LPA11 LPA/AA11-K TEST AH-BO23B-MC FEB 1981
CRLPBBO FICHE 1 OF 1

IDENTIFICATION

PRODUCT CODE:	AC-B022B-MC
DIAG. CODE	MAINDEC-11-CRLPB-B
PRODUCT NAME:	CPLPBB0 LPA/AA11-K TEST
DATE:	DEC 1980
MAINTAINER:	DIAGNOSTIC GROUP

1. ABSTRACT

THIS DIAGNOSTIC EXERCISES THE "AA11-K" ANALOG CIRCUITRY. THE PROGRAM
WHEN STARTED WILL TYPE OUT THE PROGRAM TITLE.
A MESSAGE IS THEN PRINTED GIVING THE TWO LETTER DESIGNATOR
TO BE TYPED TO RUN ANY ONE OF THE FIVE (5) SEPERATE TESTS OF WHICH
THIS PROGRAM IS COMPRISED. THE PROGRAM THEN TYPES A 'CR .' AND
THEN WAITS IN A KEYBOARD MONITOR MODE FOR TWO LETTER'S TO BE TYPED.
ALTHOUGH THESE TESTS M'Y BE RUN IN ANY ORDER IT IS IMPERATIVE
THAT TEST "AL" IS RUN FIRST AND VERIFY THAT THE AA11-K IS FULLY OPERATIONAL.
THE PROGRAM IS SET UP TO GIVE THE OPERATOR AS MUCH CONTROL
OVER THE PROGRAM AS POSSIBLE VIA THE TELETYPE.TYPING A '^C'
(OBTAINED VIA TYPING T'E 'CNTR' AND 'C' KEYS SIMULTANEOUSLY)
WHILE RUNNING ANY TES, WILL ENABLE THE PROGRAM TO RETURN TO
THE KEYBOARD MONITOR AND AWAIT A NEW LETTER DESIGNATOR TO BE
TYPED. TYPING A '^G' WHILE RUNNING WILL ENABLE THE SOFTWARE SWITCH
REGISTER VALUE TO BE CHANGED.

THIS PROGRAM IS A MODIFIED VERSION OF 'MD-11-DZAAC-B". IT WAS
MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE AA11-K OPTION
WHEN IT IS ON THE LPA11KX I/O BUS. NO RECABLING IS NEEDED.
SOME TEST DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION
TEST, WERE DELETED AS THEY COULD NOT BE CHECKED. IF THIS
DIAGNOSTIC DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO
RUN 'MD-11-DZAAC-B". YOU SHOULD RUN 'MD-11-CRLPA" BEFORE
RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 10.

2. REQUIREMENTS (EQUIPMENT)

A. PDP-1' COMPUTER WITH 16K OF MEMORY AND A CONSOLE I/O TERMINAL
B. AA11-K QUAD OPTION MODULE INSTALLED
C. <OPTIONAL> VR14 OR STORAGE SCOPE

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING BINARY TAPES.

4. STARTING PROCEDURE

THE PROGRAM STARTING ADDRESS IS '200'.
THE RESTART ADDRESS IS '204'.

5. CONSOLE SWITCH SETTINGS

THIS PROGRAM HAS BEEN MODIFIED TO RUN WITH OR
WITHOUT A HARDWARE SWITCH REGISTER.
A. ALL SWITCHES SHOULD BE DOWN (0) WHEN THE PROGRAM IS STARTED.
B. REFER TO THE INDIVIDUAL TE, DESCRIPTIONS FOR APPLICABLE
   CONSOLE SWITCH SETTINGS
C. ALL SWITCHES SET TO A 1 WILL SELECT SOFTWARE SWITCH CONTROL.

6. ERRORS

ALL ERRORS ARE ACCOMPANIED WITH A ENGLISH DESCRIPTIVE
COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE
ERROR CAN BE OBTAINED FROM THE COMMENT AT THE ERROR PC OR THE TEST ITSELF.

7.0 TEST PROCEDURE

7.1 AUTO LOGIC TEST

A. THIS TEST IS DESIGNED TO VERIFY THE DATA PATH THAT IS ADDRESSABLE
FROM THE CPU. THIS ALSO INCLUDES ALL CONTROL, INTERRUPT AND
INITILIZE SIGNALS. THE LOGIC TEST ALSO INCLUDES PROVISIONS
FOR TESTING MULTIPLE AA11-K'S.

