	051127	\$MSWR: .ASCIZ	<15><12>/SWR = /	
1336	005542	036440	000040		\$MNEW: .ASCIZ	/ NEW = /	
1337	005546	020040	042516	020127			
1338	005554	020075	000				
1339		005560			.EVEN		
1340					.SBTTL	READ AN OCTAL NUMBER FROM THE TTY	
1341							
1342							
1343							
1344							
1345							

 *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 *CHANGE IT TO BINARY.
 *THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL

READ AN OCTAL NUMBER FROM THE TTY

```

1346 ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
1347 ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
1348 ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
1349 ;*CALL:
1350 ;*      RDOCT          ;;READ AN OCTAL NUMBER
1351 ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
1352 ;*                   ;;HIGH ORDER BITS ARE IN $HIOCT
1353
1354 005560 011646          $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
1355 005562 016666 000004 000002  MOV      4(SP),2(SP)    ;;INPUT NUMBER
1356 005570 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
1357 005572 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
1358 005574 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
1359 005576 104403          1$:      RDLIN          ;;READ AN ASCII LINE
1360 005600 012600          MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
1361 005602 010037 005706  MOV      R0,5$      ;;AND SAVE IT
1362 005606 005001          CLR      R1          ;;CLEAR DATA WORD
1363 005610 005002          CLR      R2
1364 005612 112046          2$:      MOVB     (R0)+,-(SP)    ;;PICKUP THIS CHARACTER
1365 005614 001420          BEQ      3$          ;;IF ZERO GET OUT
1366 005616 122716 000060  CMPB     #'0,(SP)    ;;MAKE SURE THIS CHARACTER
1367 005622 003026          BGT      4$          ;;IS AN OCTAL DIGIT
1368 005624 122716 000067  CMPB     #'7,(SP)
1369 005630 002423          BLT      4$
1370 005632 006301          ASL     R1          ;;*2
1371 005634 006102          ROL     R2
1372 005636 006301          ASL     R1          ;;*4
1373 005640 006102          ROL     R2
1374 005642 006301          ASL     R1          ;;*8
1375 005644 006102          ROL     R2
1376 005646 042716 177770  BIC     #'C7,(SP)    ;;STRIP THE ASCII JUNK
1377 005652 062601          ADD     (SP)+,R1    ;;ADD IN THIS DIGIT
1378 005654 000756          BR      2$          ;;LOOP
1379 005656 005726          3$:      TST     (SP)+    ;;CLEAN TERMINATOR FROM STACK
1380 005660 010166 000012  MOV      R1,12(SP)  ;;SAVE THE RESULT
1381 005664 010237 005716  MOV      R2,$HIOCT
1382 005670 012602          MOV     (SP)+,R2    ;;POP STACK INTO R2
1383 005672 012601          MOV     (SP)+,R1    ;;POP STACK INTO R1
1384 005674 012600          MOV     (SP)+,R0    ;;POP STACK INTO R0
1385 005676 000002          RTI          ;;RETURN
1386 005700 005726          4$:      TST     (SP)+    ;;CLEAN PARTIAL FROM STACK
1387 005702 105010          CLR/B   (R0)      ;;SET A TERMINATOR
1388 005704 104401          TYPE   0          ;;TYPE UP THRU THE BAD CHAR.
1389 005706 000000          5$:      .WORD   0
1390 005710 104401 001312  TYPE   $QUES      ;; '?' 'CR' & 'LF'
1391 005714 000730          BR      1$          ;;TRY AGAIN
1392 005716 000000          $HIOCT: .WORD   0  ;;HIGH ORDER BITS GO HERE
1393
1394 ;
1395 ;-----
1396 ;
1397 005720 010546          $INPUT: MOV     R5,-(SP)  ;; SAVE REGISTER R5.
1398 005722 016605 000002  MOV     2(SP),R5    ;; GET FIRST PARAMETER ADDRESS.
1399 005726 012537 005764  MOV     (R5)+,WHAT  ;; GET MESSAGE ADDRESS.
1400 005732 012537 006044  MOV     (R5)+,LOLIM ;; GET LOW LIMIT FOR THE #
1401 005736 012537 006046  MOV     (R5)+,HILIM ;; GET HIGH LIMIT FOR THE #.

```

READ AN OCTAL NUMBER FROM THE TTY

1402	005742	012537	006050		MOV	(R5)+,WHERE	:	GET ADDRESS OF INBUFFER
1403	005746	112537	006052		MOVB	(R5)+,LOBITS	:	GET LOWMASK BITS.
1404	005752	112537	006053		MOVB	(R5)+,ADRCNT	:	GET # OF #'S TO BE GENERATED.
1405	005756	010566	000002		MOV	R5,2(SP)	:	SAVE THE RETURN ADDRESS.
1406	005762	104401		INLP1:	TYPE		:	TYPE THE MESSAGE.
1407	005764	000000		WHAT:	.WORD	0		
1408	005766	104404			RDOCT			
1409	005770	021637	006046		CMP	(SP),HILIM	:	READ OCTAL # FROM KEYBOARD.
1410	005774	003003			BGT	2\$:	IS IT IN HIGH LIMIT?
1411	005776	021637	006044		CMP	(SP),LOLIM	:	BRANCH IF NO.
1412	006002	002005			BGE	3\$:	IS IT MORE THAN LOW LIMIT.
1413	006004	104401	001312	2\$:	TYPE	,\$QUES	:	BRANCH IF YES.
1414	006010	104401	001313		TYPE	,\$CRLF	:	TYPE ' '? '
1415	006014	000762			BR	INLP1	:	TYPE <CR>,<LF>
1416	006016	013705	006050	3\$:	MOV	WHERE,R5	:	GET BUFFER ADDRESS.
1417	006022	011625		4\$:	MOV	(SP),(R5)+	:	SAVE THE # IN RIGHT PLACE.
1418	006024	062716	000002		ADD	#2,(SP)	:	NEXT SEQUENTIAL NUMBER.
1419	006030	105337	006053		DECB	ADRCNT	:	COUNT BY 1.
1420	006034	001372			BNE	4\$:	BRANCH IF NOT DONE.
1421	006036	005726			TST	(SP)+	:	POP THE STACK POINTER.
1422	006040	012605			MOV	(SP)+,R5	:	POP THE REG.5
1423	006042	000002			RTI			
1424	006044	000000		LOLIM:	.WORD	0		
1425	006046	000000		HILIM:	.WORD	0		
1426	006050	000000		WHERE:	.WORD	0		
1427	006052	000		LOBITS:	.BYTE	0		
1428	006053	000		ADRCNT:	.BYTE	0		
1429								
1430								
1431								
1432								
1433	006054	013716	001442	.ADVANCE:	MOV	NEXT,(SP)	:	CRUNCH STACK WITH ADDRESS OF SC
1434	006060	005037	001444		CLR	LOCK	:	RESET TIGHT LOOP ADDRESS
1435	006064	000002			RTI		:	CHECK TO SEE IF OLD TEST GETS REPEATED
1436								
1437								
1438								
1439								
1440	006066	016637	000004 001460	.SAV05:	MOV	4(SP),SAVPC	:	SAVE R7 (PC)
1441								
1442								
1443								
1444	006074	010537	001274	SV05:	MOV	R5,\$REG5	:	SAVE R5
1445	006100	010437	001272		MOV	R4,\$REG4	:	SAVE R4
1446	006104	010337	001270		MOV	R3,\$REG3	:	SAVE R3
1447	006110	010237	001266		MOV	R2,\$REG2	:	SAVE R2
1448	006114	010137	001264		MOV	R1,\$REG1	:	SAVE R1
1449	006120	010037	001262		MOV	R0,\$REG0	:	SAVE R0
1450	006124	000002			RTI		:	LEAVE.
1451								
1452								
1453								
1454	006126	013700	001262	.RES05:	MOV	\$REG0,R0	:	RESTORE R0
1455	006132	013701	001264		MOV	\$REG1,R1	:	RESTORE R1
1456	006136	013702	001266		MOV	\$REG2,R2	:	RESTORE R2
1457	006142	013703	001270		MOV	\$REG3,R3	:	RESTORE R3

READ AN OCTAL NUMBER FROM THE TTY

```

1458 006146 013704 001272      MOV      $REG4,R4      ;RESTORE R4
1459 006152 013705 001274      MOV      $REG5,R5      ;RESTORE R5
1460 006156 000002                RTI                    ;LEAVE
1461
1462                               ;
1463                               ;: CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
1464                               ;-----
1465 006160 104401 001313      .CONVR: TYPE          , $CRLF
1466 006164 010046      .CNVRT: MOV            R0,-(SP)
1467 006166 010146      MOV            R1,-(SP)
1468 006170 010346      MOV            R3,-(SP)
1469 006172 010446      MOV            R4,-(SP)
1470 006174 010546      MOV            R5,-(SP)
1471 006176 017601 000012      MOV            @12(SP),R1
1472 006202 062766 000002 000012      ADD            #2,12(SP)
1473 006210 012137 006402      MOV            (R1)+,WRDCNT
1474 006214 112137 006404      1$:  MOVB         (R1)+,CHRCNT
1475 006220 112137 006405      MOVB         (R1)+,SPACNT
1476 006224 013137 006406      MOV            @ (R1)+,BINWRD
1477 006230 122737 000003 006404      CMPB         #3,CHRCNT
1478 006236 001003                BNE            2$
1479 006240 042737 177400 006406      BIC            #177400,BINWRD
1480 006246 013704 006406      2$:  MOV            BINWRD,R4
1481 006252 113705 006404      MOVB         CHRCNT,R5
1482 006256 012700 011122      MOV            #TEMP,R0
1483 006262 010403      3$:  MOV            R4,R3
1484 006264 042703 177770      BIC            #177770,R3
1485 006270 062703 000060      ADD            #060,R3
1486 006274 110320      MOVB         R3,(R0)+
1487 006276 000241                CLC
1488 006300 006004                ROR            R4
1489 006302 000241                CLC
1490 006304 006004                ROR            R4
1491 006306 000241                CLC
1492 006310 006004                ROR            R4
1493 006312 005305                DEC            R5
1494 006314 001362                BNE            3$
1495 006316 012703 011164      MOV            #MDATA,R3
1496 006322 114023      4$:  MOVB         -(R0),(R3)+
1497 006324 105337 006404      DECB         CHRCNT
1498 006330 001374                BNE            4$
1499 006332 105737 006405      TSTB         SPACNT
1500 006336 001405                BEQ            6$
1501 006340 112723 000040      5$:  MOVB         #040,(R3)+
1502 006344 105337 006405      DECB         SPACNT
1503 006350 001373                BNE            5$
1504 006352 105013      6$:  CLRB         (R3)
1505 006354 104401 011164      TYPE         ,MDATA
1506 006360 005337 006402      DEC            WRDCNT
1507 006364 001313                BNE            1$
1508 006366 012605      MOV            (SP)+,R5
1509 006370 012604      MOV            (SP)+,R4
1510 006372 012603      MOV            (SP)+,R3
1511 006374 012601      MOV            (SP)+,R1
1512 006376 012600      MOV            (SP)+,R0
1513 006400 000002                RTI
  
```

READ AN OCTAL NUMBER FROM THE TTY

1514 006402 000000
1515 006404 000000
1516 006405 006405
1517 006406 000000

WRDCNT: 0
CHRCNT: 0
SPACNT=CHRCNT+1
BINWRD: 0

1518
1519
1520
1521
1522
1523
1524

:TRAP DISPATCH SERVICE
:ARGUMENT OF TRAP IS EXTRACTED
:AND USED AS OFFSET TO OBTAIN POINTER
:TO SELECTED SUBROUTINE

.SBTTL TRAP DECODER

1525
1526
1527
1528
1529
1530
1531
1532

:*****
:*THIS ROUTINE WILL PICKUP THE LOWER EYTE OF THE 'TRAP' INSTRUCTION
:*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
:*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
:*GO TO THAT ROUTINE.

1533 006410 010046
1534 006412 016600 000002
1535 006416 005740
1536 006420 111000
1537 006422 006300
1538 006424 016000 006444
1539 006430 000200

\$TRAP: MOV R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE

1540
1541
1542
1543
1544
1545
1546

;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO

1547
1548
1549
1550
1551
1552
1553
1554

\$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
RTI ;;RESTORE THE PSW

1555 006444 006432
1556 006446 004410
1557
1558
1559 006450 005140
1560 006452 005260
1561 006454 005560
1562 006456 004360
1563 006460 006066
1564 006462 006126
1565 006464 007356
1566 006466 007326
1567 006470 007374
1568 006472 007442
1569 006474 007506

.SBTTL TRAP TABLE

*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
*BY THE 'TRAP' INSTRUCTION.

: ROUTINE
:-----
\$TRPAD: .WORD \$TRAP2
\$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE

\$RDCHR ;;CALL=RDCHR TRAP+2(104402) TTY TYPEIN CHARACTER ROUTINE
\$RDLIN ;;CALL=RDLIN TRAP+3(104403) TTY TYPEIN STRING ROUTINE
\$RDOCT ;;CALL=RDOCT TRAP+4(104404) READ AN OCTAL NUMBER FROM TTY
.SCOP1 ;;CALL=SCOP1 TRAP+5(104405) CALL TO LOOP ON CURRENT DATA HAN
.SAV05 ;;CALL=SAV05 TRAP+6(104406) CALL TO REGISTER SAVE ROUTINE
.RES05 ;;CALL=RES05 TRAP+7(104407) CALL TO REGISTER RESTORE ROUTINE
.MSTCLR ;;CALL=MSTCLR TRAP+10(104410) CALL TO ISSUE A MASTER CLEAR
.DELAY ;;CALL=DELAY TRAP+11(104411) CALL TO DELAY
.ROMCLK ;;CALL=ROMCLK TRAP+12(104412) CALL TO CLOCK ROM ONCE
.DATACLK ;;CALL=DATACLK TRAP+13(104413) CALL TO CLOCK DATA
.TIMER ;;CALL=TIMER TRAP+14(104414) CALL TO DELAY A CLOCK TICK

TRAP TABLE

```

1570 006476 005720
1571 006500 006160
1572 006502 006164
1573 006504 006054
1574
1575
1576
1577
1578
1579
1580 006506 004737 011226
1581 006512 032777 010000 172520
1582 006520 001406
1583 006522 105777 172522
1584 006526 100003
1585 006530 112777 000207 172514
1586 006536 032777 020000 172474
1587 006544 001107
1588 006546 021637 001216
1589 006552 001404
1590 006554 011637 001216
1591 006560 105037 001203
1592 006564 104406
1593 006566 011605
1594 006570 162705 000002
1595 006574 011504
1596 006576 110437 001214
1597 006602 006304
1598 006604 061504
1599 006606 006304
1600 006610 042704 177001
1601 006614 062704 001512
1602 006620 012437 006734
1603 006624 012437 006746
1604 006630 011437 006760
1605 006634 105737 001203
1606 006640 001403
1607 006642 005737 006760
1608 006646 001040
1609 006650 104401 001313
1610 006654 104401 001313
1611 006660 005737 001444
1612 006664 001402
1613 006666 104401 010011
1614 006672 104401 007777
1615 006676 104417 007114
1616 006702 104401 010066
1617 006706 104417 007106
1618 006712 104401 001313
1619 006716 112737 177777 001203
1620 006724 005737 006734
1621 006730 001402
1622 006732 104401
1623 006734 000000
1624 006736
1625 006736 005737 006746
  
```

```

$INPUT ;;CALL=INPUT TRAP+15(104415) CALL TO OCTAL # INPUT ROUTINE
.CONVRT ;;CALL=CONVRT TRAP+16(104416) CALL TO .....
.CNVRT ;;CALL=CNVRT TRAP+17(104417) CALL TO .....
.ADVANCE ;;CALL=ADVANCE TRAP+20(104420) CALL TO ADVANCE TO NEXT
:
:-----
:*****
:ERROR HANDLER
:-----
$ERROR: JSR PC,CKSWR ;CHECK FOR SOFT SWR
BIT #SW12,@SWR ;BELL ON ERROR?
BEQ XBX ;BR IF NO BELL
TSTB @$TPS ;TTY READY.
BPL XBX ;DON'T WAIT IF TTY NOT READY.
MOVB #207,@$TPB ;PUSH A BELL AT THE TTY.
XBX: BIT #SW13,@SWR ;DELETE ERROR PRINT OUT?
BNE HALTS ;BR IF NO PRINT OUT WANTED.
CMP (SP),$ERRPC ;WAS THIS ERROR FOUND LAST TIME?
BEQ 1$ ;BR IF YES
MOV (SP),$ERRPC ;RECORD BEING HERE
CLRB $ERFLG ;PREPARE HEADER
1$: SAVO5 ;SAVE ALL PROC REGISTERS
MOV (SP),R5 ;GET THE PC OF ERROR
SUB #2,R5 ;GET ADDRESS OF TRAP CALL
MOV (R5),R4 ;GET ERROR INSTRUCTION
MOVB R4,$ITEMB ; COPY ERROR # FOR APT HANDLING
ASL R4 ;MULT BY TWO
ADD (R5),R4 ;DOUBLE IT
ASL R4 ;MULT AGAIN
BIC #177001,R4 ;CLEAR JUNK
ADD #$ERRTB,R4 ;GET POINTER
MOV (R4)+,ERRMSG ;GET ERROR MESSAGE
MOV (R4)+,DATAHD ;GET DATA HEADRER
MOV (R4),DATABP ;GET DATA TABLE
TSTB $ERFLG ;TYPE HEADREER
BEQ TYPMSG ;BR IF YES
TST DATABP ;DOES DATA TABLE EXIST?
BNE TYPDAT ;BR IF YES.
TYPMSG: TYPE , $CRLF
TYPE , $CRLF
TST LOCK
BEQ 1$
1$: TYPE ,MASTEK
TYPE ,MTSTN
CNVRT ,XTSTN ;SHOW IT
TYPE ,MERRPC ;TYPE PC.
CNVRT ,ERTAB0 ;SHOW IT
TYPE , $CRLF ;GIVE A CR/LF
MOVB #-1,$ERFLG ;NO MORE HEADER UNLESS NO DATA TABLE.
TST ERRMSG ;IS THERE AN ERROR MESSAGE?
BEQ WRKO.FM ;BR IF NO.
TYPE ;TYPE
ERRMSG: 0 ; ERROR MESSAGE
WRKO.FM: ;
TST DATAHD ;DATA HEADER?
  