B. STARTING SEQUENCE

1. TYPE 'AL' TO RUN THE AUTO LOGIC TEST.
2. THE PROGRAM WILL THEN EXECUTE A LOGIC TEST ON ALL AVAILABLE UNITS

C. CONTROL SWITCHES

1. TYPING ^C WILL ENABLE THE PROGRAM TO EXIT AND RETURN
TO THE KEYBOARD MONITOR.
2. TYPING ^G WHEN SOFTWARE SWITCH REGISTER IS ENABLED WILL
REPORT LAST VALUE AND WAIT FOR NEW VALUE.

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON CURRENT TEST
SW13=1	020000	INHIBIT ERROR TYPEOUT
SW12=1	010000	STORAGE SCOPE CONNECTED
SW10=1	002000	EXTERNAL DELAY SIGNAL CONNECTED
SW09=1	001000	TWO'S COMPLIMENT MODE
SW08=1	0004XX	LOOP ON TEST IN SWR 7:0

D. ERRORS

REF. TO 6.

E. RESTRICTIONS

IF A STORAGE SCOPE IS CONNECTED, POWER MUST BE APPLIED TO IT.

F. EXECUTION TIME

IT TAKES APPROXIMATELY 25 SECONDS PER PASS.

7.2 AUTO DISPLAY TEST

A. THIS TEST IS DESIGNED TO AID IN THE ADJUSTING AND ALIGNMENT OF THE VR14 OR STORAGE SCOPE SCOPE ON THE AA11-K DISPLAY CONTROL.

B. TYPE 'AD' TO RUN THE AUTO VISUAL DISPLAY TEST.
THE PROGRAM WILL THEN EXECUTE THE VISUAL DISPLAY TEST.

C. CONTROL SWITCHES

1. TYPING ^C AT ANY TIME WILL ENABLE THE PROGRAM TO EXIT AND RETURN TO THE MONITOR.
2. TYPING ^G WHEN SOFTWARE SWITCH REGISTER IS ENABLED WILL ENABLE THE PROGRAM TO CHANGE THE SOFTWARE SWITCH REGISTER VALUE.

CONSOLE SWITCHES	FUNCTION
CONSOLE SW10=1	SELECTS EXTERNAL DELAY MODE
CONSOLE SW09=1	SELECTS TWO'S COMPLEMENT MODE
CONSOLE SW08=0	CYCLE THRU ALL FOUR DISPLAY PATTERNS
CONSOLE SW08=1	SELECT PATTERNS IN SW 00-02
CONSOLE SW03=0	SELECT DAC 0 AND DAC 1
CONSOLE SW03=1	SELECT DAC 2 AND DAC 3
CONSOLE SW00-02=0	DISPLAY A HORIZONTAL LINE
CONSOLE SW00-02-1	DISPLAY A VERTICAL LINE
CONSOLE SW00-02=2	DISPLAY A SQUARE
CONSOLE SW00-02=3	DISPLAY AN "X"

D. ERRORS

THE ONLY ERRORS IN THIS TEST ARE DETECTED VISUALLY.

E. RESTRICTIONS

IF VR14, CHANNEL SWITCH MUST BE SET TO "1 & 2" POSITION.
IF STORAGE SCOPE, POWER MUST BE APPLIED.
THE "AUTO-CALIBRATION (AC)" MUST HAVE BEEN RUN PRIOR TO RUNNING THIS SECTION WHEN ON THE "AUTO HARDWARE TESTER".

F. EXECUTION TIME

S IT TAKES APPROXIMATELY 1 MINUTE TO COMPLETE THIS TEST BEFORE IT REPEATS.

7.4 MANUAL LOGIC LOOP

A. THIS LOOP IS PROVIDED TO ENABLE THE OPERATOR A SIMPLE PROGRAM LOOP TO AID IN REPAIR OF THE AA11-K. SWITCH REGISTER BITS 15:13 SELECT THE REGISTER TO BE LOADED AND BITS 12:00 CONTAINS THE DATA TO BE LOADED.

B. STARTING SEQUENCE

1. TYPE 'ML' TO RUN THE MANUAL LOGIC LOOP.
2. THE PROGRAM WILL NOW LOOP AND LOAD THE VALUE OF THE SWITCH REGISTER BITS 12:00 INTO THE SELECTED AA11-K REGISTER.