```

TRAP TABLE

1626	006742	001402				BEQ	TYPDAT		:BR IF NO
1627	006744	104401				TYPE			:TYPE
1628	006746	000000				DATAHD: 0			: DATA HEADER
1629	006750	005737	006760			TYPDAT: TST	DATABP		:DATA TABLE?
1630	006754	001402				BEQ	RESREG		:BR IF NO.
1631	006756	104416				CONVRT			:SHOW
1632	006760	000000				DATABP: 0			: DATA TABLE
1633	006762	104407				RESREG: RES05			:RESTORE PROC REGISTERS
1634	006764	122737	000001	001336		HALTS: CMPB	#APTENV,\$ENV		: IS APT RUNNING ?
1635	006772	001007				BNE	3\$: SKIP APT CALL IF NOT.
1636	006774	113737	001214	007006		MOVB	\$ITEMB,6\$: COPY ERROR #.
1637	007002	004737	004710			JSR	PC,\$ATY4		: CALL APT SERVICES.
1638	007006	000000				6\$: .WORD	0		: ERROR # GOES HERE.
1639	007010	000777				9\$: BR	9\$: LOCK HERE.
1640	007012	022737	004070	000042		3\$: CMP	#\$ENDAD,@#42		:IF ACT-11 AUTOMATIC MODE, HALT!!
1641	007020	001403				BEQ	1\$		
1642	007022	005777	172212			TST	@SWR		:HALT ON ERROR?
1643	007026	100005				BPL	EXITER		:BR IF NO HALT ON ERROR
1644	007030	010046				1\$: PUSHRO			:SAVE RO
1645	007032	016600	000002			MOV	2(SP),R0		:SHOW ERROR PC IN DATA LIGHTS
1646	007036	000000				HALT			:HALT
1647	007040	012600				POPPO			:GET RO
1648	007042	005237	001212			EXITER: INC	\$ERTTL		:UPDATE ERROR COUNT
1649	007046	032777	000400	172164		BIT	#\$SW08,@SWR		:GOTO TOP OF TEST?
1650	007054	001007				BNE	1\$:BR IF YES
1651	007056	032777	002000	172154		BIT	#\$SW10,@SWR		:GOTO NEXT TEST?
1652	007064	001407				BEQ	2\$:BR IF NO
1653	007066	013737	001442	001206		MOV	NEXT,\$LPADR		:SET FOR NEXT TEST
1654	007074	012706	001200			1\$: MOV	#\$STACK,SP		:RESET SP
1655	007100	000177	172102			JMP	@\$LPADR		:GOTO SPECIFIED TEST
1656	007104	000002				2\$: RTI			: \$LPADR
1657	007106	000001				ERTAB0: 1			
1658	007110	006	002			.BYTE	6,2		
1659	007112	001460				SAVPC			
1660	007114	000001				XTSTN: 1			
1661	007116	003	002			.BYTE	3,2		
1662	007120	001202				\$STNM			:ENTER HERE ON POWER FAILURE
1663									:-----
1664									
1665									
1666									
1667									
1668									
1669									
1670	007122	012737	007312	000024					
1671	007130	012737	000340	000026					
1672	007136	010046							
1673	007140	010146							
1674	007142	010246							
1675	007144	010346							
1676	007146	010446							
1677	007150	010546							
1678	007152	017746	172062						
1679	007156	010637	007316						
1680	007162	012737	007174	000024					
1681	007170	000000							

.SBTTL POWER DOWN AND UP ROUTINES

:::*****

:POWER DOWN ROUTINE

```

$PWRDN: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
        MOV    #340,@#PWRVEC+2 ;;PRIO:7
        MOV    R0,-(SP)        ;;PUSH R0 ON STACK
        MOV    R1,-(SP)        ;;PUSH R1 ON STACK
        MOV    R2,-(SP)        ;;PUSH R2 ON STACK
        MOV    R3,-(SP)        ;;PUSH R3 ON STACK
        MOV    R4,-(SP)        ;;PUSH R4 ON STACK
        MOV    R5,-(SP)        ;;PUSH R5 ON STACK
        MOV    @SWR,-(SP)      ;;PUSH @SWR ON STACK
        MOV    SP,$SAVR6      ;;SAVE SP
        MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
        HALT
  
```


POWER DOWN AND UP ROUTINES

1682	007172	000776			BR	.-2	::HANG UP
1683							
1684							
1685							::*****
1686	007174	012737	007312	000024			::POWER UP ROUTINE
1687	007202	013706	007316		\$PWRUP:	MOV	#\$ILLUP,@#PWRVEC ::SET FOR FAST DOWN
1688	007206	005037	007316			MOV	\$\$SAVR6,SP ::GET SP
1689	007212	005237	007316		1\$:	CLR	\$\$SAVR6 ::WAIT LOOP FOR THE TTY
1690	007216	001375				INC	\$\$SAVR6 ::WAIT FOR THE INC
1691	007220	104401	007556			BNE	1\$::OF WORD
1692	007224	104417	007320			TYPE	,MPFAIL
1693	007230	105037	001203			CNVRT	,PFTAB
1694	007234	005037	001216			CLRB	\$ERFLG ::CLEAR ERROR FLAG.
1695	007240	013701	002066			CLR	\$ERRPC ::CLEAR LAST ERROR PC
1696	007244	005011				MOV	KMCSR,R1 ::RESTORE DEVICE ADDRESS.
1697	007246	104410				CLR	(R1) ::CLEAR THE CSR.
1698	007250	012677	171764			MSTCLR	
1699	007254	012605				MOV	(SP)+,@SWR ::POP STACK INTO @SWR
1700	007256	012604				MOV	(SP)+,R5 ::POP STACK INTO R5
1701	007260	012603				MOV	(SP)+,R4 ::POP STACK INTO R4
1702	007262	012602				MOV	(SP)+,R3 ::POP STACK INTO R3
1703	007264	012601				MOV	(SP)+,R2 ::POP STACK INTO R2
1704	007266	012600				MOV	(SP)+,R1 ::POP STACK INTO R1
1705	007270	012737	007122	000024		MOV	(SP)+,R0 ::POP STACK INTO R0
1706	007276	012737	000340	000026		MOV	#\$PWRDN,@#PWRVEC ::SET UP THE POWER DOWN VECTOR
1707	007304	104401				MOV	#340,@#PWRVEC+2 ::PRIO:7
1708	007306	007556				TYPE	
1709	007310	000002			\$PWRMG:	.WORD	MPFAIL ::REPORT THE POWER FAILURE
1710	007312	000000				RTI	::POWER FAIL MESSAGE POINTER
1711	007314	000776			\$ILLUP:	HALT	::THE POWER UP SEQUENCE WAS STARTED
1712	007316	000000				BR	.-2 ::BEFORE THE POWER DOWN WAS COMPLETE
1713						RTI	::PUT THE SP HERE
1714	007320	000001					
1715	007322	003	002		PFTAB:	1	
1716	007324	001202			.BYTE	3,2	
1717						\$TSTNM	
1718	007326				.DELAY:		
1719	007326	012777	000020	172540		MOV	#20,@KMP04
1720	007334	104412				ROMCLK	
1721	007336	121111				121111	::NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1722	007340				1\$:		::POKE CLOCK DELAY BIT
1723	007340	104412				ROMCLK	
1724	007342	121224				121224	::NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1725	007344	032777	000020	172522			::PORT4 IBUS*11
1726	007352	001772				BIT	#BIT4,@KMP04 ::IS CLOCK BIT SET?
1727	007354	000002				BEQ	1\$::BR IF NO
1728						RTI	
1729	007356				.MSTCLR:		
1730	007356	152777	000100	172504		BISB	#BIT6,@KMCSRH ::SET MASTER CLEAR
1731	007364	142777	000300	172476		BICB	#BIT6!BIT7,@KMCSRH ::CLEAR MASTER CLEAR AND RUN
1732	007372	000002				RTI	::RETURN
1733							
1734	007374				.ROMCLK:		
1735	007374	152777	000002	172466		BISB	#BIT1,@KMCSRH ::SET ROMI
1736	007402	013677	172470			MOV	@(SP)+,@KMP06 ::LOAD INSTRUCTION IN SEL6
1737	007406	062746	000002			ADD	#2,-(SP) ::ADJUST STACK

POWER DOWN AND UP ROUTINES

1738	007412	032777	000100	171620	BIT	#SW06,@SWR	:HALT IF SW06 =1
1739	007420	001401			BEQ	1\$:BR IF SW06 =0
1740	007422	000000			HALT		:HALT BEFORE CLOCKING INSTRUCTION
1741	007424	152777	000003	172436	1\$: BISB	#BIT1!BIT0,@KMCSRH	:CLOCK INSTRUCTION
1742	007432	142777	000007	172430	BICB	#BIT2!BIT1!BIT0,@KMCSRH	:CLEAR ROMO, ROMI, STEP
1743	007440	000002			RTI		
1744							
1745	007442				.DATACLK:		
1746	007442	013637	011122		MOV	@(SP)+,TEMP	:PUT TICK COUNT IN TEMP
1747	007446	062746	000002		ADD	#2,-(SP)	:ADJUST STACK
1748	007452	152777	000020	172410	1\$: BISB	#BIT4,@KMCSRH	:SET STEP LU
1749	007460	027777	172402	172400	CMP	@KMCSR,@KMCSR	:WASTE TIME
1750	007466	142777	000020	172374	BICB	#BIT4,@KMCSRH	:CLEAR STEP LU
1751	007474	005337	011122		DEC	TEMP	:DEC TICK COUNT
1752	007500	001364			BNE	1\$:BR IF NOT DONE
1753	007502	000002			RTI		:RETURN
1754	007504	000001			3\$: .BLKW 1		
1755							
1756	007506				.TIMER:		
1757	007506	013637	011122		MOV	@(SP)+,TEMP	:MOVE COUNT TO TEMP
1758	007512	062746	000002		ADD	#2,-(SP)	:ADJUST STACK
1759	007516				1\$: ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1760	007516	104412			021364		:PORT4 IBUS* REG11
1761	007520	021364			BIT	#2,@KMP04	:IS PG ^M CLOCK BIT CLEAR?
1762	007522	032777	000002	172344	BEQ	1\$:BR IF YES
1763	007530	001772			2\$: ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1764	007532				021364		:PORT4 IBUS* REG11
1765	007532	104412			BIT	#2,@KMP04	:IS PG ^M CLOCK BIT SET?
1766	007534	021364			BNE	2\$:BR IF YES
1767	007536	032777	000002	172330	DEC	TEMP	:DEC COUNT
1768	007544	001372			BNE	1\$:BR IF NOT DONE
1769	007546	005337	011122		RTI		:RETURN
1770	007552	001361					
1771	007554	000002					
1772							
1773	007556	050200	051127	043040	MPFAIL:	.ASCIZ	<200>/PWR FAILED. RESTART AT TEST /
(2)	007614	042600	042116	050040	MEPASS:	.ASCIZ	<200>/END PASS CZKCG /
(2)	007636	051200	000		MR:	.ASCIZ	<200>/R/
(2)	007641	200	047516	042040	MERR2:	.ASCIZ	<200>/NO DEVICES PRESENT./
(2)	007666	044600	051516	043125	MERR3:	.ASCIZ	<200>/INSUFFICIENT DATA! /
(?)	007712	046200	041517	020113	MLOCK:	.ASCIZ	<200>/LOCK ON SELECTED TEST/
(2)	007741	103	051123	020072	MCSRX:	.ASCIZ	/CSR: /
(2)	007747	126	041505	020072	MVECX:	.ASCIZ	/VEC: /
(2)	007755	120	051501	042523	MPASSX:	.ASCIZ	/PASSES: /
(2)	007766	051105	047522	051522	MERRX:	.ASCIZ	/ERRORS: /
(2)	007777	124	051505	020124	MTSTN:	.ASCIZ	/TEST NO: /
(2)	010011	052	000		MASTEK:	.ASCIZ	/* /
(2)	010013	200	042523	020124	MNEW:	.ASCIZ	<200>/SET SWITCH REG TO KMC11'S DESIRED ACTIVE./
(2)	010066	041520	020072	000	MERRPC:	.ASCIZ	/PC: /
(2)	010073	200	020040	020040	XHEAD:	.ASCIZ	<200>/
(2)	010132	020200	020040	020040		.ASCIZ	<200>/
(2)	010171	200	020040	041520		.ASCIZ	<200>/ PC CSR STAT1 STAT2 STAT3/
(2)	010243	200	026455	026455		.ASCIZ	<200>/-----
(2)	010317	200	047510	020127		.ASCIZ	<200>/-----
(2)	010357	200	051503	020122	NUM:	.ASCIZ	<200>/HOW MANY KMC11'S TO BE TESTED?/
(2)	010375	200	042526	052103	CSR:	.ASCIZ	<200>/CSR ADDRESS?/
					VEC:	.ASCIZ	<200>/VECTOR ADDRESS?/

POWER DOWN AND UP ROUTINES

```
(2) 010416 041200 020122 051120
(2) 010455      200 044127 041511
(2) 010567      200 053523 052111
(2) 010625      200 053523 052111
(2) 010665      200 051511 052040
(2) 010725      200 047516 042040
(2) 010756 100200 046513 030503
(2) 011023      200 054105 042520
(2) 011044 024040 046513 024503
(2) 011054 046040 040517 044504
(2)
(2) 011074 000005
1774 011076      006      003
1775 011100 001276
1776 011102      006      003
1777 011104 001300
1778 011106      006      003
1779 011110 001302
1780 011112      006      003
1781 011114 001304
1782 011116      006      002
1783 011120 001306
1784
1785
1786
1787
1788 011122 000000
1789      011164
1790 011164 000000
1791      011226
1792
1793
1794
1795
1796
1797
1798 011226 022737 000176 001240
1799 011234 001075
1800 011236 132737 000001 001336
1801 011244 001071
1802 011246 022777 000007 167772
1803 011254 001404
1804 011256 022777 000207 167762
1805 011264 001061
1806 011266 010246
1807 011270 010346
1808 011272 010446
1809 011274 012737 177777 011432
1810 011302 005002
1811 011304 012704 177777
1812 011310 104401 005535
1813 011314 104417
1814 011316 011466
1815 011320 104401 005546
1816 011324 004737 011434
1817 011330 022703 000015
```

```
PRI0: .ASCIZ <200>/BR PRIORITY LEVEL? (4,5,6,7)?/
MODU: .ASCIZ <200>/WHICH LINE UNIT? IF NONE TYPE 'N', IF M8201 TYPE '1', IF M
LINE: .ASCIZ <200>/SWITCH PAC#1 (DDCMP LINE #)?/
BM: .ASCIZ <200>/SWITCH PAC#2 (BM873 BOOT ADD)?/
CONN: .ASCIZ <200>/IS THE LOOP BACK CONNECTOR ON?/
NOACT: .ASCIZ <200>/NO DEVICES ARE SELECTED/
CONERR: .ASCIZ <200><200>/KMC11 AT NONSTANDARD ADDRESS PC: /
CNERR: .ASCIZ <200>/EXPECTED FOUND/
KMCM: .ASCIZ / (KMC) /
MLDER: .ASCIZ / LOADING ERROR /
.EVEN
XSTATQ: 5
.BYTE 6,3
$TMP0
.BYTE 6,3
$TMP1
.BYTE 6,3
$TMP2
.BYTE 6,3
$TMP3
.BYTE 6,2
$TMP4
.EVEN
:BUFFERS FOR INPUT-OUTPUT
TEMP: 0
.=.+40
MDATA: 0
.=.+40
:ROUTINE USED TO CHANGE SOFTWARE SWITCH
:REGISTER USING THE CONSOLE TERMINAL
:-----
CKSWR: CMP #SWREG,SWR ;IS THE SOFT SWR BEING USED?
BNE CKSWR5 ;BR IF NO
BITB #1,$ENV ; IS IT RUNNING UNDER APT?
BNE CKSWR5 ; EXIT IF YES.
CMP #7,@$TKB ;WAS CTRL G TYPED? (7 BIT ASCII)
BEQ 1$ ;BR IF YES
CMP #207,@$TKB ;WAS CTRL G TYPED? (8 BIT ASCII)
BNE CKSWR5 ;BR IF NO
1$: MOV R2,-(SP) ;STORE R2
MOV R3,-(SP) ;STORE R3
MOV R4,-(SP) ;STORE R4
MOV #-1,SWFLG ;SET SOFT TYPE OUT FLAG
CKSWR1: CLR R2 ;CLEAR NEW SWR CONTENTS
MOV #-1,R4 ;SET FLAG TO ALL ONES
TYPE ,SMSWR ;TYPE 'SWR= '
CKSWR2: CNVRT SOFTSW ;TYPE OUT PRESENT CONTENTS
;OF SOFT SWITCH REGISTER
CKSWR3: TYPE ,SMNEW ;TYPE 'NEW? '
CKSWR4: JSR PC,INCHAR ;GET RESPONSE
CMP #15,R3 ;WAS IT A CR?
```

POWER DOWN AND UP ROUTINES

1818	011334	001424				BEQ	5\$:BR IF YES
1819	011336	022703	000012			CMP	#12,R3		:WAS IT A LF?
1820	011342	001416				BEQ	4\$:BR IF YES
1821	011344	022703	000025			CMP	#25,R3		:WAS IT CTRL U?
1822	011350	001754				BEQ	CKSWR1		:BR IF YES(START OVER)
1823	011352	022703	000007			CMP	#7,R3		:IF CNTL G GET NEXT CHAR
1824	011356	001762				BEQ	CKSWR4		
1825	011360	005004				CLR	R4		:IT MUST BE A DIGIT SO CLR FLAG
1826	011362	042703	177770			BIC	#177770,R3		:ONLY 0-7 ARE LEGAL SO MASK OFF BITS
1827	011366	006302				ASL	R2		:SHIFT R2 3 TIMES
1828	011370	006302				ASL	R2		
1829	011372	006302				ASL	R2		
1830	011374	050302				BIS	R3,R2		:ADD LAST DIGIT
1831	011376	000752				BR	CKSWR4		:GET NEXT CHARACTER
1832	011400	012766	002402	000006	4\$:	MOV	#.START,6(SP)		:LF WAS TYPED SO GO TO START
1833	011406	005704			5\$:	TST	R4		:IS FLAG CLEAR?
1834	011410	001002				BNE	6\$:IF NOT DON'T CHANGE SOFT SWR
1835	011412	010277	167622			MOV	R2,@SWR		:IF YES THEN WRITE NEW CONTENTS TO SOFT SWR
1836	011416	005037	011432		6\$:	CLR	SWFLG		:CLEAR TYPEOUT FLAG
1837	011422	012604				MOV	(SP)+,R4		:RESTORE R4
1838	011424	012603				MOV	(SP)+,R3		:RESTORE R3
1839	011426	012602				MOV	(SP)+,R2		:RESTORE R2
1840	011430	000207			CKSWR5:	RTS	PC		:RETURN
1841									
1842	011432	000000			SWFLG:	0			
1843									
1844	011434	105777	167604		INCHAR:	TSTB	@\$TKS		
1845	011440	100375				BPL	.-4		
1846	011442	017703	167600			MOV	@\$TKB,R3		
1847	011446	105777	167576			TSTB	@\$TPS		
1848	011452	100375				BPL	.-4		
1849	011454	010377	167572			MOV	R3,@\$TPB		
1850	011460	042703	000200			BIC	#BIT7,R3		
1851	011464	000207				RTS	PC		
1852									
1853	011466	000001			SOFTSW:	1			
1854	011470	006	002			.BYTE	6,2		
1855	011472	000176				SWREG			