C. CONTROL SWITCHES

SW15:13	REGISTER SELECTED
000	DAC #0
001	DAC #1
010	DAC #2
011	DAC #3
1XX	STATUS REGISTER

D. ERRORS

NO PROVISIONS ARE MADE FOR LOGIC ERRORS.

E. RESTRICTIONS

NONE

F. EXECUTION TIME

THIS IS A NON-ENDING PROGRAM LOOP THAT CAN BE EXITED BY TYPING ^C.

7.5 MANUAL DISPLAY LOOP

A. THIS LOOP IS PROVIDED FOR THE OPERATOR TO VERIFY OPERATION OF THE D/A CONVERTER AND MULTIPLEXER.

B. STARTING SEQUENCE

1. TYPE 'MD' TO RUN MANUAL DISPLAY LOOP
2. THE PROGRAM WILL NOW LOAD A "RAMP PATTERN" INTO EACH D/A CONVERTER.

C. CONTROL SWITCHES

1. TYPING ^C WILL RETURN CONTROL TO THE KEYBOARD MONITOR.
2. SW10=1 WILL SELECT EXTERNAL DELAY MODE.

D. ERRORS

NO PROVISIONS ARE MADE FOR LOGIC ERRORS.

7.6 MANUAL CALIBRATION LOOP

A. THIS LOOP IS PROVIDED TO ENABLE THE OPERATOR TO ADJUST THE D/A CONVERTER.

B. STARTING SEQUENCE

1. TYPE 'MC' TO RUN THE MANUAL CALIBRATION LOOP.
2. THE PROGRAM WILL NOW LOAD THE CONTENTS OF THE SWITCH REGISTER INTO EACH D/A REGISTER AND AFTER A DELAY CLEAR THE D/A REGISTER.

C. CONTROL SWITCHES

1. TYPING ^C WILL EXIT THE LOOP AND RETURN TO THE KEYBOARD MONITOR.
2. TYPING ^G WHEN SOFTWARE SWITCH REGISTER IS ENABLED TO CHANGE THE SWITCH REGISTER VALUE
3. SWITCH REGISTER BITS 11:0 ARE LOADED IN TO ALL DAC'S.

D. ERRORS

NO PROVISIONS ARE MADE FOR LOGIC ERRORS.

8. AUTO DISPLAY TEST PATTERN DESCRIPTIONS

DISPLAY HORIZONTAL LINE

A HORIZONTAL LINE IS DISPLAYED ON THE SCOPE BY INITIALLY SETTING THE X AND Y DAC'S TO ZERO AND THEN INCREMENTING THE X VALUE WHILE HOLDING THE Y VALUE CONSTANT.

DISPLAY VERTICAL LINE

A VERTICAL LINE IS DISPLAYED ON THE SCOPE IN THE SAME MANNER AS FOR A HORIZONTAL LINE EXCEPT NOW THE Y VALUE IS INCREMENTED WHILE HOLDING THE X VALUE CONSTANT.

DISPLAY SQUARE

A SQUARE IS DISPLAYED BY INITIALLY SETTING THE X AND Y VALUES TO NEGATIVE FULL SCALE, THEN X IS INCREMENTED TO POSITIVE SCALE (BOTTOM LINE) THEN Y IS INCREMENTED TO POSITIVE FULL SCALE (RIGHT LINE) THEN X IS DECREMENTED TO NEGATIVE FULL SCALE (TOP LINE) AND FINALLY Y IS DECREMENTED TO NEGATIVE FULL SCALE (LEFT LINE). MODE 01 (INTENSIFY ON LOADING X) AND MODE 10 (INTENSIFY ON LOADING Y) ARE USED.

DISPLAY X

AN X IS DISPLAYED BY INITIALLY SETTING THE X AND Y VALUES TO NEGATIVE FULL SCALE AND THEN INCREMENTING BOTH TO POSITIVE FULL SCALE (LOWER LEFT TO UPPER RIGHT DIAGONAL) THEN X IS RESET TO NEGATIVE FULL SCALE, Y REMAINS AT POSITIVE FULL SCALE AND THEN X IS INCREMENTED WHILE Y IS DECREMENTED UNTIL BOTH REACH FULL SCALE AGAIN (UPPER LEFT TO LOWER RIGHT DIAGONAL). MODE 01 (INTENSIFY ON LOADING X) IS USED.

9. MISCELLANEOUS

9.1 AA11-K BUS & VECTOR ADDRESS MODIFICATION

MODIFY LOCATION '$BASE' IF BASE ADDRESS IS NOT 170416.
MODIFY LOCATION '$VECT1' IF THE VECTOR AND PRIORITY IS NOT 100360.

*NOTE IF EITHER VALUE IS CHANGED, THE PROGRAM MUST BE RESTARTED AT 200.

9.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP.
THIS DIAGNOSTIC HAS THE "APT" HOOKS BUT HAS NOT BEEN TESTED.

9.3 POWER FAIL

A POWER FAIL WILL CAUSE A RESTART MESSAGE ON POWER UP AT
WHICH TIME THE PROGRAM IS RESTARTED.

9.4 MULTIPLE AA11-K INTERFACE TESTING

THE PROGRAM WILL "AUTO-SIZE" THE NUMBER OF AA11-K'S. THE PROGRAM
WILL REPORT THE NUMBER TO THE OPERATOR WHEN THE "AL" TEST IS
SELECTED THE FIRST TIME. IF THE OPERATOR WISHES TO INHIBIT
"AUTO-SIZE" BIT 15 OF LOCATION "$ENV" MUST BE SET.

9.5 RESTRICTION

POWER MUST BE APPLIED TO A STORAGE SCOPE IF CONNECTED.

9.6 EXECUTION TIME

1. EXECUTION TIME OF THE AUTO LOGIC TEST IS:
   25 SECONDS.
2. EXECUTION TIME OF THE AUTO DISPLAY TEST IS:
   60 SECONDS
3. EXECUTION TIME OF THE MANUAL LOGIC LOUP IS:
   OPERATOR DEPENDANT
4. EXECUTION TIME OF THE MANUAL DISPLAY LOOP IS:
   OPERATOR DEPENDANT
5. EXECUTION TIME OF THE MANUAL CALIBRATION LOOP IS:
   OPERATOR DEPENDANT

9.7 USER LINK TO I/O DEVICE

A SPECIAL USER LINK HAS BEEN PROVIDED IN ORDER FOR THE OPERATOR TO EXAMINE OR MODIFY LOCATIONS ON THE LPA11-KX I/O BUS. (NOTE: THIS CANNOT BE DONE DIRECTLY.)

PROCEDURE:

1) START THE PROCESSOR AT LOCATION $UTK: (214)

2) THE DIALOG TO EXAMINE A LOCATION IS AS FOLLOWS:

```
E OR D              "E"
DEVICE ADDRS=       "OCTAL ADDRS"
XXXXXX
```

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

3) THE DIALOG TO MODIFY A LOCATION IS AS FOLLOWS:

```
E OR D              "D"
DATA=               "DATA TO BE DEPOSITED"
```

4) THE PROGRAM WILL STAY IN THIS LOOP UNTIL THE OPERATOR IS FINISHED. AT THIS TIME THE PROCESSOR SHOULD BE HALTED.

NOTE: THE OPERATORS RESPONSE IS ENCLOSE IN QUOTES.

10. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY

DIAGNOSTICS FOR THE LPA11 ARE WRITTEN AT THREE LEVELS: (1) TOTAL PDP-11 SYSTEM, (2) LPA11 SYSTEM; AND, (3) LPA11 OPTIONS.

LEVEL 1, IS DESIGNED TO ISOLATE A FAILURE TO THE LPA11 SYSTEM. ALL OPTIONS ON THE PDP-11 ARE EXERCISED.

LEVEL 2 DIAGNOSTICS ISOLATE A FAILURE TO THE INDIVIDUAL OPTION WITHIN THE LPA11. THE LEVEL 2 DIAGNOSTIC IS MD-11-CRLPA. WHEN THE USER RUNS CRLPA HE CAN GENERALLY TELL WHICH OPTION DIAGNOSTIC (LEVEL 3) TO RUN NEXT. M8254 AND M8200-YC ERRORS MAY "LOOK" ALIKE AND CRLPA MAY NOT BE ABLE TO DISTINGUISH BETWEEN THEM. ARBITRATION ERRORS WILL NOT BE DETECTED BY THIS DIAGNOSTIC.