POWER DOWN AND UP ROUTINES

```

1856
1857
1858
1859
1860
1861
1862
1863
1864
1865 011474 005737 001470
1866 011500 001004
1867 011502 104401 010725
1868 011506 000000
1869 011510 000776
1870 011512 000241
1871 011514 006137 001500
1872 011520 005537 001500
1873 011524 062737 000004 001504
1874 011532 062737 000010 001502
1875 011540 022737 002300 001502
1876 011546 001006
1877 011550 012737 002100 001502
1878 011556 012737 002302 001504
1879 011564 033737 001500 001470
1880 011572 001747
1881 011574 013700 001502
1882 011600 013702 001504
1883 011604 012037 002066
1884 011610 011037 002056
1885 011614 042737 177000 002056
1886 011622 012037 002050
1887 011626 012037 002052
1888 011632 012037 002054
1889 011636 012237 001324
1890 011642 012237 001212
1891 011646 012700 000002
1892 011652 013737 002066 002070
1893 011660 005237 002070
1894 011664 013737 002070 002072
1895 011672 005237 002072
1896 011676 013737 002072 002074
1897 011704 060037 002074
1898 011710 013737 002074 002076
1899 011716 060037 002076
1900
1901 011722 013737 002056 002060
1902 011730 060037 002060
1903 011734 013737 002060 002062
1904 011742 060037 002062
1905 011746 013737 002062 002064
1906 011754 060037 002064
1907
1908 011760 032737 000002 001446
1909 011766 001447
1910 011770
1911 011770 005737 000042
    
```

```

:
:ROUTINE USED TO "CYCLE" THROUGH UP TO 16 KMC11'S
:THIS ROUTINE SETS UP THE CONTROL ADDRESS FOR THE DIAGNOSTIC
:AND RUNS THE SPECIFIED KMC11'S. THIS ROUTINE *MUST*
:BE RUN FIRST BEFORE ENTERING THE DIAGNOSTIC FOR THE
:SETUP NECESSARY.
:
    
```

```

CYCLE: TST KMACTV ;ARE ANY KMC11'S TO BE TESTED?
        BNE 1$ ;BR IF OK.
        TYPE ,NOACT ;NO KMC11'S SELECTED!!
        HALT ;STOP THE SHOW.
        BR -2 ;DISQUALIFY CONT. SW.
1$: CLC ;CLEAR PROC. CARRY BIT.
    ROL RUN ;UPDATE POINTER
    ADC RUN ;CATCH CARRY FROM RUN
    ADD #4,MILK ;UPDATE POINTER
    ADD #10,CREAM ;UPDATE ADDRESS POINTER.
    CMP #KM.MAP+200,CREAM
    BNE 2$ ;KEEP GOING; NOT ALL TESTED FOR.
    MOV #KM.MAP,CREAM ;RESET ADDRESS POINTER.
    MOV #CNT.MAP,MILK ;RESET PASS COUNT POINTER
2$: BIT RUN,KMACTV ;IS THIS ONE ACTIVE?
    BEQ 1$ ;BR IF NO
    MOV CREAM,R0 ;GET ADDRESS POINTER
    MOV MILK,R2 ;GET PASS COUNT POINTER
    MOV (R0)+,KMCSR ;LOAD SYSTEM CTRL. REG
    MOV (R0),KMRVEC ;LOAD VECTOR
    BIC #177000,KMRVEC ;CLEAR UNWANTED BITS
    MOV (R0)+,STAT1 ;LOAD STAT1
    MCV (R0)+,STAT2 ;LOAD STAT2
    MOV (R0)+,STAT3 ;LOAD STAT3
    MOV (R2)+,$PASS ;LOAD PASS COUNT
    MOV (R2)+,$ERTTL ;LOAD ERROR COUNT
    MOV #2,R0 ;SAVE CORE THIS WAY!
    MOV KMCSR,KMCSRH
    INC KMCSRH
    MOV KMCSRH,KMCTL
    INC KMCTL
    MOV KMCTL,KMP04
    ADD R0,KMP04
    MOV KMP04,KMP06
    ADD R0,KMP06
    MOV KMRVEC,KMRLVL ;PTY LVL
    ADD R0,KMRLVL
    MOV KMRLVL,KMTVEC ;TX VEC
    ADD R0,KMTVEC
    MOV KMTVEC,KMTLVL ;TX LVL
    ADD R0,KMTLVL
4$: BIT #SW01,STRTSW ;IS TEST NO. SELECTED
    BEQ 7$ ;BR IF NO
    TST @#42 ;RUNNING IN AUTO MODE?
    
```

POWER DOWN AND UP ROUTINES

1912	011774	001044			BNE	7\$:BR IF YES
1913	011776	104401	001313		TYPE	,\$CRLF		
1914	012002	104415			INPUT			
1915	012004	007777			MTSTN			
1916	012006	000001			1			
1917	012010	001000			1000			
1918	012012	001202			\$TSTNM			
1919	012014	000			.BYTE	0		
1920	012015	001			.BYTE	1		
1921	012016	012700	013762		MOV	#TST1,R0		
1922	012022	022710			5\$: CMP	(PC)+,(R0)		:CMP FIRST WORD TO 12737
1923	012024	012737			MOV	(PC)+,@(PC)+		
1924	012026	001020			BNE	6\$:BR IF NOT SAME
1925	012030	023760	001202	000002	CMP	\$TSTNM,2(R0)		:DOES \$TSTNM MATCH?
1926	012036	001014			BNE	6\$:BR IF NO
1927	012040	022760	001202	000004	CMP	#\$TSTNM,4(R0)		:IS LAST WORD OK?
1928	012046	001010			BNE	6\$:BR IF NO
1929	012050	010037	001206		MOV	R0,\$LPADR		:IT IS A LEGAL TEST SO DO IT
1930	012054	104401	007636		TYPE	,MR		
1931	012060	042737	000002	001446	BIC	#\$SW01,STRTSW		
1932	012066	000412			BR	8\$		
1933	012070	005720			6\$: TST	(R0)+		:POP R0
1934	012072	020027	016400		CMP	R0,#TLAST+10		:AT END YET?
1935	012076	001351			BNE	5\$:BR IF NO
1936	012100	104401	001312		TYPE	,\$QUES		:YES ILLEGAL TEST NO.
1937	012104	000731			BR	4\$:TRY AGAIN
1938								
1939	012106	012737	013762	001206	7\$: MOV	#TST1,\$LPADR		:PREPARE \$LPADR ADDRESS
1940	012114	013701	002066		8\$: MOV	KMCSR,R1		:R1 = BASE KMC11 ADDRESS
1941	012120	000177	167062		JMP	@\$LPADR		:GO START TESTING.
1942								
1943								
1944								
1945								
1946								
1947								
1948								
1949								
1950								
1951								
1952	012124							
1953	012124	000005			AUTO.SIZE:	RESET		:INSURE A BUS INIT.
1954	012126	012702	002100		CSRMAP: MOV	#KM.MAP,R2		:LOAD MAP POINTER.
1955	012132	005022			1\$: CLR	(R2)+		:ZERO ENTIRE MAP
1956	012134	022702	002300		CMP	#KM.END,R2		:ALL DONE?
1957	012140	001374			BNE	1\$:BR IF NO
1958	012142	005037	001472		CLR	KMNUM		:SET OCTAL NUMBER OF KMC11'S TO 0
1959	012146	012702	002100		MOV	#KM.MAP,R2		:R2 POINTS TO KMC MAP
1960	012152	005037	001470		CLR	KMACTV		:CLEAR ACTIVE
1961	012156	032737	000001	001446	BIT	#\$SW00,STRTSW		:QUESTIONS?
1962	012164	001002			BNE	+6		:BR IF YES
1963	012166	000137	012554		JMP	7\$:IF NO SKIP QUESTIONS
1964	012172	012737	000001	001306	MOV	#1,\$TMP4		:START WITH 1
1965	012200	104415			INPUT			
1966	012202	010317			NUM			
1967	012204	000001			1			

:ROUTINE USED TO "AUTO SIZE" THE KMC11
 :CSR AND VECTOR.
 :NOTE: THE CSR MAY BE ANY WHERE IN THE FLOATING
 ADDRESS RANGE (160000:164000)
 AND THE VECTOR MAY BE ANY WHERE IN THE
 FLOATING VECTOR RANGE (300:770)

POWER DOWN AND UP ROUTINES

1968	012206	000020				16.		
1969	012210	001302				\$TMP2		
1970	012212	000				.BYTE	0	
1971	012213	001				.BYTE	1	
1972	012214	013737	001302	001472		MOV	\$TMP2,KMNUM	:KMNUM = HOW MANY
1973	012222	104401	001313		12\$:	TYPE	,\$CRLF	
1974	012226	104416				CONVRT		:TYPE WHICH KMC IS BEING DONE
1975	012230	013214				WHICH		:\$TMP4 IS WHICH KMC
1976	012232	005237	001306			INC	\$TMP4	
1977	012236	104415				INPUT		
1978	012240	010357				CSR		
1979	012242	160000				160000		
1980	012244	164000				164000		
1981	012246	001304				\$TMP3		
1982	012250	000				.BYTE	0	
1983	012251	001				.BYTE	1	
1984	012252	013722	001304			MOV	\$TMP3,(R2)+	:STORE CSR IN MAP
1985	012256	104415				INPUT		
1986	012260	010375				VEC		
1987	012262	000000				0		
1988	012264	000776				776		
1989	012266	001304				\$TMP3		
1990	012270	000				.BYTE	0	
1991	012271	001				.BYTE	1	
1992	012272	013712	001304			MOV	\$TMP3,(R2)	:STORE VECTOR IN MAP
1993	012276	104401			10\$:	TYPE		
1994	012300	010416				PRIO		:ASK WHAT BR LEVEL
1995	012302	004737	013506			JSR	PC,INTTY	:GET RESPONSE
1996	012306	022703	000024			CMP	#24,R3	:
1997	012312	101014				BHI	50\$:BR IF LESS THAN 4
1998	012314	022703	000027			CMP	#27,R3	:
1999	012320	103411				BLO	50\$:BR IF GREATER THAN 7
2000	012322	012704	000011			MOV	#11,R4	:R4 = NUMBER OF SHIFTS
2001	012326	006303				ASL	R3	:SHIFT R3 LEFT
2002	012330	005304				DEC	R4	:DEC SHIFT COUNT
2003	012332	001375				BNE	,-4	:BR IF NOT DONE
2004	012334	042703	170777			BIC	#170777,R3	:BIC UNWANTED BITS
2005	012340	050312				BIS	R3,(R2)	:PUT BR LEVEL IN STATUS MAP
2006	012342	000403				BR	8\$:CONTINUE
2007	012344	104401			50\$:	TYPE		
2008	012346	001312				\$QUES		:RESPONSE IS OUT OF LIMITS
2009	012350	000752				BR	10\$:TRY AGAIN
2010	012352				8\$:			
2011	012352				9\$:			
2012	012352	104401			16\$:	TYPE		
2013	012354	010455				MODU		:ASK WHICH LINE UNIT
2014	012356	004737	013506			JSR	PC,INTTY	:GET REPLY
2015	012362	022703	000021			CMP	#21,R3	: '1'
2016	012366	001422				BEQ	30\$:
2017	012370	022703	000022			CMP	#22,R3	: '2'
2018	012374	001412				BEQ	31\$:
2019	012376	022703	000116			CMP	#116,R3	: 'N'
2020	012402	001403				BEQ	32\$:
2021	012404	104401				TYPE		
2022	012406	001312				\$QUES		:IF NOT A 1,2 OR N TYPE '?'
2023	012410	000760				BR	16\$:TRY AGIAN

POWER DOWN AND UP ROUTINES

```

2024 012412 052722 010000          32$: BIS    #BIT12,(R2)+ ;SET BIT 12 IN STAT2 IF NO LU
2025 012416 022222                   CMP    (R2)+,(R2)+ ;POP OVER STAT2 AND STAT3
2026 012420 000450                   BR     33$
2027 012422 052712 020000          31$: BIS    #BIT13,(R2) ;SET BIT 13 IN STAT2 IF M8202
2028 012426 052762 000002 000004  BIS    #BIT1,4(R2) ;SET BIT1 IN STAT3 FOR HIGH SPEED MICRO-CODE.
2029 012434 104401                   30$: TYPE
2030 012436 010665                   CONN
2031 012440 004737 013506          JSR    PC,INTTY ;ASK IF LOOP-BACK IS ON
2032 012444 022703 000131          CMP    #131,R3 ;GET REPLY
2033 012450 001406                   BEQ    17$ ;Y
2034 012452 022703 000116          CMP    #116,R3
2035 012456 001406                   BEQ    18$ ;N
2036 012460 104401                   TYPE
2037 012462 001312                   $QUES
2038 012464 000763                   BR     30$ ;IF NOT Y OR N TYPE '?'
2039 012466 052722 040000          17$: BIS    #BIT14,(R2)+ ;TRY AGAIN
2040 012472 000402                   BR     19$ ;TURNAROUND IS CONNECTED
2041 012474 042722 040000          18$: BIC    #BIT14,(R2)+ ;NO TURNAROUND
2042 012500                   19$:
2043 012500 104415                   INPUT
2044 012502 010567                   LINE
2045 012504 000000                   0
2046 012506 000377                   377
2047 012510 001304                   $TMP3
2048 012512 000                   .BYTE 0
2049 012513 001                   .BYTE 1
2050 012514 113722 001304          MOVB  $TMP3,(R2)+ ;STORE SWITCH PAC IN MAP
2051 012520 104415                   INPUT
2052 012522 010625                   BM
2053 012524 000000                   0
2054 012526 000377                   377
2055 012530 001304                   $TMP3
2056 012532 000                   .BYTE 0
2057 012533 001                   .BYTE 1
2058 012534 113722 001304          MOVB  $TMP3,(R2)+ ;STORE SWITCH PAC IN MAP
2059 012540 005722                   TST   (R2)+ ;POP OVER STAT3
2060 012542 005337 001302          33$: DEC   $TMP2 ;DEC KMC COUNT
2061 012546 001225                   BNE   12$ ;BR IF MORE TO DO
2062 012550 000137 013114          JMP   13$ ;CONTINUE
2063 012554 012701 160000          7$: MOV   #160000,R1 ;SET FOR FIRST ADDRESS TO BE TESTED
2064 012560 012737 013206 000004  MOV   #6$,@#4 ;SET FOR NON-EXISTANT DEVICE TIME OUT
2065 012566 005011                   2$: CLR   (R1) ;CLEAR SEL0
2066 012570 005711                   TST  (R1) ;IF KMC11 KMCSR S/B 0
2067 012572 001140                   BNE  3$ ;IF NO DEV ; TRAP TO 4. IF NO BIT 8 THEN NO KMC1
2068 012574 005061 000006          CLR   6(R1) ;CLEAR SEL6
2069 012600 005761 000006          TST  6(R1) ;IF KMC11 THEN KMRIC S/B =0!
2070 012604 001133                   BNE  3$ ;BR IF NOT KMC11
2071 012606 012711 002000          MOV   #BIT10,(R1) ;SET ROM0
2072 012612 005061 000004          CLR   4(R1) ;CLEAR SEL4
2073 012616 012761 125252 000006  MOV   #125252,6(R1) ;WRITE THIS TO SEL6
2074 012624 052711 020000          BIS   #BIT13,(R1) ;WRITE IT!
2075 012630 022761 125252 000004  CMP   #125252,4(R1) ;WAS IT WRITTEN?
2076 012636 001116                   BNE  3$ ;IF NO IT IS NOT CRAM
2077                                     ;AT THIS POINT IT IS ASSUMED THAT R1 HOLDS A KMC11 CSR ADDRESS.
2078 012640                   21$:
2079 012640 010122                   22$: MOV   R1,(R2)+ ;STORE CSR IN CORE TABLE.

```


POWER DOWN AND UP ROUTINES

2080	012642	012711	001000		15\$:	MOV	#BIT9,(R1)	:CLEAR LINE UNIT LOOP
2081	012646	005061	000004			CLR	4(R1)	:CLEAR PORT4
2082	012652	012761	122113	000006		MOV	#122113,6(R1)	:LOAD INSTRUCTION (CLR DTR)
2083	012660	052711	000400			BIS	#BIT8,(R1)	:CLOCK INSTRUCTION
2084	012664	012761	021264	000006		MOV	#021264,6(R1)	:LOAD INSTRUCTION
2085	012672	052711	000400			BIS	#BIT8,(R1)	:CLOCK INSTRUCTION
2086	012676	122761	000377	000004		CMPB	#377,4(R1)	:IS IT ALL ONES?
2087	012704	001003				BNE	.+10	:BR IF NO
2088	012706	052712	010000			BIS	#BIT12,(R2)	:IF YES, NO LINE UNIT, SET STATUS BIT
2089	012712	000441				BR	20\$	
2090	012714	032761	000002	000004		BIT	#BIT1,4(R1)	:IS SWITCH A ONE?
2091	012722	001406				BEQ	.+16	:BR IF M8201
2092	012724	052712	060000			BIS	#BIT13!BIT14,(R2)	:M8202 ASSUME CONNECTOR
2093	012730	052762	000002	000004		BIS	#BIT1,4(R2)	:SET BIT1 IN STAT3 FOR HIGH SPEED MICRO-CODE...
2094	012736	000427				BR	20\$:CONNECTOR ON)
2095	012740	032761	000010	000004		BIT	#BIT3,4(R1)	:IS MRDY SET
2096	012746	001023				BNE	20\$:BR IF M8201 NO CONNECTOR (ON LINE)
2097	012750	012761	000100	000004		MOV	#BIT6,4(R1)	:LOAD PORT4
2098	012756	012761	122113	000006		MOV	#122113,6(R1)	:LOAD INSTRUCTION
2099	012764	052711	000400			BIS	#BIT8,(R1)	:CLOCK INSTRUCTION(SET DTR)
2100	012770	012761	021264	000006		MOV	#021264,6(R1)	:LOAD INSTRUCTION
2101	012776	052711	000400			BIS	#BIT8,(R1)	:CLOCK INSTRUCTION(READ MODEM REG)
2102	013002	032761	000010	000004		BIT	#BIT3,4(R1)	:IS MRDY SET NOW?
2103	013010	001402				BEQ	20\$:BR IF NO CONNECTOR
2104	013012	052712	040000			BIS	#BIT14,(R2)	:SET STATUS BIT FOR CONNECTOR
2105	013016	005722			20\$:	TST	(R2)+	:POP POINTER
2106	013020	012761	021324	000006		MOV	#021324,6(R1)	:PUT INSTRUCTION IN PORT6
2107	013026	012711	001400			MOV	#BIT9!BIT8,(R1)	:PORT4_LU 15
2108	013032	156122	000004			BISB	4(R1),(R2)+	:STORE_DDCMP LINE # IN TABLE
2109	013036	012761	021344	000006		MOV	#021344,6(R1)	:PORT6_INSTRUCTION
2110	013044	012711	001400			MOV	#BIT8!BIT9,(R1)	:CLOCK_INSTR.
2111	013050	156122	000004			BISB	4(R1),(R2)+	:STORE BM873 ADD IN TABLE
2112	013054	005722				TST	(R2)+	:POP OVER STAT3
2113	013056	005011				CLR	(R1)	:CLEAR ROMI
2114	013060	005237	001472			INC	KMNUM	:UPDATE DEVICE COUNTER
2115	013064	022737	000020	001472		CMP	#20,KMNUM	:ARE MAX. NO. OF DEV FOUND?
2116	013072	001410				BEQ	13\$:YES DON'T LOOK FOR ANY MORE.
2117	013074	005011			3\$:	CLR	(R1)	:CLEAR BIT 10
2118	013076	005061	000006			CLR	6(R1)	:CLEAR SEL 6
2119	013102	062701	000010		14\$:	ADD	#10,R1	:UPDATE CSR POINTER ADDRESS
2120	013106	022701	164000			CMP	#164000,R1	
2121	013112	001225				BNE	2\$:BR IF MORE ADDRESS TO CHECK.
2122	013114	005037	001470		13\$:	CLR	KMACTV	
2123	013120	005737	001472			TST	KMNUM	:WERE ANY KMC11'S FOUND AT ALL?
2124	013124	001423				BEQ	5\$:ERROR AUTO SIZER FOUND NO KMC11'S IN THIS SYS.
2125	013126	013701	001472			MOV	KMNUM,R1	
2126	013132	010137	001476			MOV	R1,SAVNUM	:SAVE NUMBER OF DEVICES
2127	013136	000241			4\$:	CLC		
2128	013140	006137	001470			ROL	KMACTV	:GENERATE ACTIVE REGISTER OF DEVICES.
2129	013144	005237	001470			INC	KMACTV	:SET THE BIT
2130	013150	005301				DEC	R1	
2131	013152	001371				BNE	4\$:BR IF MORE TO GENERATE
2132	013154	012737	000006	000004		MOV	#6,@#4	:RESTORE TRAP VECTOR
2133	013162	013737	001470	001474		MOV	KMACTV,SAVACT	:SAVE ACTIVE REGISTER
2134	013170	000137	013222			JMP	VECMAP	:GO FIND THE VECTOR NOW.
2135	013174	104401	007641		5\$:	TYPE	,MERR2	:NOTIFY OPR THAT NO KMC11'S FOUND.

POWER DOWN AND UP ROUTINES

2136	013200	005000			CLR	R0		:MAKE DATA LIGHTS ZERO
2137	013202	000000			HALT			:STOP THE SHOW
2138	013204	000776			BR	.-2		:DISABLE CONT. SW.
2139	013206	012716	013102		6\$: MOV	#14\$, (SP)		:ENTERED BY NON-EXISTANT TIME-OUT.
2140	013212	000002			RTI			:RETURN TO MAINSTREAM
2141								
2142	013214	000001			WHICH:	1		
2143	013216	002	002		.BYTE	2,2		
2144	013220	001306			\$TMP4			
2145								
2146	013222	032737	000001	001446	VECMAP:	BIT	#SW00,STRTSW	
2147	013230	001114			BNE	5\$		
2148	013232	012737	000340	000022	MOV	#340,@#22		:SET IOT TRAP PRIO TO 7
2149	013240	012737	013414	000020	MOV	#4\$,@#20		:SET IOT TRAP VECTOR
2150	013246	012702	002100		MOV	#KM.MAP,R2		:SET SOFTWARE POINTER
2151	013252	012700	000300		MOV	#300,R0		:FLOATING VECTORS START HERE.
2152	013256	012701	000302		MOV	#302,R1		:PC OF IOT INSTR.
2153	013262	010120			1\$: MOV	R1,(R0)+		:START FILLING VECTOR AREA
2154	013264	012721	000004		MOV	#4,(R1)+		:WITH +2; IOT
2155	013270	022021			CMP	(R0)+,(R1)+		:ADD 2 TO R0 +R1
2156	013272	020127	001000		CMP	R1,#1000		
2157	013276	101771			BLOS	1\$:BR IF MORE TO FILL
2158	013300	013737	001470	001276	MOV	KMACTV,\$TMP0		:STORE TEMPORALLY
2159	013306	006037	001276		2\$: ROR	\$TMP0		:BRING OUT A BIT
2160	013312	103063			BCC	5\$:BR IF ALL DONE
2161	013314	012704	000012		MOV	#12,R4		:R4 IS INDEX REGISTER
2162	013320	016437	013472	177776	MOV	BRLVL(R4),PS		:SET PS TO 7
2163	013326	011201			MOV	(R2),R1		
2164	013330	012761	000200	000004	MOV	#200,4(R1)		
2165	013336	012711	001000		MOV	#BIT9,(R1)		:SET ROMI
2166	013342	012761	121111	000006	MOV	#121111,6(R1)		:PUT INSTRUCTION IN PORT6
2167	013350	012711	001400		MV	#BIT9!BIT8,(R1)		:FORCE AN INTERRUPT
2168	013354	105200			7\$: INCB	R0		:STALL
2169	013356	001376			BNE	.-2		:FOR TIME TO INTERUPT
2170	013360	162704	000002		SUB	#2,R4		:GET NEXT LOWEST PS LEVEL
2171	013364	001404			BEQ	6\$:BR IF R4 = 0
2172	013366	016437	013472	177776	MOV	BRLVL(R4),PS		:MOVE NEXT LOWER LEVEL IN PS
2173	013374	000767			BR	7\$:BR TO DELAY
2174	013376	052762	005300	000002	6\$: BIS	#5300,2(R2)		:NO INTERUPT ASSUME 300 AT LEVEL 5 AND FIX KMC11
2175	013404	005011			3\$: CLR	(R1)		:CLEAR ROMI
2176	013406	062702	000010		ADD	#10,R2		:POP SOFTWARE POINTER
2177	013412	000735			BR	2\$:KEEP GOING
2178	013414	051662	000002		4\$: BIS	(SP),2(R2)		:GET VECTOR ADDRESS
2179	013420	042762	000007	000002	BIC	#7,2(R2)		:CLEAR JUNK
2180	013426	016405	013474		MOV	BRLVL+2(R4),R5		:GET BR LEVEL OF KMC11
2181	013432	006305			ASL	R5		:SHIFT LEVEL 4 PLACES
2182	013434	006305			ASL	R5		:TO THE LEFT FOR THE
2183	013436	006305			ASL	R5		:STATUS TABLE
2184	013440	006305			ASL	R5		
2185	013442	042705	170777		BIC	#170777,R5		:CLEAR UNWANTED BITS
2186	013446	050562	000002		BIS	R5,2(R2)		:PUT BR LEVEL IN STATUS TABLE
2187	013452	022626			CMP	(SP)+,(SP)+		:POP IOT JUNK OFF STACK
2188	013454	012716	013404		MOV	#3\$, (SP)		:SET FOR RETURN
2189	013460	000002			RTI			
2190	013462	012737	004134	000020	5\$: MOV	#\$SCOPE,@#20		: RESTORE SCOPE VECTOR
2191	013470	000207			RTS	PC		:ALL DONE WITH "AUTO SIZING"

POWER DOWN AND UP ROUTINES

2192					
2193	013472	000000			
2194	013474	000000			
2195	013476	000200			
2196	013500	000240			
2197	013502	000300			
2198	013504	000340			
2199					
2200					
2201	013506	105777	165532		
2202	013512	100375			
2203	013514	017703	165526		
2204	013520	105777	165524		
2205	013524	100375			
2206	013526	010377	165520		
2207	013532	042703	000240		
2208	013536	000207			
2209					
2210	013540				
2211	013540	000005			
2212	013542	010046			
2213	013544	010146			
2214	013546	010246			
2215	013550	010346			
2216	013552	005037	013754		
2217	013556	005037	013760		
2218	013562	013700	001376		
2219	013566	010037	001476		
2220	013572	012701	001346		
2221	013576	013737	001372	013756	
2222	013604	113737	001366	013754	
2223	013612	113737	001367	013760	
2224	013620	013737	001374	001470	
2225	013626	013737	001470	001474	
2226	013634	012702	001402		
2227	013640	012703	002100		
2228	013644	005023			
2229	013646	022703	002300		
2230	013652	003374			
2231	013654	012703	002100		
2232	013660	013723	013756		
2233	013664	112163	000001		
2234	013670	006213			
2235	013672	006213			
2236	013674	053713	013760		
2237	013700	006313			
2238	013702	006313			
2239	013704	006313			
2240	013706	006313			
2241	013710	053723	013754		
2242	013714	012223			
2243	013716	005723			
2244	013720	005300			
2245	013722	001407			
2246	013724	062737	000010	013756	
2247	013732	062737	000010	013754	

BRLVL:	PRO	:LEVEL 0	
	PRO	:LEVEL 0	
	PR4	:LEVEL 4	
	PR5	:LEVEL 5	
	PR6	:LEVEL 6	
	PR7	:LEVEL 7	

INTTY:	TSTB	@\$TKS	:WAIT FOR DONE
	BPL	.-4	
	MOV	@\$TKB,R3	:PUT CHAR IN R3
	TSTB	@\$TPS	:WAIT UNTIL PRINTER IS READY
	BPL	.-4	
	MOV	R3,@\$TPB	:ECHO CHAR
	BIC	#BIT7!BIT5,R3	:MASK OFF LOWER CASE
	RTS	PC	:RETURN

APT.SIZE:	RESET		
	MOV	R0,-(SP)	:::PUSH R0 ON STACK
	MOV	R1,-(SP)	:::PUSH R1 ON STACK
	MOV	R2,-(SP)	:::PUSH R2 ON STACK
	MOV	R3,-(SP)	:::PUSH R3 ON STACK
	CLR	VECTR	: CLEAR THE LOCAL VARIABLE
	CLR	PRIORITY	: CLEAN UP LOCAL VARIABLE
	MOV	\$CDW1,R0	: GET THE DEVICE COUNT
	MOV	R0,SAVNUM	: SAVE THE NO. OF DEVICES
	MOV	#\$MAMS1,R1	: GET EXTRA INFO, BITS POINTER
	MOV	\$BASE,BASE	: GET BASE CSR ADDRESS
	MOVB	\$VECT1,VECTR	: GET THE VECTOR
	MCVB	\$VECT1+1,PRIORITY	: GET THE PRIORITY
	MOV	\$DEVM,KMACTV	: SAVE THE KMC'S SELECTED ACTIVE
	MOV	KMACTV,SAVACT	: SAVE THE ACTIVE REGISTER
	MOV	#\$DDW0,R2	: GET ADDRESS OF FIRST DEVICE DESCRIPTOR WORD
	MOV	#KM.MAP,R3	: GET POINTER TO DEVICE MAP
3\$:	CLR	(R3)+	: CLEAR DEVICE MAP
	CMP	#KM.END,R3	: IS WHOLE DEV.MAP CLEARED?
	BGT	3\$: NO, THEN GO ON.
	MOV	#KM.MAP,R3	: RESTORE DEV.MAP POINTER.
1\$:	MOV	BASE,(R3)+	: LOAD CSR ADDRESS
	MOVB	(R1)+,1(R3)	: GET EXTRA INFO. BITS
	ASR	(R3)	: SET IT IN RIGHT POSITION.
	ASR	(R3)	: SET IT IN RIGHT POSITION.
	BIS	PRIORITY,(R3)	: GET PRIORITY IN STAT1
	ASL	(R3)	: SET THEM IN RIGHT POSITION
	ASL	(R3)	:
	ASL	(R3)	:
	ASL	(R3)	:
	BIS	VECTR,(R3)+	: GET THE VECTOR IN STAT1.
	MOV	(R2)+,(R3)+	: GET THE STAT2 FROM DDWXX
	TST	(R3)+	: SKIP OVER STAT3
	DEC	R0	: COUNT BY 1
	BEQ	2\$: ALL DONE?
	ADD	#10,BASE	: INCREMENT BASE CSR ADDRESS BY 10
	ADD	#10,VECTR	: INCREMENT VECTOR ADDRESS BY 10

POWER DOWN AND UP ROUTINES

```
2248 013740 000747  
2249 013742  
2250 013742 012603  
2251 013744 012602  
2252 013746 012601  
2253 013750 012600  
2254 013752 000207  
2255 013754 000000  
2256 013756 000000  
2257 013760 000000
```

2\$: BR 1\$; SET THE NEXT MAP ENTRY

MOV (SP)+,R3 ;:POP STACK INTO R3
MOV (SP)+,R2 ;:POP STACK INTO R2
MOV (SP)+,R1 ;:POP STACK INTO R1
MOV (SP)+,R0 ;:POP STACK INTO R0
RTS PC ; RETURN

VECTR: .WORD 0
BASE: .WORD 0
PRIRTY: .WORD 0

FREE RUNNING TESTS

2258
 2259
 2260
 2261
 2262
 2263
 2264
 2265
 2266
 2267
 2268
 2269
 2270
 2271
 2272
 2273
 2274
 2275
 2276
 2277
 2278
 2279
 2280
 2281
 2282
 2283
 2284
 2285
 2286
 2287
 2288
 2289
 2290
 2291
 2292
 2293
 2294
 2295
 2296
 2297
 2298
 2299
 2300
 2301
 2302
 2303
 2304
 2305
 2306
 2307
 2308
 2309
 2310
 2311
 2312
 2313

013762	000004			
013764	012737	000001	001202	
013772	012737	015006	001442	
014000	004737	022474		
014004	013700	021360		
014010	062700	000002		
014014	012702	021362		
014020	105022			
014022	005300			
014024	001375			
014026	005037	021306		
014032	005037	021310		
014036	012711	040000		
014042	012711	100000		
014046	105227	000000		
014052	001375			
014054	005037	011122		
014060	005711			
014062	100405			
014064	005237	011122		
014070	001373			
014072	104014			
014074	000771			
014076	052711	004043		
014102	005037	011122		
014106	105711			
014110	100404			
014112	005237	011122		
014116	001373			
014120	104014			
014122	012761	021430	000004	
014130	005061	000006		
014134	142711	000040		
014140	005037	011122		
014144	105711			
014146	100020			

```

:***** TEST 1 *****
:*FREE RUNNING FLAG MODE DATA TEST
:*TRANSMIT A MESSAGE AND VERIFY THE RECEIVED DATA
:*LINE UNIT LOOP IS SET FOR THIS TEST.
:*ALL FOLLOWING TESTS ARE FREE RUNNING AND ARE PERFORMED
:*ONLY ON KMC'S WITH LINE UNITS. IF YOU WISH TO PERFORM
:*THESE FREE RUNNING TESTS ON A KMC (NORMALLY THE FREE RUNNING TESTS
:*WILL RUN BY LOADING AND STARTING DZKCG
:* WITH SWITCH 7 = 1
:*****

: TEST 1
:-----
:*****
TST1: SCOPE
      MOV #1,$STSTM ; LOAD THE NO. OF THIS TEST
      MOV #TST2,NEXT ; POINT TO THE START OF NEXT TEST.
;
; $SKIPT 14$ ;R1 CONTAINS BASE KMC11 ADDRESS
      JSR PC,LDRVRF ;FIRST TEST LOAD & VERIFY.
      MOV RCOUNT,R0 ;CLEAR RECEIVER BUFFER
      ADD #2,R0 ;CLEAR 2 MORE LOCATIONS
      MOV #RBUF,R2 ;CLEAR OUT RECEIVE BUFFER
10$: CLR (R2)+ ;CLEAR BUFFER
      DEC R0 ;DONE YET!
      BNE 10$ ;NO
      CLR TFLAG ;SET TFLAG TO 0
      CLR RFLAG ;SET RFLAG TO 0
      MOV #BIT14,(R1) ;MASTER CLEAR
; BIT #BIT15,STAT1 ;CRAM?
; BEQ .+6 ;BR IF NO
      MOV #BIT15,(R1) ;IF CRAM SET RUN
      INCB #0 ;DELAY
      BNE .-4 ;DELAY
1$: CLR TEMP ;GET SET TO DELAY
      TST (R1) ;RUN SET?
      BMI .+14 ;BR IF YES
      INC TEMP ;INC DELAY
      BNE 1$ ;BR IF NOT DONE
      ERROR 14 ;ERROR RUN NOT SET
      BR 1$ ;TRY AGAIN
      BIS #4043,(R1) ;BASEMC I, LU LOOP
      CLR TEMP ;GET SET TO DELAY
2$: TSTB (R1) ;RDI SET?
      BMI .+12 ;BR IF YES
      INC TEMP ;INC DELAY
      BNE 2$ ;BR IF NOT DONE
      ERROR 14 ;ERROR,RDI NOT SET
      MOV #BASEMC,4(R1) ;SET UP BASEMC ADDRESS
      CLR 6(R1) ;CLEAR COUNT
      BICB #40,(R1) ;CLEAR RQI
      CLR TEMP ;GET SET TO DELAY
3$: TSTB (R1) ;IS RDI GONE?
      BPL 8$ ;BR IF YES
  
```


FREE RUNNING TESTS

2314	014150	005237	011122		INC	TEMP	:INC DELAY
2315	014154	001373			BNE	3\$:BR IF NOT DONE
2316	014156	105761	000002		TSTB	2(R1)	:IS THERE A CNTL 0 ERROR
2317	014162	100011			BPL	18\$:BR IF NO
2318	014164	016137	000004	001302	MOV	4(R1), \$TMP2	:SAVE SEL4 FOR TYPEOUT
2319	014172	016137	000006	001304	MOV	6(R1), \$TMP3	:SAVE SEL6 FOR TYPEOUT
2320	014200	104016			ERROR	16	:CNTL 0 ERROR
2321	014202	000137	015006		JMP	14\$:FATAL ERROR STOP
2322	014206	104014			ERROR	14	:ERROR RDI STILL SET
2323	014210						
2324	014210	152711	000041		BISB	#41, (R1)	:ASK FOR CNTL I
2325	014214	105711			TSTB	(R1)	:WAIT FOR RDI
2326	014216	100376			BPL	64\$:BR IF NOT SETY
2327	014220	005061	000006		CLR	6(R1)	:SET FULL DUPLEX
2328	014224	142711	000040		BICB	#40, (R1)	:CLEAR RQI
2329	014230	105711			TSTB	(R1)	:RDI UP?
2330	014232	100776			BMI	65\$:BR IF YES
2331	014234	152711	000044		BISB	#44, (R1)	:REC BA/CC
2332	014240	005037	011122		CLR	TEMP	:GET SET TO DELAY
2333	014244	105711			TSTB	(R1)	:IS RDI SET?
2334	014246	100404			BMI	.+12	:BR IF YES
2335	014250	005237	011122		INC	TEMP	:INC DELAY
2336	014254	001373			BNE	4\$:BR IF DELAY NOT DONE
2337	014256	104014			ERROR	14	:ERROR RDI NOT SET
2338	014260	012761	021362	000004	MOV	#RBUF, 4(R1)	:LOAD REC BA
2339	014266	013761	021360	000006	MOV	RCOUNT, 6(R1)	:LOAD REC COUNT
2340	014274	142711	000040		BICB	#40, (R1)	:CLEAR RQI
2341	014300	005037	011122		CLR	TEMP	:GET SET TO DELAY
2342	014304	105711			TSTB	(R1)	:RDI GONE?
2343	014306	100004			BPL	.+12	:BR IF YES
2344	014310	005237	011122		INC	TEMP	:INC DELAY
2345	014314	001373			BNE	5\$:BR IF NO DONE
2346	014316	104014			ERROR	14	:ERROR RDI STILL SET
2347	014320	152711	000040		BISB	#40, (R1)	:XMIT BA/CC
2348	014324	005037	011122		CLR	TEMP	:GET SET TO DELAY
2349	014330	105711			TSTB	(R1)	:RDI SET?
2350	014332	100404			BMI	.+12	:BR IF YES
2351	014334	005237	011122		INC	TEMP	:INC DELAY
2352	014340	001373			BNE	6\$:BR IF NOT DONE
2353	014342	104014			ERROR	14	:ERROR RDI NOT SET
2354	014344	012761	021314	000004	MOV	#TBUF, 4(R1)	:LOAD XMIT BUFFER
2355	014352	013761	021312	000006	MOV	TCOUNT, 6(R1)	:LOAD COUNT
2356	014360	142711	000040		BICB	#40, (R1)	:CLEAR RQI
2357	014364	005037	011122		CLR	TEMP	:GET SET TO DELAY
2358	014370	105711			TSTB	(R1)	:RDI GONE?
2359	014372	100004			BPL	.+12	:BR IF YES
2360	014374	005237	011122		INC	TEMP	:INC DELAY
2361	014400	001373			BNE	7\$:BR IF NOT DONE DELAY
2362	014402	104014			ERROR	14	:ERROR RDI STILL SET
2363	014404	005037	011122		CLR	TEMP	:GET SET TO DELAY
2364	014410	012737	000022	001276	MOV	#22, \$TMP0	:GET SET FOR LONG DELAY
2365	014416	105761	000002		TSTB	2(R1)	:RDI SET?
2366	014422	100407			BMI	17\$:BR IF YES
2367	014424	005237	011122		INC	TEMP	:INC DELAY
2368	014430	001372			BNE	11\$:BR IF DELAY NOT DONE
2369	014432	005337	001276		DEC	\$TMP0	:DEC DELAY COUNT