LEVEL THREE DIAGNOSTICS AID IN DETERMINING IF THE ERROR WAS IN FACT ON THE OPTION THE CRLPA SPECIFIED. THE USER MAY "LOOP" ON THE ERROR. WITHIN LEVEL THREE, THERE ARE TWO GROUPS OF DIAGNOSTICS. THE FIRST GROUP REQUIRES NO "EXTRA" WORK BY THE USER IN ORDER TO RUN. GROUP "A" DIAGNOSTICS DO NOT CHECK ARBITRATION, AND REQUIRE EXTRA TIME FOR EXECUTION. THE SECOND GROUP (GROUP 'B") REQUIRES THAT THE USER RECONFIGURE THE PDP-11 SYSTEM. THIS RECONFIGURATION INVOLVES CABLING THE UNIBUS TO THE LPA'S I/O BUS.

THE DIAGNOSTIC FOR THE M8254 FALLS INTO THE GROUP "B" CATAGORY.

THE LPA11-KX DIAGNOSTIC KIT WILL INCLUDE:

OPTION	GROUP	DIAG. #	DIAG. TITLE
LPA11-KX	LEVEL 2	MD-11-CRLPA	LPA11-K SYSTEM EXER.
M8254	"B"	MD-11-CRLPN	M8254 (IPBM) DIAG.
AA11-K	A	MD-11-CRLPB	LPA/AA11-K DIAG.
	B	MD-11-DZAAC	AA11-K DIAG.
AR11	A	MD-11-CRLPC	LPA/AR11 DIAG. #1
	A	MD-11-CRLPD	LPA/AR11 DIAG. #2
	A	MD-11-CRLPE	LPA/AR11 DIAG. #3
	B	MD-11-DZARA	AR11 DIAG. #1
	B	MD-11-DZARB	AR11 DIAG. #2
	B	MD-11-DZARC	AR11 DIAG. #3
DR11-K	A	MD-11-CRLPF	LPA/DR11-K DIAG.
	B	MD-11-DZDRG	DR11-K DIAG.
KW11-K	A	MD-11-CRLPG	LPA/KW11-K DIAG.
	B	MD-11-DZKWK	KW11-K DIAG.
LPS11	A	MD-11-CRLPH	LPA/LPS11 DIAG. #1
	A	MD-11-CRLPI	LPA/LPS11 DIAG. #2
	A	MD-11-CRLPJ	LPA/LPS11 DIAG. #3
	B	MD-11-DZLPC	LPS11 DIAG. #1
	B	MD-11-DZLPD	LPS11 DIAG. #2
	B	MD-11-DZLPI	LPS11 DIAG. #3
AD11-K	A	MD-11-CRLPK	LPA/AD11-K DIAG.
	B	MD-11-DZADL	AD11-K DIAG.
M8200-YC	B	MD-11-CRLPL	LPA/DMC-11 DIAG. TST I
	B	MD-11-CRLPM	LPA/DMC-11 DIAG. TST II

THIS IS A HISTORY FILE OF CRLPB-B

PRODUCT CODE: MAINDEC-11-DZAAC-B

PRODUCT NAME: AA11-K DIAGNOSTIC TEST

DATE: DECEMBER 1976

MAINTAINER: DIANOSTIC GROUP

****

PRODUCT CODE: MAINDEC-11-DRLPB-A

PRODUCT NAME: LPA/AA11-K DIAGNOSTIC TEST

DATE: JANUARY 1978

MAINTAINER: DIAGNOSTIC GROUP

REASON FOR DEVELOPMENT:

1) TO ENABLE THE OPERATOR TO CHECK OUT THE AA11-K OPTION
   WHEN IT IS ON THE LPA11-KX I/O BUS.

CHANGES MADE:

1) TOOK OUT CERTAIN TESTS FROM ORIGINAL DIAGNOSTIC (I.E.
   INTERRUPTS,TIME DEPENDENT CODE).

2) REPLACED DIRECT LINKS TO DEVICE WITH MACRO CALLS TO THE
   KMC-11 MICRO CODE. KMC-11 MICRO CODE (FILE:DRLPX2) HANDLES
   DIRECT COMMUNICATIONS WITH THE DEVICE.

FILE: DRLPA.MAC
      CONTAINS MACRO LINKS BETWEEN PDP-11 CODE AND KMC-11
      MICRO CODE. FILE: DRLPX2 NEEDS TO BE ASSEMBLED WITH
      DRLPB (SEE .CTL FILE).

FILE: DRLPX2
      MICRO CODE FILE THAT GETS LOADED INTO THE KMC-11
      VIA ROUTINES IN DRLPA.MAC.