FREE RUNNING TESTS

2370	014436	001367				BNE	11\$:BR IF NOT DONE DELAY
2371	014440	104014				ERROR	14		:ERROR RDO NOT SET
2372	014442	016137	000002	001300	17\$:	MOV	2(R1), \$TMP1		:SAVE SEL2
2373	014450	001001				BNE	.+4		:BR IF OK
2374	014452	104014				ERROR	14		:ERROR!!! SEL2 = 0!!!!!!
2375	014454	032761	000004	000002		BIT	#BIT2,2(R1)		:REC OR XMIT?
2376	014462	001032				BNE	13\$:BR IF REC
2377	014464	005737	021306		12\$:	TST	TFLAG		:FIRST TIME HERE?
2378	014470	001401				BEQ	.+4		:BR IF YES
2379	014472	104014				ERROR	14		:ERROR MULTIPLE XMIT DONES
2380	014474	012737	177777	021306		MOV	#-1, TFLAG		:SET TFLAG TO -1
2381	014502	132761	000001	000002		BITB	#BIT0,2(R1)		:IS IT CONTROL 0
2382	014510	001401				BEQ	.+4		:BR IF NO
2383	014512	104014				ERROR	14		:XMIT ERROR
2384	014514	022761	021314	000004		CMP	#TBUF,4(R1)		:XMIT 3A CORRECT?
2385	014522	001401				BEQ	.+4		:BR IF YES
2386	014524	104014				ERROR	14		:XMIT BA ERROR
2387	014526	023761	021312	000006		CMP	TCOUNT,6(R1)		:COUNT OK?
2388	014534	001401				BEQ	.+4		:BR IF YES
2389	014536	104014				ERROR	14		:XMIT COUNT ERROR
2390	014540	142761	000207	000002		BICB	#207,2(R1)		:CLEAR RDO AND BITS 0-2
2391	014546	000453				BR	15\$:CONTINUE
2392	014550	005737	021310		13\$:	TST	RFLAG		:FIRST TIME HERE?
2393	014554	001401				BEQ	.+4		:BR IF YES
2394	014556	104014				ERROR	14		:ERROR MULTIPLE REC DONES
2395	014560	012737	177777	021310		MOV	#-1, RFLAG		:SET RFLAG TO -1
2396	014566	132761	000001	000002		BITB	#BIT0,2(R1)		:IS IT CNTL 0
2397	014574	001401				BEQ	.+4		:BR IF NO
2398	014576	104014				ERROR	14		:RECEIVE ERROR
2399	014600	022761	021362	000004		CMP	#RBUF,4(R1)		:REC BA CORRECT?
2400	014606	001401				BEQ	.+4		:BR IF YES
2401	014610	104014				ERROR	14		:REC BA ERROR
2402	014612	023761	021360	000006		CMP	RCOUNT,6(R1)		:COUNT OK?
2403	014620	001401				BEQ	.+4		:BR IF YES
2404	014622	104014				ERROR	14		:REC COUNT ERROR
2405	014624	013700	021360			MOV	RCOUNT,R0		:GET SET TO CHECK DATA
2406	014630	012702	021314			MOV	#TBUF,R2		:R2 POINTS TO GOOD DATA
2407	014634	012703	021362			MOV	#RBUF,R3		:R3 POINTS TO RECEIVE DATA
2408	014640	010337	001302		9\$:	MOV	R3, \$TMP2		:SAVE ADDRESS FOR TYPEOUT
2409	014644	112205				MOVB	(R2)+,R5		:R5 = XMIT DATA
2410	014646	112304				MOVB	(R3)+,R4		:R4 = RECEIVE DATA
2411	014650	120504				CMPB	R5,R4		:CHECK DATA
2412	014652	001401				BEQ	.+4		:BR IF OK
2413	014654	104013				ERROR	13		:DATA ERROR
2414	014656	005300				DEC	R0		:DEC COUNT
2415	014660	001367				BNE	9\$:BR IF NOT DONE
2416	014662	005713				TST	(R3)		:THIS SHOULD BE 0, ELSE
2417	014664	001401				BEQ	.+4		:IT RECEIVED TO MUCH!!
2418	014666	104014				ERROR	14		:ERROR
2419	014670	142761	000207	000002		BICB	#207,2(R1)		:CLEAR RDO AND BITS 0-2
2420	014676	005737	021310		15\$:	TST	RFLAG		:REC DONE?
2421	014702	001640				BEQ	16\$:BR IF NO
2422	014704	005737	021306			TST	TFLAG		:XMIT DONE?
2423	014710	001635				BEQ	16\$:BR IF NO
2424	014712	004737	022442			JSR	PC, SHUTDOWN		:SHUTDOWN KMC
2425	014716	012700	014744			MOV	#25\$,R0		:POINTER TO EXPECTED SOFT COUNTS

FREE RUNNING TESTS

2426 014722 012701 021433
2427 014726 012702 000010
2428 014732 122021
2429 014734 001007
2430 014736 005302
2431 014740 001374
2432 014742 000421
2433 014744 000 000 000
2434 014747 000 000 000
2435 014752 000 000
2436 014754 113737 021433 001300
2437 014762 113737 021435 001302
2438 014770 113737 021437 001304
2439 014776 113737 021441 001306
2440 015004 104017
2441 015006
2442 015006
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454 015006 000004
2455 015010 012737 000002 001202
2456 015016 012737 015200 001442
2457
2458
2459 015024 004737 022030
2460 015030 004537 022410
2461 015034 021314
2462 015036 000044
2463 015040 012700 000010
2464 015044 012703 000015
2465 015050 005037 011122
2466 015054 105761 000002
2467 015060 100407
2468 015062 005237 011122
2469 015066 001372
2470 015070 005303
2471 015072 001370
2472 015074 104014
2473 015076 000431
2474 015100 132761 000001 000002
2475 015106 001002
2476 015110 104014
2477 015112 000423
2478 015114 012705 000004
2479 015120 016104 000006
2480 015124 020504
2481 015126 001404

21\$: MOV #BASEMC+3,R1 ; POINTER TO ACTUAL COUNTS
MOV #10,R2 ; COUNT
22\$: CMPB (R0)+,(R1)+ ; COMPARE SOFT ERROR COUNTS
BNE 23\$; IF ERROR BR 23\$
DEC R2 ; DEC COUNT
BNE 22\$; CONTINUE CHECKING IF NOT DONE
BR 24\$; ALL COUNTS OK, GET OUT
25\$: .BYTE 0,0,0,0,0,0,0,0 ; EXPECTED ERROR COUNTS

23\$: MOVB BASEMC+3,\$TMP1
MOVB BASEMC+5,\$TMP2
MOVB BASEMC+7,\$TMP3
MOVB BASEMC+11,\$TMP4
ERROR 17
24\$:
14\$: ; SCOPE ; SCOPE THIS TEST

:***** TEST 2 *****
:*OVERUN TEST
:*IN FREE RUNNING MODE SEND MESSAGE WITH NO RECEIVE
:*BUFFER AVAILABLE, VERIFY THAT AN OVERRUN ERROR OCCURS
:*****
:
: TEST 2
:-----
:*****
TST2: SCOPE ; LOAD THE NO. OF THIS TEST
MOV #2,\$TSTNM ; POINT TO THE START OF NEXT TEST.
MOV #TST3,NEXT ; R1 CONTAINS BASE KMC11 ADDRESS
:
: \$SKIPT 10\$; LOAD KMC BASEMC ADDRESS
JSR PC,BASELD ; LOAD XMIT BA/CC
JSR R5,XFRELD ; BA
TBUF ; CC
44 ; RO = RETRANSMISSION COUNT
MOV #10,R0 ; DELAY COUNT
MOV #15,R3 ; CLEAR DELAY COUNTER
CLR TEMP ; IS RDY 0 SET?
1\$: TSTB 2(R1) ; BR IF SET
BMI .+20 ; INC DELAY COUNTER
INC TEMP ; BR IF NOT DONE DELAY
BNE 1\$; DEC DELAY COUNT
DEC R3 ; BR IF DELAY NOT DONE
BNE 1\$; ERROR, RDY 0 NOT SET
ERROR 14 ; GET OUT
BR 10\$; IS IT CNTL 0?
BITB #BIT0,2(R1) ; BR IF YES
BNE 11\$; ERROR, NOT CNTL 0
ERROR 14 ; CONTINUE
BR 10\$; PUT 'EXPECTED' IN R5
11\$: MOV #BIT2,R5 ; PUT 'FOUND' IN R4
MOV 6(R1),R4 ; IS ORUN SET?
CMP R5,R4 ; BR IF YES
BEQ 12\$

FREE RUNNING TESTS

```
2482 015130 022704 000001          CMP      #1,R4          ;DATA CK ERROR?
2483 015134 001413          BEQ      13$           ;BR IF YES
2484 015136 104015          ERROR   15           ;ERROR, ORUN NOT SET
2485 015140 042761 000207 000002    12$:    BIC      #207,2(R1)  ;CLEAR RDO
2486 015146 005037 011122          CLR      TEMP        ;RESET DELAY
2487 015152 005300          DEC      R0          ;DEC RETRANS COUNT
2488 015154 001337          BNE     1$           ;CONTINUE
2489 015156 004737 022442          JSR     PC,SHUTDOWN  ;SHUTDOWN KMC
2490 015162 104420          10$:    ADVANCE          ;SCOPE THIS TEST
2491 015164 042761 000207 000002    13$:    BIC      #207,2(R1)  ;IGNOR THIS ERROR
2492 015172 005037 011122          CLR      TEMP        ;RESET DELAY
2493 015176 000726          BR      1$           ;CONTINUE
2494
2495
2496          ;***** TEST 3 *****
2497          ;*LOST DATA TEST
2498          ;*IN FREE RUNNING MODE SEND A MESSAGE LONGER THAN THE RECEIVE
2499          ;*BUFFER, VERIFY THAT A LOST DATA ERROR OCCURS.
2500          ;*****
2501
2502          ; TEST 3
2503          ;-----
2504          ;*****
2505 015200 000004          TST3:  SCOPE
2506 015202 012737 000003 001202      MOV      #3,$STSTM    ; LOAD THE NO. OF THIS TEST
2507 015210 012737 015336 001442      MOV      #TST4,NEXT  ; POINT TO THE START OF NEXT TEST.
2508          ;R1 CONTAINS BASE KMC11 ADDRESS
2509 015216 104410          :        MSTCLR      ;MASTER CLEAR KMC11
2510          :
2511 015220 004737 022030          :        $SKIPT     10$
2512 015224 004537 022356          :        JSR      PC,BASELD ;LOAD KMC BASEMC ADDRESS
2513 015230 021362          :        JSR      R5,RFRELD ;LOAD RECEIVE BA/CC
2514 015232 000020          :        RBUF
2515 015234 004537 022410          :        20          ;BA
2516 015240 021314          :        JSR      R5,XFRELD ;CC
2517 015242 000044          :        TBUF      ;LOAD XMIT BA/CC
2518 015244 012703 000015          :        44          ;BA
2519 015250 005037 011122          :        MOV      #15,R3  ;CC
2520 015254 105761 000002          1$:    CLR      TEMP        ;DELAY COUNT
2521 015260 100407          :        TSTB     2(R1)   ;CLEAR DELAY COUNTER
2522 015262 005237 011122          :        BMI     .+20     ;IS RDY 0 SET?
2523 015266 001372          :        INC     TEMP    ;BR IF SET
2524 015270 005303          :        BNE     1$      ;INC DELAY COUNTER
2525 015272 001370          :        DEC     R3       ;BR IF NOT DONE DELAY
2526 015274 104014          :        BNE     1$      ;DEC DELAY COUNT
2527 015276 000417          :        ERROR   14     ;BR IF DELAY NOT DONE
2528 015300 132761 000001 000002    :        BR      10$    ;ERROR, RDY 0 NOT SET
2529 015306 001002          :        BITB   #BIT0,2(R1) ;GET OUT
2530 015310 104014          :        BNE     11$    ;IS IT CNTL 0?
2531 015312 000411          :        ERROR   14     ;BR IF YES
2532 015314 012705 000020          :        BR      10$    ;ERROR NOT CNTL 0
2533 015320 016104 000006          11$:    MOV      #BIT4,R5  ;CONTINUE
2534 015324 020504          :        MOV     6(R1),R4 ;PUT 'EXPECTED' IN R5
2535 015326 001401          :        CMP     R5,R4   ;PUT 'FOUND' IN R4
2536 015330 104015          :        BEQ     12$    ;IS LOST DATA SET?
2537 015332 004737 022442          :        ERROR   15     ;BR IF YES
2538          :        JSR     PC,SHUTDOWN ;ERROR, LOST DATA NOT SET
2539          :        ;SHUTDOWN KMC
```

FREE RUNNING TESTS

2538 015336
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550 015336 000004
2551 015340 012737 000004 001202
2552 015346 012737 015464 001442
2553
2554 015354 104410
2555
2556 015356 004737 022030
2557 015362 004537 022410
2558 015366 177320
2559 015370 140044
2560 015372 012703 000015
2561 015376 005037 011122
2562 015402 105761 000002
2563 015406 100407
2564 015410 005237 011122
2565 015414 001372
2566 015416 005303
2567 015420 001370
2568 015422 104014
2569 015424 000417
2570 015426 132761 000001 000002
2571 015434 001002
2572 015436 104014
2573 015440 000411
2574 015442 012705 000400
2575 015446 016104 000006
2576 015452 020504
2577 015454 001401
2578 015456 104015
2579 015460 004737 022442
2580 015464
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592 015464 000004
2593 015466 012737 000005 001202

```
10$: ;SCOPE ;SCOPE THIS TEST

:***** TEST 4 *****
:*TRANSMIT NON-EXISTENT MEMORY TEST
:*IN FREE RUNNING MODE, LOAD A TRANSMIT BA THAT WILL TIME OUT
:*VERIFY THAT A NON-EXISTENT MEMORY ERROR OCCURS
:*****

: TEST 4
:-----
:*****
TST4: SCOPE ; LOAD THE NO. OF THIS TEST
MOV #4,$STNM ; POINT TO THE START OF NEXT TEST.
MOV #TST5,NEXT ;R1 CONTAINS BASE KMC11 ADDRESS
;MASTER CLEAR KMC11

MSTCLR
$SKIPT 10$ ;LOAD KMC BASEMC ADDRESS
JSR PC,BASELD ;LOAD XMIT BA/CC
JSR R5,XFREL
177320 ;BA
140044 ;CC
MOV #15,R3 ;DELAY COUNT
CLR TEMP ;CLEAR DELAY COUNTER
1$: TSTB 2(R1) ;IS RDY 0 SET?
BMI .+20 ;BR IF SET
INC TEMP ;INC DELAY COUNTER
BNE 1$ ;BR IF NOT DONE DELAY
DEC R3 ;DEC DELAY COUNT
BNE 1$ ;BR IF DELAY NOT DONE
ERROR 14 ;ERROR, RDY 0 NOT SET
BR 10$ ;GET OUT
BITB #BIT0,2(R1) ;IS IT CNTL 0?
BNE 11$ ;BR IF YES
ERROR 14 ;ERROR, NOT CNTL 0
BR 10$ ;CONTINUE
11$: MOV #BIT8,R5 ;PUT 'EXPECTED' IN R5
MOV 6(R1),R4 ;PUT 'FOUND' IN R4
CMP R5,R4 ;IS NON-EX-MEM SET?
BEQ .+4 ;BR IF YES
ERROR 15 ;ERROR NON-EX-MEM NOT SET
JSR PC,SHUTDOWN ;SHUTDOWN KMC
10$: ;SCOPE ;SCOPE THIS TEST
```

```
:***** TEST 5 *****
:*RECEIVE NON-EXISTENT MEMORY TEST
:*IN FREE RUNNING MODE, LOAD A RECEIVE BA THAT WILL TIME OUT
:*VERIFY THAT A NON-EXISTENT MEMORY ERROR OCCURS
:*****

: TEST 5
:-----
```

```
:*****
TST5: SCOPE ; LOAD THE NO. OF THIS TEST
MOV #5,$STNM
```


FREE RUNNING TESTS

```
2650 015672 100405          BMI      14$          :BR IF YES
2651 015674 005237 011122  INC      TEMP        :INC DELAY
2652 015700 001372          BNE      13$          :BR IF NOT DONE DELAY
2653 015702 104014          ERROR    14          :ERROR, RDO NOT SET
2654 015704 000770          BR       13$          :TRY AGAIN
2655 015706 132761 000001 000002 14$:  BITB    #BIT0,2(R1) :IS IS CNTL 0?
2656 015714 001002          BNE      11$          :BR IF YES
2657 015716 104014          ERROR    14          :ERROR NOT CNTL 0
2658 015720 000407          BR       10$          :CONTINUE
2659 015722 012705 001000 11$:  MOV     #BIT9,R5      :PUT 'EXPECTED' IN R5
2660 015726 016104 000006  MOV     6(R1),R4      :PUT 'FOUND' IN R4
2661 015732 020504          CMP     R5,R4        :IS PROC ERROR SET?
2662 015734 001401          BEQ     +4           :BR IF YES
2663 015736 104015          ERROR    15          :ERROR, PROC ERROR NOT SET
2664 015740          10$:  :SCOPE              :SCOPE THIS TEST
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676 015740 000004          :***** TEST 7 *****
2677 015742 012737 000007 001202  :*PROCESSOR ERROR TEST
2678 015750 012737 016056 001442  :*IN FREE RUNNING MODE DO A RQI WITH AN ILLEGAL 10 CODE
2679
2680 015756 104410          :*VERIFY THAT A PROCESSOR ERROR OCCURS
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
```

: TEST 7

:*****
TST7: SCOPE
MOV #7,\$STNM ; LOAD THE NO. OF THIS TEST
MOV #TST10,NEXT ; POINT TO THE START OF NEXT TEST.
MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
\$SKIPT 10\$;MASTER CLEAR KMC11
JSR PC,BASELD ;LOAD KMC BASEMC ADDRESS
BISB #46,(R1) ;RQI AND ILLEGAL CODE
TSTB (R1) ;WAIT FOR RDI
BPL -2 ;BR IF NO RDI
BICB #40,(R1) ;CLEAR RQI
CLR TEMP ;CLEAR COUNTER
1\$: TSTB 2(R1) ;RDY 0 SET?
BMI +14 ;BR IF YES
INC TEMP ;BUMP COUNTER DELAY
BNE 1\$;BR IF NOT DONE
ERROR 14 ;ERROR NO RDY 0
BR 1\$;TRY AGAIN
BITB #BIT0,2(R1) ;IS IT CNTL 0
BNE 11\$;BR IF YES
ERROR 14 ;ERROR, NOT CNTL 0
BR 10\$;CONTINUE
11\$: MOV #BIT9,R5 ;PUT 'EXPECTED' IN R5
MOV 6(R1),R4 ;PUT 'FOUND' IN R4
CMP R5,R4 ;IS PROC ERROR SET?
BEQ +4 ;BR IF YES
ERROR 15 ;ERROR PROC ERROR NOT SET
10\$: :SCOPE ;SCOPE THIS TEST

FREE RUNNING TESTS

2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761

016056 000004
016060 012737 000010 001202
016066 012737 016156 001442
016074 104410
016076 004737 022136
016102 004537 022356
016106 021362
016110 000044
016112 004537 022410
016116 021314
016120 000044
016122 012703 000003
016126 005037 011122
016132 105761 000002
016136 100406
016140 005237 011122
016144 001372
016146 005303
016150 001370
016152 000401
016154 104014
016156

```
***** TEST 10 *****  
*HALF DUPLEX TEST  
*IN FREE RUNNING MODE, SET HALF DUPLEX AND L U LOOP  
*SEND A MESSAGE AND VERIFY THAT THERE ARE NO DONES  
*****  
: TEST 10  
:-----  
:*****  
TST10: SCOPE  
MOV #10,$STNM ; LOAD THE NO. OF THIS TEST  
MOV #TST11,NEXT ; POINT TO THE START OF NEXT TEST.  
;R1 CONTAINS BASE KMC11 ADDRESS  
;MASTER CLEAR KMC11  
MSTCLR  
$SKIPT 10$  
JSR PC,BASELH ;LOAD BASEMC AND HALF DUPLEX  
JSR R5,RFRELD ;LOAD RECEIVE BUFFER  
RBUF ;BA  
44 ;CC  
JSR R5,XFRELD ;LOAD TRANSMIT BUFFER  
TBUF ;BA  
44 ;CC  
MOV #3,R3 ;LOAD DELAY COUNT  
CLR TEMP ;CLEAR DELAY  
4$: TSTB 2(R1) ;IS DONE SET?  
BMI 5$ ;BR IF YES (ERROR)  
INC TEMP ;INC DELAY  
BNE 4$ ;BR IF DELAY NOT DONE  
DEC R3 ;DEC DELAY COUNT  
BNE 4$ ;BR IF DELAY NOT DONE  
BR 10$  
5$: ERROR 14 ;ERROR DONE WITH HALF-DUPLEX  
10$:
```

```
***** TEST 11 *****  
*RESUME TEST  
*THIS TEST SENDS AND RECEIVES A BUFFER AND SHUTS DOWN THE  
*KMC. THEN A MASTER CLEAR IS ISSUED AND A BASEMC WITH RESUME  
*BIT SET IS GIVEN, ANOTHER BUFFER IS SENT AND RECEIVED.  
*DATA IS CHECKED.  
*****
```

: TEST 11
:-----

```
:*****  
TST11: SCOPE  
MOV #11,$STNM ; LOAD THE NO. OF THIS TEST  
MOV #TST12,NEXT ; POINT TO THE START OF NEXT TEST.  
;R1 CONTAINS BASE KMC11 ADDRESS  
;MASTER CLEAR KMC11  
MSTCLR  
$SKIPT 10$  
CLR RESUME ;CLR RESUME FLAG  
1$: TST RESUME ;FIRST OR SECOND PASS?  
BNE 2$ ;BR IF SECOND  
JSR PC,BASELD ;BASEMC
```