      DRLPX2.P11 IS ASSEMBLED WITH MACY11 (ONLY) AS ANY OTHER
      .P11 FILE. THE RESULTS OF ITS ASSEMBLY IS A .OBJ
      MODULE AS WAS THE RESULT OF THE ASSEMBLY OF THE
      DIAGNOSTIC .P11 FILE. BOTH .OBJ FILES GET LINKED
      WITH LNKX11 (ONLY).

FILE: DRLPB.CTL
      THIS FILE EXPLAINS SEQUENCE OF ASSEMBLES AND LINKS.
      IT IS IN TOPS-20 FORMAT.

****
VERSION "B" WAS CREATED BECAUSE OF A MICRO-CODE CHANGE
TO THE LPA-11 (DMA-11 V5).

```
LNKX11   V023   24-OCT-80   9:33

#CRLPBB.BIN/B:42000,CRLPBB.MAP=CRLPBB,CRLPX2/E


LOAD MAP

IDENT:  4.01

TRANSFER ADDRESS: 000001
LOW LIMIT: 042000
HIGH LIMIT: 046000
**********
MODULE  MAINDE
SECTION ENTRY   ADDRESS SIZE
<. ABS.>        000000  000000
        DRLPX2  042000
<       >       042000  000000
**********
MODULE  DRLPX2
SECTION ENTRY   ADDRESS SIZE
<       >       042000  000000
<ABCODE>        042000  004000


 RUN-TIME:  0 SECONDS
 2K CORE USED
```

```
                        .REM    [

                                        CRLPAB.MAC

                WELCOME, THIS DIAGNOSTIC IS ONE IN A SERIES OF DIAGNOSTIC
                DESIGNED IN ORDER TO AID YOU IN TESTING THE LPA-11XX OPTION.
                I HOPE THAT YOU HAVE READ THE DOCUMENTATION SECTION OF THIS
                DIAGNOSTIC.  IF YOU HAVE,YOU KNOW ABOUT ALL OF THE DIAGNOSTICS
                THAT ARE AVAILIBLE FOR TESTING THE LPA SYSTEM.

                                        GOOD LUCK !

                [
                .GLOBL  DRLPX2
```

```
                        .TITLE  MMAST.MAC
                        .IDENT  /4.01/
                ;
                ; LPA11-K MICRO CODE
                ;
                ; CHARLES A. SAMUELSON
                ; NOVEMBER, 1977
                ;

                        .TITLE  DMAST.MAC
                        .IDENT  /4.01/
                ;
                ; LPA11-K MICRO CODE
                ;
                ; CHARLES A. SAMUELSON
                ; NOVEMBER, 1977
                ;
```

```
                                        .REM    !

                                        THIS IS A LIST OF TESTS DELETED FROM THIS DIAGNOSTIC.
                                        THESE TESTS COULD NOT BE DONE THROUGH THE LPA-11

                                        TEST THAT THE LOW BYTE OF STATUS REG CAN BE CLEARED
                                        TEST THAT THE HIGH BYTE OF STATUS REG CAN BE CLEARED
                                        TEST THAT DISPLAY DOES INTR. A1 LEVEL INDICATED
                                        TEST THAT DISPLAY DOES NOT INTR. AT LEVEL INDICATED
                                        TEST THE SCOPE SETTLING TIME

                                        !
                                .TITLE  MAINDEC-11-CRLPB-B      AA11-K  DIAGNOSTIC
                                ;*COPYRIGHT (C) 1980
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY EDWARD C. BADGER
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

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

                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                          R0=     %0              ;;GENERAL REGISTER
000001                          R1=     %1              ;;GENERAL REGISTER
000002                          R2=     %2              ;;GENERAL REGISTER
000003                          R3=     %3              ;;GENERAL REGISTER
000004                          R4=     %4              ;;GENERAL REGISTER
000005                          R5=     %5              ;;GENERAL REGISTER
000006                          R6=     %6              ;;GENERAL REGISTER
000007                          R7=     %7              ;;GENERAL REGISTER
000006                          SP=     %6              ;;STACK POINTER
000007                          PC=     %7              ;;PROGRAM COUNTER

                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
```

```
(1)          000100                  PR2=    100             ;;PRIORITY LEVEL 2
(1)          000140                  PR3=    140             ;;PRIORITY LEVEL 3
(1)          000200                  PR4=    200             ;;PRIORITY LEVEL 4
(1)          000240                  PR5=    240             ;;PRIORITY LEVEL 5
(1)          000300                  PR6=    300             ;;PRIORITY LEVEL 6
(1)          000340                  PR7=    340             ;;PRIORITY LEVEL 7
(1)
(1)                                  ;*"SWITCH REGISTER" SWITCH DEFINITIONS
(1)          100000                  SW15=   100000
(1)          040000                  SW14=   40000
(1)          020000                  SW13=   20000
(1)          010000                  SW12=   10000
(1)          004000                  SW11=   4000
(1)          002000                  SW10=   2000
(1)          001000                  SW09=   1000
(1)          000400                  SW08=   400
(1)          000200                  SW07=   200
(1)          000100                  SW06=   100
(1)          000040                  SW05=   40
(1)          000020                  SW04=   20
(1)          000010                  SW03=   10
(1)          000004                  SW02=   4
(1)          000002                  SW01=   2
(1)          000001                  SW00=   1
(1)                                  .EQUIV  SW09,SW9
(1)                                  .EQUIV  SW08,SW8
(1)                                  .EQUIV  SW07,SW7
(1)                                  .EQUIV  SW06,SW6
(1)                                  .EQUIV  SW05,SW5
(1)                                  .EQUIV  SW04,SW4
(1)                                  .EQUIV  SW03,SW3