FREE RUNNING TESTS

2762	016214	000402			BR	3\$:CONTINUE
2763	016216	004737	022246		JSR	PC,RESUM	:BASEMC WITH RESUME BIT
2764	016222	004537	022356		JSR	R5,RFRELD	:RECEIVE BUFFER
2765	016226	021362			RBUF		:BA
2766	016230	000044			44		:CC
2767	016232	004537	022410		JSR	R5,XFRELD	:XMIT BUFFER
2768	016236	021314			TBUF		:BA
2769	016240	000044			44		:CC
2770	016242	012703	000030		MOV	#30,R3	:DELAY COUNT
2771	016246	012700	000002		MOV	#2,R0	:NEED TWO DONES
2772	016252	005037	011122		CLR	TEMP	:CLEAR DELAY COUNTER
2773	016256	105761	000002		4\$: TSTB	2(R1)	:IS RDY 0 SET?
2774	016262	100407			BMI	.+20	:BR IF SET
2775	016264	005237	011122		INC	TEMP	:INC DELAY COUNTER
2776	016270	001372			BNE	4\$:BR IF NOT DONE DELAY
2777	016272	005303			DEC	R3	:DEC DELAY COUNT
2778	016274	001370			BNE	4\$:BR IF DELAY NOT DONE
2779	016276	104014			ERROR	14	:ERROR, RDY 0 NOT SET
2780	016300	000433			BR	10\$:GET OUT
2781	016302	042761	000207	000002	BIC	#207,2(R1)	:CLEAR DONE
2782	016310	005300			DEC	R0	:TWO DONES YET?
2783	016312	001361			BNE	4\$:BR IF NOT
2784	016314	012702	021314		MOV	#TBUF,R2	:ADDRESS OF GOOD DATA
2785	016320	012703	021362		MOV	#RBUF,R3	:ADDRESS OF RECEIVED DATA
2786	016324	012700	000044		MOV	#44,R0	:COUNT
2787	016330	112205			6\$: MOVB	(R2)+,R5	:LOAD GOOD DATA
2788	016332	112304			MOVB	(R3)+,R4	:LOAD FOUND DATA
2789	016334	120504			CMPB	R5,R4	:COMPARE DATA
2790	016336	001401			BEQ	7\$:BR IF OK
2791	016340	104012			ERROR	12	:DATA ERROR
2792	016342	005300			7\$: DEC	R0	:DONE YET?
2793	016344	001371			BNE	6\$:BR IF NOT
2794	016346	004737	022442		JSR	PC,SHUTDOWN	:SHUTDOWN KMC
2795	016352	005737	020050		TST	RESUME	:
2796	016356	001004			BNE	8\$:BR IF ALL DONE
2797	016360	012737	177777	020050	MOV	#-1,RESUME	:SET FLAG FOR SECOND PASS
2798	016366	000705			BR	1\$:CONTINUE
2799	016370				8\$:		
2800	016370				10\$:	:SCOPE	:SCOPE THIS TEST

```

2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815

```

***** TEST 12 *****
: *FREE RUNNING DATA TEST (INTERRUPT DRIVEN EXERCISER)
: *THIS TEST REPEATEDLY QUEUES UP 7 RECEIVE BUFFERS AND
: *7 TRANSMIT BUFFERS AND CHECKS DATA WHEN ALL 7 BUFFERS
: *ARE RECEIVED. TRANSMIT COUNTS RANGE FROM 2 TO 104.
: *DATA IS A BINARY COUNT PATTERN. THE RESUME FUNCTION
: *IS CHECKED IN THIS TEST. THIS TEST USES THE TURNAROUND CONNECTOR
: *IF IT IS PRESENT, OTHERWISE LINE UNIT LOOP IS SET.
: *****

```

: TEST 12
:-----
: *****
: *TST12: SCOPE
: MOV #12,$STNM ; LOAD THE NO. OF THIS TEST

```

```

2816 016370 000004
2817 016372 012737 000012 001202

```


FREE RUNNING TESTS

2818	016400	012737	003662	001442	MOV	#\$EOP,NEXT		: POINT TO THE START OF NEXT TEST.
2819								:R1 CONTAINS BASE KMC11 ADDRESS
2820	016406	104410			MSTCLR			:MASTER CLEAR KMC11
2821					\$SKIPT	ENDEX1		
2822	016410	012737	000340	177776	MOV	#340,PS		:LOCK OUT INTERRUPTS
2823	016416	013700	002050		MOV	STAT1,R0		:GET BR LEVEL
2824	016422	006200			ASR	R0		:SHIFT RIGHT 4 TIMES
2825	016424	006200			ASR	R0		
2826	016426	006200			ASR	R0		
2827	016430	006200			ASR	R0		
2828	016432	042700	177437		BIC	#177437,R0		:PUT BR LEVEL IN R0
2829	016436	012777	017132	163412	MOV	#IISR,@KMRVEC		:LOAD INPUT VECTOR
2830	016444	010077	163410		MOV	R0,@KMRLVL		:LOAD LEVEL
2831	016450	012777	017422	163404	MOV	#OISR,@KMTVEC		:LOAD OUTPUT VECTOR
2832	016456	010077	163402		MOV	R0,@KMTLVL		:LOAD LEVEL
2833								
2834								:INITIALIZE ALL BUFFER LISTS AND COUNT LISTS
2835								
2836	016462	012737	000104	021306	MOV	#104,TFLAG		:TFLAG CONTAINS COUNT
2837	016470	012700	020054		MOV	#XMITBA+2,R0		:R0 POINTS TO BA LIST
2838	016474	012703	020346		MOV	#RBUF,R3		:R3 CONTAINS BUFFER ADDRESS
2839	016500	010320			1\$: MOV	R3,(R0)+		:LOAD BA LIST WITH REC BA
2840	016502	062703	000104		ADD	#104,R3		:UPDATE BUFFER ADDRESS
2841	016506	022700	020072		CMP	#XMITBA+20,R0		:END OF REC BUFFERS?
2842	016512	001372			BNE	1\$:NO LOAD NEXT ONE
2843	016514	012720	020110		2\$: MOV	#TBUF,(R0)+		:LOAD BA LIST WITH XMIT BA
2844	016520	022700	020110		CMP	#XMITBA+36,R0		:END OF XMIT BUFFERS?
2845	016524	001373			BNE	2\$:NO LOAD NEXT BUFFER
2846	016526	012700	020222		MOV	#RCNTAB+2,R0		:R0 POINTS TO COUNT LIST
2847	016532	013720	021306		3\$: MOV	TFLAG,(R0)+		:LOAD COUNT OF 104
2848	016536	022700	020240		CMP	#RCNTAB+20,R0		:END OF REC COUNT LIST?
2849	016542	001373			BNE	3\$:BR IF NO
2850	016544	012737	000005	021304	MOV	#5,FLAG ;LOOP COUNT		
2851	016552	012711	040000		MOV	#BIT14,(R1)		:SET MASTER CLEAR
2852					:	BIT	#BIT15,STAT1	:IOP?
2853					:	BEQ	+.6	:BR IF NO
2854	016556	012711	100000		MOV	#BIT15,(R1)		:SET RUN ON IOP
2855	016562	012700	177777		MOV	#-1,R0		:R0 IS INPUT DONE COUNTER
2856	016566	005037	020050		CLRTAB: CLR	RESUME		:CLEAR RESUME FLAG
2857	016572	012705	020256		MOV	#RDNTAB,R5		:GET READY TO CLEAR ALL RECEIVE
2858	016576	005025			2\$: CLR	(R5)+		:BUFFERS
2859	016600	022705	021302		CMP	#RBUFE,R5		:END OF BUFFER?
2860	016604	001374			BNE	2\$:BR IF NO
2861	016606	012704	020240		MOV	#XCNTAB,R4		:R4 POINTS TO XMIT COUNT LIST
2862	016612	013724	021306		4\$: MOV	TFLAG,(R4)+		:LOAD XMIT CHAR COUNT
2863	016616	022704	020256		CMP	#XCNTAB+16,R4		:DONE?
2864	016622	001373			BNE	4\$:BR IF NO
2865	016624	005002			5\$: CLR	R2		:R2 IS OUTPUT DONE COUNTER
2866	016626	005004			CLR	R4		:R4 IS USED AS INDEX IN OISR
2867	016630	005711			TST	(R1)		:IS RUN SET?
2868	016632	100376			BPL	-.2		:WAIT FOR RUN
2869	016634	152761	000100	000002	BISB	#BIT6,2(R1)		:SET IEO
2870	016642	032737	040000	002050	BIT	#BIT14,STAT1		:LOOP BACK CONNECTOR?
2871	016650	001002			BNE	+.6		:BR IF YES
2872	016652	052711	004000		BIS	#BIT11,(R1)		:SET LINE UNIT LOOP
2873	016656	022737	000005	021304	CMP	#5,FLAG		:FIRST TIME?

FREE RUNNING TESTS

2874	016664	001003			BNE	1\$:BR IF NOT
2875	016666	052711	000143		BIS	#143,(R1)	:SET IEI,RQI,BASEMC I
2876	016672	000402			BR	3\$:CONTINUE
2877	016674	052711	000144		1\$: BIS	#144,(R1)	:SET IEI,RQI,REC BA/CC
2878	016700	005037	011122		3\$: CLR	TEMP	:SET UP FOR DELAY COUNT
2879	016704	012737	000022	001300	MOV	#22,\$TMP1	:GET SET FOR DELAY
2880	016712	005037	177776		CLR	PS	:ALLOW INTERRUPTS
2881	016716	022700	000020		SCAN: CMP	#20,R0	:INPUT DONE?
2882	016722	001402			BEQ	SCAN2	:BR IF YES
2883	016724	000137	017102		JMP	SCAN1	:BR IF NO
2884	016730	022702	000034		SCAN2: CMP	#34,R2	:XMIT DONE FOR ALL MESSAGES?
2885	016734	001402			BEQ	8\$:BR IF YES
2886	016736	000137	017102		JMP	SCAN1	:BR IF NO
2887	016742	022704	000034		8\$: CMP	#34,R4	:REC DONE FOR ALL MESSAGES?
2888	016746	001402			BEQ	9\$:BR IF YES
2889	016750	000137	017102		JMP	SCAN1	:BR IF NO
2890	016754				9\$:		
2891	016754	012700	020256		MOV	#RDNTAB,R0	:GET FIRST REC BUFFER
2892	016760	012002			5\$: MOV	(R0)+,R2	:R2 POINTS TO BUFFER
2893	016762	005005			CLR	R5	:R5=EXPECTED
2894	016764	005003			CLR	R3	:R3 = COUNT
2895	016766	010237	001302		6\$: MOV	R2,\$TMP2	:SAVE ADDRESS FOR TYPEOUT
2896	016772	112204			MOVB	(R2)+,R4	:GET RECEIVE DATA
2897	016774	120504			CMPB	R5,R4	:IS IT CORRECT?
2898	016776	001401			BEQ	.+4	:BR IF YES
2899	017000	104013			ERROR	13	:DATA ERROR
2900	017002	005205			INC	R5	:NEXT CHARACTER
2901	017004	005203			INC	R3	:INC COUNT
2902	017006	021003			CMP	(R0),R3	:DONE YET?
2903	017010	001366			BNE	6\$:BR IF NO
2904	017012	062700	000002		ADD	#2,R0	:GET NEXT REC BUFFER
2905	017016	022700	020312		CMP	#RDNTAB+34,R0	:DONE YET?
2906	017022	001356			BNE	5\$:BR IF NO
2907	017024	012700	000001		MOV	#1,R0	:SET R0 TO 1
2908	017030	032737	000001	021304	4\$: BIT	#BIT0,FLAG	:CHANGE CHAR COUNT FOR NEXT LOOP
2909	017036	001003			BNE	1\$:BR TO SUB 40
2910	017040	005337	021306		DEC	TFLAG	:DEC BY ONE
2911	017044	000403			BR	2\$:CONTINUE
2912	017046	162737	000040	021306	1\$: SUB	#40,TFLAG	:SUBTRACT 40 FROM XMIT COUNT
2913	017054	005337	021304		2\$: DEC	FLAG	:DEC LOOP COUNT
2914	017060	001242			BNE	CLRTAB	:GO DO IT AGAIN
2915	017062	152711	000146		ENDEX: BISB	#146,(R1)	:SHUT DOWN KMC
2916	017066	005737	021304		1\$: TST	FLAG	:HAS INTERRUPT OCCURED?
2917	017072	001775			BEQ	1\$:BR IF NO
2918	017074	000400			BR	ENDEX1	:ALL OK GET OUT
2919	017076	000004			ENDEX1: SCOPE		:SCOPE THIS TEST
2920	017100	104420			ENDEX2: ADVANCE		
2921	017102	005337	011122		SCAN1: DEC	TEMP	:DECREMENT DELAY COUNTER
2922	017106	001402			BEQ	1\$:BR IF ZERO
2923	017110	000137	016716		JMP	SCAN	:BR IF NOT DONE DELAY
2924	017114	005337	001300		1\$: DEC	\$TMP1	:DEC DELAY COUNT
2925	017120	001402			BEQ	2\$:BR IF DONE DELAY
2926	017122	000137	016716		JMP	SCAN	:BR IF NOT DONE
2927	017126	104001			2\$: ERROR	1	:ERROR HUNG
2928	017130	000762			BR	ENDEX1	:GET OUT
2929							

FREE RUNNING TESTS

;INPUT INTERRUPT SERVICE ROUTINE

2930									
2931									
2932	017132	022700	000017		IISR:	CMP	#17,R0		;PROC. ERROR DONE?
2933	017136	001421				BEQ	12\$;BR IF YES
2934	017140	005737	020050			TST	RESUME		;IS THIS A RESUME INTERRUPT
2935	017144	001432				BEQ	8\$;BR IF NO
2936	017146	032711	000002			BIT	#BIT1,(R1)		;CNTL OR BASEMC?
2937	017152	001407				BEQ	13\$;BR IF CNTL I
2938	017154	012761	021430	000004		MOV	#BASEMC,4(R1)		;LOAD BASEMC ADDRESS
2939	017162	012761	010000	000006		MOV	#BIT12,6(R1)		;WITH RESUME BIT SET
2940	017170	000404				BR	12\$;CONTINUE
2941	017172	005061	000006		13\$:	CLR	6(R1)		;SELECT FULL DUPLEX
2942	017176	005037	020050			CLR	RESUME		;CLEAR RESUME FLAG
2943	017202	142711	000040		12\$:	BICB	#40,(R1)		;CLEAR RQI
2944	017206	105711				TSTB	(R1)		;IS RQI GONE?
2945	017210	100776				BMI	.-2		;BR IF NO
2946	017212	005737	020050			TST	RESUME		;BASEMC OR CNTL I?
2947	017216	001403				BEQ	14\$;BR IF IT WAS CNTL I
2948	017220	152711	000041			BISB	#41,(R1)		;ASK FOR CNTL I
2949	017224	000002				RTI			;RETURN
2950	017226	105011			14\$:	CLRB	(R1)		;CLEAR BSEL 0
2951	017230	000002				RTI			;RETURN
2952	017232	005700			8\$:	TST	R0		;FIRST TIME HERE?
2953	017234	100006				BPL	7\$;LOAD BASEMC IF MINUS
2954	017236	012761	021430	000004		MOV	#BASEMC,4(R1)		;SET UP BASEMC ADDRESS
2955	017244	005061	000006			CLR	6(R1)		;CLEAR COUNT
2956	017250	000434				BR	3\$;CONTINUE
2957	017252	001003			7\$:	BNE	1\$;CNTL I FULL DUPLEX IF 0
2958	017254	005061	000006			CLR	6(R1)		;SELECT FULL DUPLEX
2959	017260	000430				BR	3\$;CONTINUE
2960	017262	032700	000010		1\$:	BIT	#BIT3,R0		;XMIT?
2961	017266	001013				BNE	2\$;BR IF YES
2962	017270	000241				CLC			;CLEAR CARRY
2963	017272	006100				ROL	R0		;MAKE R0 EVEN
2964	017274	016061	020052	000004		MOV	RECBA(R0),4(R1)		;LOAD REC BUFFER
2965	017302	016061	020220	000006		MOV	RCNTAB(R0),6(R1)		;LOAD COUNT
2966	017310	000241				CLC			;CLEAR CARRY
2967	017312	006000				ROR	R0		;GET R0 BACK
2968	017314	000412				BR	3\$;CONTINUE
2969	017316	000241			2\$:	CLC			;CLEAR CARRY
2970	017320	006100				ROL	R0		;MAKE IT EVEN
2971	017322	016061	020052	000004		MOV	XMITBA(R0),4(R1)		;LOAD XMIT BUFFER
2972	017330	016061	020220	000006		MOV	RCNTAB(R0),6(R1)		;LOAD COUNT
2973	017336	000241				CLC			;CLEAR CARRY
2974	017340	006000				ROR	R0		;PUT IT BACK
2975	017342	142711	000040		3\$:	BICB	#40,(R1)		;CLEAR RQI
2976	017346	105711				TSTB	(R1)		;WAIT FOR
2977	017350	100776				BMI	.-2		;RDI TO GO AWAY
2978	017352	005200				INC	R0		;INC COUNT
2979	017354	001003				BNE	6\$;IF 0 ASK FOR CNTL I
2980	017356	152711	000041			BISB	#41,(R1)		;ASK FOR CNTL I
2981	017362	000002				RTI			;RETURN
2982	017364	022700	000017		6\$:	CMP	#17,R0		;DONE YET?
2983	017370	001411				BEQ	4\$;BR IF YES
2984	017372	032700	000010			BIT	#BIT3,R0		;XMIT?
2985	017376	001003				BNE	5\$;BR IF YES

FREE RUNNING TESTS

2986	017400	152711	000044		BISB	#44,(R1)	:ASK FOR REC BA/CC
2987	017404	000002			RTI		:RETURN
2988	017406	152711	000040	5\$:	BISB	#40,(R1)	:ASK FOR XMIT BA/CC
2989	017412	000002			RTI		:RETURN
2990	017414	152711	000046	4\$:	BISB	#46,(R1)	:FORCE PROC. ERROR
2991	017420	000002			RTI		:RETURN
2992							
2993							
2994							:OUTPUT INTERRUPT SERVICE ROUTINE
2995	017422	032761	000001 000002	OISR:	BIT	#BIT0,2(R1)	:IS THIS AN ERROR?
2996	017430	001463			BEQ	1\$:BR IF NO
2997	017432	005737	021304		TST	FLAG	:IS THIS SHUT DOWN INTERRUPT?
2998	017436	001006			BNE	9\$:BR IF NO
2999	017440	005237	021304		INC	FLAG	:YES MAKE FLAG NON-ZERO
3000	017444	022761	001000 000006		CMP	#BIT9,6(R1)	:SHUT DOWN BIT SET?
3001	017452	001525			BEQ	10\$:YES ALL IS OK
3002	017454	022700	000017	9\$:	CMP	#17,R0	:RESUME INTERRUPT?
3003	017460	001035			BNE	11\$:BR IF NO
3004	017462	022761	001000 000006		CMP	#BIT9,6(R1)	:PROC. ERROR BIT SET?
3005	017470	001031			BNE	11\$:BR IF NO
3006	017472	005200			INC	R0	:BUMP COUNTER (TO 20)
3007	017474	012711	040000		MOV	#BIT14,(R1)	:MASTER CLEAR DEVICE
3008				:	BIT	#BIT15,STAT1	:KMC OR KMC?
3009				:	BEQ	+.14	:BR IF KMC
3010	017500	012711	100000		MOV	#BIT15,(R1)	:SET RUN ON KMC
3011	017504	105227	000000		INCB	#0	:DELAY ON KMC
3012	017510	001375			BNE	.-4	
3013	017512	012737	177777 020050		MOV	#-1,RESUME	:SET RESUME FLAG
3014	017520	005711			TST	(R1)	:RUN SET?
3015	017522	100376			BPL	.-2	:BR IF NO
3016	017524	012761	000100 000002		MOV	#BIT6,2(R1)	:SET IEO
3017	017532	032737	040000 002050		BIT	#BIT14,STAT1	:LOOP BACK CONNECTOR?
3018	017540	001002			BNE	+.6	:BR IF YES
3019	017542	052711	004000		BIS	#BIT11,(R1)	:SET LINE UNIT LOOP
3020	017546	052711	000143		BIS	#143,(R1)	:ASK FOR PORT (BASEMC REQUEST)
3021	017552	000002			RTI		:RETURN
3022	017554	016137	000004 001302	11\$:	MOV	4(R1), \$TMP2	:SAVE FOR ERROR TYPEOUT
3023	017562	016137	000006 001304		MOV	6(R1), \$TMP3	:SAVE FOR ERROR TYPEOUT
3024	017570	104016			ERROR	16	:CNTL 0 ERROR
3025	017572	022626			CMP	(SP)+,(SP)+	:ADJUST STACK
3026	017574	000137	017076		JMP	ENDEX1	:GET OUT
3027	017600	032761	000004 000002	1\$:	BIT	#BIT2,2(R1)	:RECEIVE?
3028	017606	001053			BNE	2\$:BR IF YES
3029	017610	022761	020110 000004		CMP	#TBUFF,4(R1)	:IS XMIT BA CORRECT?
3030	017616	001412			BEQ	4\$:BR IF OK
3031	017620	022761	020111 000004		CMP	#TBUFF+1,4(R1)	:IS XMIT BA CORRECT?
3032	017626	001406			BEQ	4\$:BR IF YES
3033	017630	012705	020110		MOV	#TBUFF,R5	:R5 = EXPECTED
3034	017634	016137	000004 001302		MOV	4(R1), \$TMP2	:SAVE FOUND FOR TYPEOUT
3035	017642	104002			ERROR	2	:XMIT BA ERROR
3036	017644	005005		4\$:	CLR	R5	:R5 IS INDEX REG
3037	017646	026561	020240 000006	5\$:	CMP	XCNTAB(R5),6(R1)	:IS CHAR COUNT OK?
3038	017654	001406			BEQ	6\$:BR IF YES
3039	017656	062705	000002		ADD	#2,R5	:INC INDEX
3040	017662	022705	000016		CMP	#16,R5	:DONE LIST YET?
3041	017666	001367			BNE	5\$:BR IF NO

FREE RUNNING TESTS

3042	017670	104003				ERROR	3		:XMIT COUNT ERROR
3043	017672	016162	000004	020312	6\$:	MOV	4(R1),XDNTAB(R2)	:	STORE XMIT DONE BA
3044	017700	062702	000002			ADD	#2,R2	:	INC INDEX
3045	017704	016162	000006	020312		MOV	6(R1),XDNTAB(R2)	:	STORE XMIT DONE CC
3046	017712	062702	000002			ADD	#2,R2	:	INC INDEX
3047	017716	142761	000207	000002		BICB	#207,2(R1)	:	CLEAR RDO
3048	017724	000002				RTI		:	RETURN
3049	017726	105011			10\$:	CLRB	(R1)	:	CLEAR SEL0
3050	017730	105061	000002			CLRB	2(R1)	:	CLEAR SEL2
3051	017734	000002				RTI		:	RETURN
3052	017736	012705	000002		2\$:	MOV	#2,R5	:	SET UP R5 AS INDEX
3053	017742	026561	020052	000004		CMP	RECBA(R5),4(R1)	:	COMPARE WITH LIST OF CORRECT BA'S
3054	017750	001406				BEQ	3\$:	BR IF OK?
3055	017752	062705	000002			ADD	#2,R5	:	INCREMENT R5
3056	017756	022705	000020			CMP	#20,R5	:	END CF LIST?
3057	017762	001367				BNE	2\$+4	:	BR IF NO
3058	017764	104004				ERROR	4	:	REC BA ERROR
3059	017766	005005			3\$:	CLR	R5	:	R5 IS INDEX
3060	017770	026561	020240	000006	7\$:	CMP	XCNTAB(R5),6(R1)	:	CHECK FOR CORRECT REC COUNT
3061	017776	001406				BEQ	8\$:	BR IF YES
3062	020000	062705	000002			ADD	#2,R5	:	INCREMENT R5
3063	020004	022705	000016			CMP	#16,R5	:	END OF LIST?
3064	020010	001367				BNE	7\$:	BR IF NOT
3065	020012	104005				ERROR	5	:	REC COUNT ERROR
3066	020014	016164	000004	020256	8\$:	MOV	4(R1),RDNTAB(R4)	:	STORE REC BA
3067	020022	062704	000002			ADD	#2,R4	:	INC INDEX
3068	020026	016164	000006	020256		MOV	6(R1),RDNTAB(R4)	:	STORE REC DONE CC
3069	020034	062704	000002			ADD	#2,R4	:	INC INDEX
3070	020040	142761	000207	000002		BICB	#207,2(R1)	:	CLEAR RDO
3071	020046	000002				RTI		:	RETURN
3072									
3073									
3074									
3075									
3076	020050	000000							
3077	020052					RESUME:	0		
3078	020052	000017				RECBA:			
3079						XMITBA:	.BLKW 17	:	REC & XMIT BA LIST
3080	020110					TBUFF:		:	TRANSMIT DATA
3081	020110	000	001	002		.BYTE	0,1,2,3,4,5,6,7		
3082	020113	003	004	005					
3083	020116	006	007						
3084	020120	010	011	012		.BYTE	10,11,12,13,14,15,16,17		
3085	020123	013	014	015					
3086	020126	016	017						
3087	020130	020	021	022		.BYTE	20,21,22,23,24,25,26,27		
3088	020133	023	024	025					
3089	020136	026	027						
3090	020140	030	031	032		.BYTE	30,31,32,33,34,35,36,37		
3091	020143	033	034	035					
3092	020146	036	037						
3093	020150	040	041	042		.BYTE	40,41,42,43,44,45,46,47		
3094	020153	043	044	045					
3095	020156	046	047						
3096	020160	050	051	052		.BYTE	50,51,52,53,54,55,56,57		
3097	020163	053	054	055					

FREE RUNNING TESTS

```
3098 020166 056 057
3099 020170 060 061 062 .BYTE 60,61,62,63,64,65,66,67
3100 020173 063 064 065
3101 020176 066 067
3102 020200 070 071 072 .BYTE 70,71,72,73,74,75,76,77
3103 020203 073 074 075
3104 020206 076 077
3105 020210 100 101 102 .BYTE 100,101,102,103,104,105,106,107
3106 020213 103 104 105
3107 020216 106 107
3108
3109 020220 000010 RCNTAB: .BLKW 10 ;RECEIVE COUNT TABLE
3110 020240 000007 XCNTAB: .BLKW 7 ;TRANSMIT COUNT TABLE
3111
3112 020256 000016 RDNTAB: .BLKW 16 ;RECEIVE DONE TABLE (BA/CC)
3113 020312 000016 XDNTAB: .BLKW 16 ;XMIT DONE TABLE (BA/CC)
3114
3115 020346 RBUFF: ;RECEIVER BUFFERS
3116 020346 000104 RBUFF1: .BLKB 104
3117 020452 000104 RBUFF2: .BLKB 104
3118 020556 000104 RBUFF3: .BLKB 104
3119 020662 000104 RBUFF4: .BLKB 104
3120 020766 000104 RBUFF5: .BLKB 104
3121 021072 000104 RBUFF6: .BLKB 104
3122 021176 000104 RBUFF7: .BLKB 104
3123 021302 000000 RUFFE: 0 ;END OF RECEIVER BUFFERS
3124
3125
3126
3127 ;BUFFER AREA
3128 ;-----
3129
3130 021304 000000 FLAG: 0
3131 021306 000000 TFLAG: 0
3132 021310 000000 RFLAG: 0
3133 021312 000044 TCOUNT: 44
3134 021314 041101 042103 043105 TBUF: .ASCII/ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789/
3135 021322 044107 045111 046113
3136 021330 047115 050117 051121
3137 021336 052123 053125 054127
3138 021344 055131 030460 031462
3139 021352 032464 033466 034470
3140
3141 021360 000044 .EVEN
3142 021362 021430 RCOUNT: 44
3143 RBUF: .+.46
3144 021430 022030 .EVEN
3145 BASEMC: .+.256.
3146
3147 ;SUBROUTINES
3148 ;-----
3149
3150
3151 022030 BASELD:
3152 ;THIS SUBROUTINE LOADS THE KMC WITH A BASEMC ADDRESS
3153 ;AND PUTS KMC INTO FULL-DUPLEX MODE
```


SUBROUTINES

3154									
3155	022030	012711	040000			MOV	#BIT14,(R1)	:MASTER CLEAR	
3156				:		BIT	#BIT15,STAT1	:CRAM?	
3157				:		BEQ	+.6	:BR IF NO	
3158	022034	012711	100000			MOV	#BIT15,(R1)	:IF CRAM SET RUN	
3159	022040	105227	000000			INCB	#0	:DELAY	
3160	022044	001375				BNE	-.4	:BR IF NOT DONE DELAY	
3161	022046	005711			1\$:	TST	(R1)	:IS RUN SET?	
3162	022050	100376				BPL	1\$:BR IF NO	
3163	022052	052711	004000			BIS	#BIT11,(R1)	:SET LU LOOP	
3164	022056	152711	000043			BISB	#43,(R1)	:BASEMC REQUEST	
3165	022062	105711			2\$:	TSTB	(R1)	:RDY I SET?	
3166	022064	100376				BPL	2\$:BR IF NO	
3167	022066	012761	021430	000004		MOV	#BASEMC,4(R1)	:LOAD BASEMC ADDRESS	
3168	022074	005061	000006			CLR	6(R1)	:CLEAR CC	
3169	022100	142711	000040			BICB	#40,(R1)	:CLEAR RQI	
3170	022104	105711			3\$:	TSTB	(R1)	:RDY I CLEAR?	
3171	022106	100776				BMI	3\$:BR IF NO	
3172	022110	152711	000041			BISB	#41,(R1)	:ASK FOR CNTL I	
3173	022114	105711			64\$:	TSTB	(R1)	:WAIT FOR RDI	
3174	022116	100376				BPL	64\$:BR IF NOT SETY	
3175	022120	005061	000006			CLR	6(R1)	:SET FULL DUPLEX	
3176	022124	142711	000040			BICB	#40,(R1)	:CLEAR RQI	
3177	022130	105711			65\$:	TSTB	(R1)	:RDI UP?	
3178	022132	100776				BMI	65\$:BR IF YES	
3179	022134	000207				RTS	PC	:RETURN	
3180									
3181	022136					BASELH:			
3182								:THIS SUBROUTINE LOADS THE KMC WITH A BASEMC ADDRESS	
3183								:AND PUTS KMC INTO HALF-DUPLEX MODE	
3184									
3185	022136	012711	040000			MOV	#BIT14,(R1)	:MASTER CLEAR	
3186				:		BIT	#BIT15,STAT1	:CRAM?	
3187				:		BEQ	+.6	:BR IF NO	
3188	022142	012711	100000			MOV	#BIT15,(R1)	:IF CRAM SET RUN	
3189	022146	105227	000000			INCB	#0	:DELAY	
3190	022152	001375				BNE	-.4	:BR IF NOT DONE DELAY	
3191	022154	005711			1\$:	TST	(R1)	:IS RUN SET?	
3192	022156	100376				BPL	1\$:BR IF NO	
3193	022160	052711	004000			BIS	#BIT11,(R1)	:SET LU LOOP	
3194	022164	152711	000043			BISB	#43,(R1)	:BASEMC REQUEST	
3195	022170	105711			2\$:	TSTB	(R1)	:RDY I SET?	
3196	022172	100376				BPL	2\$:BR IF NO	
3197	022174	012761	021430	000004		MOV	#BASEMC,4(R1)	:LOAD BASEMC ADDRESS	
3198	022202	005061	000006			CLR	6(R1)	:CLEAR CC	
3199	022206	142711	000040			BICB	#40,(R1)	:CLEAR RQI	
3200	022212	105711			3\$:	TSTB	(R1)	:RDY I CLEAR?	
3201	022214	100776				BMI	3\$:BR IF NO	
3202	022216	152711	000041			BISB	#41,(R1)	:ASK FOR CNTL I	
3203	022222	105711			64\$:	TSTB	(R1)	:WAIT FOR RDI	
3204	022224	100376				BPL	64\$:BR IF NOT SETY	
3205	022226	012761	002000	000006		MOV	#BIT10,6(R1)	:SET HALF DUPLEX	
3206	022234	142711	000040			BICB	#40,(R1)	:CLEAR RQI	
3207	022240	105711			65\$:	TSTB	(R1)	:RDI UP?	
3208	022242	100776				BMI	65\$:BR IF YES	
3209	022244	000207				RTS	PC	:RETURN	

SUBROUTINES

3210					
3211	022246				RESUM:
3212					;THIS SUBROUTINE LOADS THE KMC WITH A BASEMC ADDRESS
3213					;WITH RESUME BIT SET AND PUTS KMC INTO FULL-DUPLEX MODE
3214					
3215	022246	012711	040000		MOV #BIT14,(R1) ;MASTER CLEAR
3216				:	BIT #BIT15,STAT1 ;CRAM?
3217				:	BEQ .+6 ;BR IF NO
3218	022252	012711	100000		MOV #BIT15,(R1) ;IF CRAM SET RUN
3219	022256	105227	000000		INCB #0 ;DELAY
3220	022262	001375			BNE .-4 ;BR IF NOT DONE DELAY
3221	022264	005711		1\$:	TST (R1) ;IS RUN SET?
3222	022266	100376			BPL 1\$;BR IF NO
3223	022270	052711	004000		BIS #BIT11,(R1) ;SET LU LOOP
3224	022274	152711	000043		BISB #43,(R1) ;BASEMC REQUEST
3225	022300	105711		2\$:	TSTB (R1) ;RDY I SET?
3226	022302	100376			BPL 2\$;BR IF NO
3227	022304	012761	021430	000004	MOV #BASEMC,4(R1) ;LOAD BASEMC ADDRESS
3228	022312	012761	010000	000006	MOV #BIT12,6(R1) ;SET RESUME BIT
3229	022320	142711	000040		BICB #40,(R1) ;CLEAR RQI
3230	022324	105711		3\$:	TSTB (R1) ;RDY I CLEAR?
3231	022326	100776			BMI 3\$;BR IF NO
3232	022330	152711	000041		BISB #41,(R1) ;ASK FOR CNTL I
3233	022334	105711		64\$:	TSTB (R1) ;WAIT FOR RDI
3234	022336	100376			BPL 64\$;BR IF NOT SETY
3235	022340	005061	000006		CLR 6(R1) ;SET FULL DUPLEX
3236	022344	142711	000040		BICB #40,(R1) ;CLEAR RQI
3237	022350	105711		65\$:	TSTB (R1) ;RDI UP?
3238	022352	100776			BMI 65\$;BR IF YES
3239	022354	000207			RTS PC ;RETURN
3240					
3241	022356				RFRELD:
3242					;THIS SUBROUTINE LOADS THE KMC WITH A RECEIVE BA/CC
3243					
3244	022356	152711	000044		BISB #44,(R1) ;REC BA/CC REQUEST
3245	022362	105711		1\$:	TSTB (R1) ;RDY I SET?
3246	022364	100376			BPL 1\$;BR IF NO
3247	022366	012561	000004		MOV (R5)+,4(R1) ;LOAD REC BA
3248	022372	012561	000006		MOV (R5)+,6(R1) ;LOAD REC CC
3249	022376	142711	000040		BICB #40,(R1) ;CLEAR RQI
3250	022402	105711		2\$:	TSTB (R1) ;IS RDY I CLEAR
3251	022404	100776			BMI 2\$;BR IF NO
3252	022406	000205			RTS R5 ;RETURN
3253					
3254	022410				XFRELD:
3255					;THIS SUBROUTINE LOADS THE KMC WITH A TRANSMIT BA/CC
3256					
3257	022410	152711	000040		BISB #40,(R1) ;XMIT BA/CC REQUEST
3258	022414	105711		1\$:	TSTB (R1) ;RDY I SET?
3259	022416	100376			BPL 1\$;BR IF NO
3260	022420	012561	000004		MOV (R5)+,4(R1) ;LOAD XMIT BA
3261	022424	012561	000006		MOV (R5)+,6(R1) ;LOAD XMIT CC
3262	022430	142711	000040		BICB #40,(R1) ;CLEAR RQI
3263	022434	105711		2\$:	TSTB (R1) ;IS RDY I CLEAR
3264	022436	100776			BMI 2\$;BR IF NO
3265	022440	000205			RTS R5 ;RETURN

SUBROUTINES

```

3266
3267
3268 022442
3269
3270
3271 022442 042761 000207 000002
3272 022450 152711 000046
3273 022454 105711
3274 022456 100376
3275 022460 142711 000040
3276 022464 105761 000002
3277 022470 100375
3278 022472 000207
3279 022474 012711 040000
3280 022500 042711 140000
3281 022504 005000
3282 022506 004737 022614
3283 022512 005011
3284 022514 010061 000004
3285 022520 012261 000006
3286 022524 012711 002000
3287 022530 012711 022000
3288 022534 005200
3289 022536 022700 002000
3290 022542 003363
3291 022544 004737 022614
3292 022550 005000
3293 022552 005011
3294 022554 010061 000004
3295 022560 012711 002000
3296 022564 026122 000006
3297 022570 001402
3298 022572 104401 011054
3299 022576 005200
3300 022600 022700 002000
3301 022604 003362
3302 022606 012711 040000
3303 022612 000207
3304
3305 022614 012702 023650
3306 022620 032737 000002 002054
3307 022626 001402
3308 022630 012702 027654
3309 022634 000207
3310
3311 023650
3312
  
```

SHUTDOWN: ;THIS SUBROUTINE FORCES THE KMC TO UPDATE THE BASEMC TABLE

```

      BIC #207,2(R1) ;CLEAR ANY OUTPUT DONES
      BISB #46,(R1) ;ASK FOR ILLEGAL REQUEST
1$:   TSTB (R1) ;RDI SET?
      BPL 1$ ;BR IF NO
      BICB #40,(R1) ;CLEAR RQI
2$:   TSTB 2(R1) ;OUTPUT DONE SET?
      BPL 2$ ;BR IF NOT
      RTS PC ;RETURN
LDRVRF: MOV #BIT14,(R1) ;MASTER CLEAR KMC-11.
      BIC #BIT15!BIT14,(R1) ;AND SHUT IT DOWN.
      CLR R0 ;CLEAR UPC POINTER.
      JSR PC,SETMAP ;SET MICRO-CODE POINTER IN R2.
3$:   CLR (R1) ;START WITH THE CLEAN WORLD.
      MOV R0,4(R1) ;LOAD CRAM ADDRESS.
      MOV (R2)+,6(R1) ;LOAD INSTRUCTION WORD.
      MOV #BIT10,(R1) ;SET ROM 0.
      MOV #BIT13!BIT10,(R1) ;WRITE IT...
      INC R0 ;UPDATE UPC POINTER.
      CMP #2000,R0 ;OVER FLOW?
      BGT 3$ ;BR IF NO.
VERFY: JSR PC,SETMAP ;SET MICRO-CODE POINTER IN R2.
      CLR R0 ;SET UPC POINTER.
6$:   CLR (R1) ;START WITH THE CLEAN WORLD.
      MOV R0,4(R1) ;LOAD CRAM ADDRESS.
      MOV #BIT10,(R1) ;SET ROM 0.
      CMP 6(R1),(R2)+ ;CHECK IF RIGHT?
      BEQ 9$ ;BR IF GOOD.
      TYPE ,MLDER ;LOADING ERROR.
9$:   INC R0 ;BUMP UPC POINTER.
      CMP #2000,R0 ;IS IT DONE?
      BGT 6$ ;BR IF NO.
      MOV #BIT14,(R1) ;MASTER CLEAR KMC-11.
      RTS PC ;RETURN.
SETMAP: MOV #LOMAP,R2 ;LOAD ADDRESS OF LOW SPEED.
      BIT #BIT1,STAT3 ;IS IT HIGH SPEED?
      BEQ 3$ ;BR IF NO.
3$:   MOV #HIMAP,R2 ;LOAD HIGH SPEED ADDRESS.
      RTS PC ;RETURN TO CALLER.
LOMAP: ;LOW SPEED (REMOTE) MICRO-CODE.
  