(1)                                  .EQUIV  SW02,SW2
(1)                                  .EQUIV  SW01,SW1
(1)                                  .EQUIV  SW00,SW0
(1)
(1)                                  ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)          100000                  BIT15=  100000
(1)          040000                  BIT14=  40000
(1)          020000                  BIT13=  20000
(1)          010000                  BIT12=  10000
(1)          004000                  BIT11=  4000
(1)          002000                  BIT10=  2000
(1)          001000                  BIT09=  1000
(1)          000400                  BIT08=  400
(1)          000200                  BIT07=  200
(1)          000100                  BIT06=  100
(1)          000040                  BIT05=  40
(1)          000020                  BIT04=  20
(1)          000010                  BIT03=  10
(1)          000004                  BIT02=  4
(1)          000002                  BIT01=  2
(1)          000001                  BIT00=  1
(1)                                  .EQUIV  BIT09,BIT9
(1)                                  .EQUIV  BIT08,BIT8
(1)                                  .EQUIV  BIT07,BIT7
(1)                                  .EQUIV  BIT06,BIT6
```

```
(1)                             .EQUIV  BIT05,BIT5
(1)                             .EQUIV  BIT04,BIT4
(1)                             .EQUIV  BIT03,BIT3
(1)                             .EQUIV  BIT02,BIT2
(1)                             .EQUIV  BIT01,BIT1
(1)                             .EQUIV  BIT00,BIT0
(1)
(1)                             ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)          000004             ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)          000010             RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)          000014             TBITVEC=14              ;;"T" BIT
(1)          000014             TRTVEC= 14              ;;TRACE TRAP
(1)          000014             BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)          000020             IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)          000024             PWRVEC= 24              ;;POWER FAIL
(1)          000030             EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)          000034             TRAPVEC=34              ;;"TRAP" TRAP
(1)          000060             TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)          000064             TPVEC=  64              ;;TTY PRINTER VECTOR
(1)          000240             PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
1178
1179         170416                     ABASE=170416
1180         100360                     AVECT1=100360
1181         000200                     APRIOR=200
1182
1183                            .SBTTL  OPERATIONAL SWITCH SETTINGS
(1)                             ;*
(1)                             ;*      SWITCH                  USE
(1)                             ;*      ------          ----------------
(1)                             ;*        15            HALT ON ERROR
(1)                             ;*        14            LOOP ON TEST
(1)                             ;*        13            INHIBIT ERROR TYPEOUTS
(1)                             ;*        12            STORAGE SCOPE CONNECTED
(1)                             ;*        10            EXTERNAL DELAY SIGNAL CONNECTED
(1)                             ;*         8            LOOP ON TEST IN SWR<7:0>
1184                            ;*         9            TWO'S COMPLEMENT MODE
1185                            .SBTTL  TRAP CATCHER
(1)
(1)          000000                     .=0
(1)                             ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)                             ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)                             ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)          000174                     .=174
(1)  000174  000000             DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
(1)  000176  000000             SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
(1)                             .SBTTL  STARTING ADDRESS(ES)
(1)  000200  000137  001502             JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
1186 000204  000137  001520             JMP     BEGIN1                  ;JUMP TO RESTART ADDRESS
1187 000210  000240                     NOP
1188 000212  000240                     NOP
1189 000214  000137  022022             JMP     $UTK                    ;JUMP TO USER LINK
```

```
1191                                    .SBTTL  ACT11 HOOKS
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;HOOKS REQUIRED BY ACT11
 (1)            000220                          $SVPC=.                 ;SAVE PC
 (1)            000046                          .=46
 (1)  000046    007060                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
 (1)            000052                          .=52
 (1)  000052    000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
 (1)            000220                          .=$SVPC                 ;; RESTORE PC
1192            001000                          .=1000
1193                                    .SBTTL  APT PARAMETER BLOCK
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
 (2)                                    ;;*****************************************************************
 (1)            001000                          .$X=.   ;;SAVE CURRENT LOCATION
 (1)            000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
 (1)  000024    000200                          200     ;;FOR APT START UP