```

SUBROUTINES

3313 027654
3314
3315 027654
3316

. =27654

HIMAP:

;HIGH SPEED (LOCAL) MICRO-CODE.

SUBROUTINES

3317

033654	052200	040522	051516	EM2:	.ASCIZ	<200>/TRANSMIT BA ERROR/
033677	200	051124	047101	EM3:	.ASCIZ	<200>/TRANSMIT COUNT ERROR/
033725	200	042522	042503	EM4:	.ASCIZ	<200>/RECEIVE BA ERROR/
033747	200	042522	042503	EM5:	.ASCIZ	<200>/RECEIVE COUNT ERROR/
033774	051200	041505	044505	EM11:	.ASCIZ	<200>/RECEIVE DATA ERROR/
034020	043200	042522	020105	EM12:	.ASCIZ	<200>/FREE RUNNING ERROR/
034044	041600	047117	051124	EM13:	.ASCIZ	<200>/CONTROL OUT ERROR/
034067	200	047111	042524	EM14:	.ASCIZ	<200>/INTERNAL DDCMP ERROR COUNTS NON ZERO/
034135	200	054105	042520	DH1:	.ASCIZ	<200>/EXPECTED FOUND ADDRESS/
034167	200	054105	042520	DH2:	.ASCIZ	<200>/EXPECTED FOUND/
034210	020200	042523	032114	DH3:	.ASCIZ	<200>/ SEL4 SEL6/
034231	200	040502	042523	DH4:	.ASCIZ	<200>/BASEMC+3 THRU BASEMC+12 /
034263	200	046513	030503	DH5:	.ASCIZ	<200>/KMC11 IS HUNG/
						.EVEN
034302	000003			DT1:	3	
034304	006	004			.BYTE	6,4
034306	001266				\$REG2	
034310	006	004			.BYTE	6,4
034312	001272				\$REG4	
034314	004	002			.BYTE	4,2
034316	001262				\$REG0	
034320	000003			DT2:	3	
034322	006	004			.BYTE	6,4
034324	001274				\$REG5	
034326	006	004			.BYTE	6,4
034330	001272				\$REG4	
034332	004	002			.BYTE	4,2
034334	001266				\$REG2	
034336	000003			DT3:	3	
034340	006	004			.BYTE	6,4
034342	001274				\$REG5	
034344	006	004			.BYTE	6,4
034346	001272				\$REG4	
034350	004	002			.BYTE	4,2
034352	001302				\$TMP2	
034354	000002			DT4:	2	
034356	003	007			.BYTE	3,7
034360	001274				\$REG5	
034362	003	002			.BYTE	3,2
034364	001272				\$REG4	
034366	000002			DT5:	2	
034370	006	004			.BYTE	6,4
034372	001274				\$REG5	
034374	006	002			.BYTE	6,2
034376	001272				\$REG4	
034400	000003			DT6:	3	
034402	003	010			.BYTE	3,10
034404	001274				\$REG5	
034406	003	004			.BYTE	3,4
034410	001272				\$REG4	
034412	004	002			.BYTE	4,2
034414	021304				FLAG	
034416	000003			DT7:	3	

SUBROUTINES

034420	003	010		.BYTE	3,10
034422	001274			\$REG5	
034424	003	004		.BYTE	3,4
034426	001272			\$REG4	
034430	004	002		.BYTE	4,2
034432	001266			\$REG2	
034434	000003		DT10:	3	
034436	003	007		.BYTE	3,7
034440	001274			\$REG5	
034442	003	004		.BYTE	3,4
034444	001272			\$REG4	
034446	006	002		.BYTE	6,2
034450	001302			\$TMP2	
034452	000002		DT11:	2	
034454	006	004		.BYTE	6,4
034456	001302			\$TMP2	
034460	006	002		.BYTE	6,2
034462	001304			\$TMP3	
034464	000010		DT12:	10	
034466	003	002		.BYTE	3,2
034470	001300			\$TMP1	
034472	003	002		.BYTE	3,2
034474	021434			BASEMC+4	
034476	003	002		.BYTE	3,2
034500	001302			\$TMP2	
034502	003	002		.BYTE	3,2
034504	021436			BASEMC+6	
034506	003	002		.BYTE	3,2
034510	001304			\$TMP3	
034512	003	002		.BYTE	3,2
034514	021440			BASEMC+10	
034516	003	002		.BYTE	3,2
034520	001306			\$TMP4	
034522	003	002		.BYTE	3,2
034524	021442			BASEMC+12	
034526	000002		DT13:	2	
034530	006	004		.BYTE	6,4
034532	001274			\$REG5	
034534	006	002		.BYTE	6,2
034536	001272			\$REG4	
034540			CORMAX:		
	000001		.END		

SYMBOL TABLE

ABASE = 000000	AUSTRT 003126	CONN 010665	ERCT05 002330	KMNUM 001472
ACDW1 = 000000	AUSWR = 000000	CONTAB 003330	ERCT06 002334	KMPO4 002074
ACDW2 = 000000	AUTO.S 012124	CONVRT= 104416	ERCT07 002340	KMPO6 002076
ACPUOP= 000000	AVECT1= 000000	CORMAX 034540	ERCT10 002344	KMRLVL 002060
ADDW0 = 000000	AVECT2= 000000	CR = 000015	ERCT11 002350	KMRVEC 002056
ADDW1 = 000000	BASE 013756	CREAM 001502	ERCT12 002354	KMS100 002102
ADDW10= 000000	BASELD 022030	CRLF = 000200	ERCT13 002360	KMS101 002112
ADDW11= 000000	BASELH 022136	CSR 010357	ERCT14 002364	KMS102 002122
ADDW12= 000000	BASEMC 021430	CSRMAP 012126	ERCT15 002370	KMS103 002132
ADDW13= 000000	BINWRD 006406	CYCLE 011474	ERCT16 002374	KMS104 002142
ADDW14= 000000	BIT0 = 000001	DATABP 006760	ERCT17 002400	KMS105 002152
ADDW15= 000000	BIT00 = 000001	DATACL= 104413	ERR 003244	KMS106 002162
ADDW2 = 000000	BIT01 = 000002	DATAHD 006746	ERRMSG 006734	KMS107 002172
ADDW3 = 000000	BIT02 = 000004	DDISP = 177570	ERRPC 003322	KMS110 002202
ADDW4 = 000000	BIT03 = 000010	DELAY = 104411	ERRVEC= 000004	KMS111 002212
ADDW5 = 000000	BIT04 = 000020	DEVTAB 003342	ERTAB0 007106	KMS112 002222
ADDW6 = 000000	BIT05 = 000040	DH1 034135	EXIT = 000205	KMS113 002232
ADDW7 = 000000	BIT06 = 000100	DH2 034167	EXITER 007042	KMS114 002242
ADDW8 = 000000	BIT07 = 000200	DH3 034210	FLAG 021304	KMS115 002252
ADDW9 = 000000	BIT08 = 000400	DH4 034231	FLOAT 003156	KMS116 002262
ADEVCT= 000000	BIT09 = 001000	DH5 034263	FY 003202	KMS117 002272
ADEVN = 000000	BIT1 = 000002	DISPLA 001242	HALTS 006764	KMS200 002104
ADRCNT 006053	BIT10 = 002000	DISPRE 000174	HILIM 006046	KMS201 002114
ADVANC= 104420	BIT11 = 004000	DSWR = 177570	HIMAP 027654	KMS202 002124
AENV = 000002	BIT12 = 010000	DT1 034302	HT = 000011	KMS203 002134
AENVN = 000000	BIT13 = 020000	DT10 034434	IISR 017132	KMS204 002144
AFATAL= 000000	BIT14 = 040000	DT11 034452	INCHAR 011434	KMS205 002154
AMADR1= 000000	BIT15 = 100000	DT12 034464	INIFLG 001506	KMS206 002164
AMADR2= 000000	BIT2 = 000004	DT13 034526	INLP1 005762	KMS207 002174
AMADR3= 000000	BIT3 = 000010	DT2 034320	INPUT = 104415	KMS210 002204
AMADR4= 000000	BIT4 = 000020	DT3 034336	INTTY 013506	KMS211 002214
AMAMS1= 000000	BIT5 = 000040	DT4 034354	IOTVEC= 000020	KMS212 002224
AMAMS2= 000000	BIT6 = 000100	DT5 034366	KMACTV 001470	KMS213 002234
AMAMS3= 000000	BIT7 = 000200	DT6 034400	KMCM 011044	KMS214 002244
AMAMS4= 000000	BIT8 = 000400	DT7 034416	KMCR00 002100	KMS215 002254
AMSGAD= 000000	BIT9 = 001000	DZDMH = 000000	KMCR01 002110	KMS216 002264
AMSGLG= 000000	BM 010625	EMTVEC= 000030	KMCR02 002120	KMS217 002274
AMSGTY= 000000	BPTVEC= 000014	EM11 033774	KMCR03 002130	KMS300 002106
AMTYP1= 000000	BRLVL 013472	EM12 034020	KMCR04 002140	KMS301 002116
AMTYP2= 000000	BRW 004354	EM13 034044	KMCR05 002150	KMS302 002126
AMTYP3= 000000	CHRCNT 006404	EM14 034067	KMCR06 002160	KMS303 002136
AMTYP4= 000000	CKSWR 011226	EM2 033654	KMCR07 002170	KMS304 002146
APASS = 000000	CKSWR1 011302	EM3 033677	KMCR10 002200	KMS305 002156
APRIOR= 000000	CKSWR2 011314	EM4 033725	KMCR11 002210	KMS306 002166
APTCSU= 000040	CKSWR3 011320	EM5 033747	KMCR12 002220	KMS307 002176
APTENV= 000001	CKSWR4 011324	ENDEX 017062	KMCR13 002230	KMS310 002206
APTSIZ= 000200	CKSWR5 011430	ENDEX1 017076	KMCR14 002240	KMS311 002216
APTSPO= 000100	CLKX 001452	ENDEX2 017100	KMCR15 002250	KMS312 002226
APT.SI 013540	CLRTAB 016566	ERCT00 002304	KMCR16 002260	KMS313 002236
ASWREG= 000000	CNERR 011023	ERCT01 002310	KMCR17 002270	KMS314 002246
ATESTN= 000000	CNT.MA 002302	ERCT02 002314	KMCSR 002066	KMS315 002256
AUDONE 003354	CNVRT = 104417	ERCT03 002320	KMCSRH 002070	KMS316 002266
AUNIT = 000000	CONERR 010756	ERCT04 002324	KMCTL 002072	KMS317 002276

SYMBOL TABLE

KMTLVL	002064	PACT13	002356	RFIELD	022356	SW8	= 000400	X3	= 000103
KMTVEC	002062	PACT14	002362	ROMCLK=	104412	SW9	= 001000	X4	= 000104
KM.END	002300	PACT15	002366	RUN	001500	TBITVE=	000014	X5	= 000105
KM.MAP	002100	PACT16	002372	R6	=%000006	TBUF	021314	X6	= 000106
LDRVRF	022474	PACT17	002376	R7	=%000007	TBUFF	020110	X7	= 000107
LF	= 000012	PARBIT=	040000	SAVACT	001474	TCOUNT	021312	ZERO	001462
LINE	010567	PERFOR=	004537	SAVNUM	001476	TEMP	011122	\$APTHD	002034
LOBITS	006052	PFTAB	007320	SAVPC	001460	TFLAG	021306	\$ATYC	004716
LOCK	001444	PIRQ	= 177772	SAVSP	001456	TIMER	= 104414	\$ATY1	004672
LOKFLG	001510	PIRQVE=	000240	SAV05	= 104406	TKVEC	= 000060	\$ATY3	004700
LOLIM	006044	POP0	= 012600	SCAN	016716	TLAST	= 016370	\$ATY4	004710
LOMAP	023650	POP1SP=	005726	SCAN1	017102	TPVEC	= 000064	\$AUTOB	001234
MASKX	001454	POP2SP=	022626	SCAN2	016730	TRAPVE=	000034	\$BASE	001372
MASTEK	010011	PRI0	010416	SCAN1	017102	TRTVEC=	000014	\$BDADR	001222
MCSRX	007741	PRI1	013760	SCOPI	= 104405	TST1	013762	\$BDDAT	001226
MDATA	011164	PRI2	= 000000	SETMAP	022614	TST10	016056	\$CDW1	001376
MEMLIM	001466	PRI3	= 000040	SHUTDO	022442	TST11	016156	\$CDW2	001400
MEPASS	007614	PRI4	= 000100	SOFTSW	011466	TST12	016370	\$CHARC	004666
MERRPC	010066	PRI5	= 000140	SPACNT=	006405	TST2	015006	\$CMTAG	001200
MERRX	007766	PRI6	= 000200	STACK	= 001200	TST3	015200	\$CM1	= 000006
MERR2	007641	PRI7	= 000240	STAT	001450	TST4	015336	\$CM2	= 000014
MERR3	007666	PR0	= 000300	STAT1	002050	TST5	015464	\$CM3	= 000006
MILK	001504	PR1	= 000340	STAT2	002052	TST6	015622	\$CM4	= 000005
MLDER	011054	PR2	= 177776	STAT3	002054	TST7	015740	\$CNTLG	005530
MLOCK	007712	PS	= 177776	STKLMT=	177774	TTST	004146	\$CNTLU	005523
MNEW	010013	PSW	= 177776	STRTSW	001446	TWOSYN=	010000	\$CPUOP	001344
MODU	010455	PUSHRO=	010046	SV05	006074	TYPDAT	006750	\$CRAP	= 177777
MPASSX	007755	PUSH1S=	005746	SWFLG	011432	TYPE	= 104401	\$CRLF	001313
MPFAIL	007556	PUSH2S=	024646	SWR	001240	TYPMMSG	006650	\$DDW0	001402
MR	007636	PWRVEC=	000024	SWREG	000176	VEC	010375	\$DDW1	001404
MRESET=	004000	QV.FLG	001511	SW0	= 000001	VECMAP	013222	\$DDW10	001426
MSTCLR=	104410	RBUF	021362	SW00	= 000001	VECTR	013754	\$DDW11	001430
MTITLE	001000	RBUFF	020346	SW01	= 000002	VERFY	022544	\$DDW12	001432
MTSTN	007777	RBUFFE	021302	SW02	= 000004	WHAT	005764	\$DDW13	001434
MVECX	007747	RBUFF1	020346	SW03	= 000010	WHERE	006050	\$DDW14	001436
NEXT	001442	RBUFF2	020452	SW04	= 000020	WHICH	013214	\$DDW15	001440
NOACT	010725	RBUFF3	020556	SW05	= 000040	WRDCNT	006402	\$DDW2	001406
NODEV	003240	RBUFF4	020662	SW06	= 000100	WRKO.F	006736	\$DDW3	001410
NUM	010317	RBUFF5	020766	SW07	= 000200	XBX	006536	\$DDW4	001412
OISR	017422	RBUFF6	021072	SW08	= 000400	XCNTAB	020240	\$DDW5	001414
OK	003220	RBUFF7	021176	SW09	= 001000	XCSR	004104	\$DDW6	001416
ONE	001464	RCNTAB	020220	SW1	= 000002	XDNTAB	020312	\$DDW7	001420
PACT00	002302	RCOUNT	021360	SW10	= 002000	XERR	004126	\$DDW8	001422
PACT01	002306	RDCHR	= 104402	SW11	= 004000	XFRELD	022410	\$DDW9	001424
PACT02	002312	RDLIN	= 104403	SW12	= 010000	XHEAD	010073	\$DEVCT	001326
PACT03	002316	RDNTAB	020256	SW13	= 020000	XMITBA	020052	\$DEVVM	001374
PACT04	002322	RDOCT	= 104404	SW14	= 040000	XPASS	004120	\$DOAGN	004100
PACT05	002326	RECBA	020052	SW15	= 100000	XSTATQ	011074	\$ENDAD	004070
PACT06	002332	RESREG	006762	SW2	= 000004	XTSTN	007114	\$ENDCT	004054
PACT07	002336	RESUM	022246	SW3	= 000010	XVEC	004112	\$ENV	001336
PACT10	002342	RESUME	020050	SW4	= 000020	X0	= 000110	\$ENVVM	001337
PACT11	002346	RESVEC=	000010	SW5	= 000040	X1	= 000101	\$EOP	003662
PACT12	002352	RES05	= 104407	SW6	= 000100	X2	= 000102	\$EOPCT	004046
		RFLAG	021310	SW7	= 000200				

SYMBOL TABLE

\$ERFLG	001203	\$LPADR	001206	\$NWTST=	000000	\$SVPC	= 000040	\$TYPEC	004622
\$ERMAX	001215	\$LPERR	001210	\$OVER	004330	\$SWR	= 164000	\$TYPEX	004670
\$ERROR	006506	\$MADR1	001350	\$PASS	001324	\$SWREG	001340	\$UNIT	001330
\$ERRPC	001216	\$MADR2	001354	\$PASTM	002042	\$SWRMK=	000000	\$UNITM	002044
\$ERRTB	001512	\$MADR3	001360	\$PWRDN	007122	\$TESTN	001322	\$USWR	001342
\$ERTTL	001212	\$MADR4	001364	\$PWRMG	007306	\$TIMES	001310	\$VECT1	001366
\$ETABL	001336	\$MAIL	001316	\$PWRUP	007174	\$TKB	001246	\$VECT2	001370
\$ETEND	001442	\$MAMS1	001346	\$QUES	001312	\$TKS	001244	\$XTSTR	004174
\$FATAL	001320	\$MAMS2	001352	\$RDCHR	005140	\$TMP0	001276	\$Y	= 000000
\$FFLG	005136	\$MAMS3	001356	\$RDLIN	005260	\$TMP1	001300	\$\$GET4=	000000
\$FILLC	001256	\$MAMS4	001362	\$RDOCT	005560	\$TMP2	001302	.	= 034540
\$FILLS	001255	\$MBADR	002036	\$RDSZ =	000007	\$TMP3	001304	.ADVAN	006054
\$GDADR	001220	\$MFLG	005134	\$REGAD	001260	\$TMP4	001306	.BEGIN	003464
\$GDDAT	001224	\$MNEW	005546	\$REG0	001262	\$TN	= 000013	.CNVRT	006164
\$GET42	004060	\$MSGAD	001332	\$REG1	001264	\$TPB	001252	.CONVR	006160
\$HD =	000000	\$MSGLG	001334	\$REG2	001266	\$TPFLG	001257	.DATAC	007442
\$HIBTS	002034	\$MSGTY	001316	\$REG3	001270	\$TPS	001250	.DELAY	007326
\$HIOCT	005716	\$MSWR	005535	\$REG4	001272	\$TRAP	006410	.MSTCL	007356
\$ICNT	001204	\$SMTYP1	001347	\$REG5	001274	\$TRAP2	006432	.RES05	006126
\$ILLUP	007312	\$SMTYP2	001353	\$RTNAD	004102	\$TRP =	000021	.ROMCL	007374
\$INPUT	005720	\$SMTYP3	001357	\$S	= 000014	\$TRPAD	006444	.SAV05	006066
\$INTAG	001235	\$SMTYP4	001363	\$SAVR6	007316	\$STSM	002040	.SCOPI	004360
\$ITEMB	001214	\$MXCNT	004356	\$SCOPE	004134	\$STSTM	001202	.START	002402
\$LF	001314	\$N =	000012	\$SETUP=	000000	\$TTYIN	005514	.TIMER	007506
\$LFLG	005135	\$NULL	001254	\$SVLAD	004316	\$TYPE	004410	.\$X	= 002034

. ABS. 034540 000

ERRORS DETECTED: 0

DSKZ:CZKCGA,DSKZ:CZKCGA/SOL=CZKCGA.MAC,CZKCGA.P11/EQ:DZDMH
RUN-TIME: 29 18 .7 SECONDS
RUN-TIME RATIO: 122/48=2.5
CORE USED: 46K (91 PAGES)