 (1)            000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
 (1)  000044    001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
 (1)            001000                          .=.$X   ;;RESET LOCATION COUNTER
 (2)                                    ;;*****************************************************************
 (1)                                    ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
 (1)                                    ;INTERFACE SPEC.
 (1)
 (1)  001000                            $APTHD:
 (1)  001000    000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
 (1)  001002    001172                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
 (1)  001004    000030                  $TSTM:  .WORD   30      ;;RUN TIM OF LONGEST TEST
 (1)  001006    000060                  $PASTM: .WORD   60      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
 (1)  001010    000120                  $UNITM: .WORD   120     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
 (1)  001012    000031                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
1194                                    .SBTTL  COMMON TAGS
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
 (1)                                    ;*USED IN THE PROGRAM.
 (1)
 (1)            001100                          .=1100
 (1)  001100                            $CMTAG:                         ;;START OF COMMON TAGS
 (1)  001100  000000                            .WORD   0
 (1)  001102     000                    $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
 (1)  001103     000                    $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
 (1)  001104  000000                    $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
 (1)  001106  000000                    $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
 (1)  001110  000000                    $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
 (1)  001112  000000                    $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
 (1)  001114     000                    $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
 (1)  001115     001                    $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
 (1)  001116  000000                    $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
 (1)  001120  000000                    $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
 (1)  001122  000000                    $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
 (1)  001124  000000                    $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
 (1)  001126  000000                    $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
 (1)  001130  000000                            .WORD   0               ;;RESERVED--NOT TO BE USED
 (1)  001132  000000                            .WORD   0
 (1)  001134     000                    $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
 (1)  001135     000                    $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
 (1)  001136  000000                            .WORD   0
 (1)  001140  177570                    SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
 (1)  001142  177570                    DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
 (1)  001144  177560                    $TKS:   177560                  ;;TTY KBD STATUS
 (1)  001146  177562                    $TKB:   177562                  ;;TTY KBD BUFFER
 (1)  001150  177564                    $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
 (1)  001152  177566                    $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
 (1)  001154     000                    $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
 (1)  001155     002                    $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
 (1)  001156     012                    $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
 (1)  001157     000                    $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
 (1)  001160  000000                    $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
 (1)                                                                    ;;WHICH  ($REG0) WAS OBTAINED
 (3)  001162  000000                    $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
 (3)  001164  000000                    $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
 (1)  001166     077                    $QUES:  .ASCII  /?/             ;;QUESTION MARK
 (1)  001167     015                    $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
 (1)  001170  000012                    $LF:    .ASCII  <12>            ;;LINE FEED
 (2)                                    ;;**********************************************************************
 (2)                                    .SBTTL  APT MAILBOX-ETABLE
 (2)
 (3)                                    ;;**********************************************************************
 (2)                                    .EVEN
 (2)  001172                            $MAIL:                          ;;APT MAILBOX
 (2)  001172  000000                    $MSGTY: .WORD   AMSGTY           ;;MESSAGE TYPE CODE
 (2)  001174  000000                    $FATAL: .WORD   AFATAL           ;;FATAL ERROR NUMBER
 (2)  001176  000000                    $TESTN: .WORD   ATESTN           ;;TEST NUMBER
 (2)  001200  000000                    $PASS:  .WORD   APASS            ;;PASS COUNT
 (2)  001202  000000                    $DEVCT: .WORD   ADEVCT           ;;DEVICE COUNT
 (2)  001204  000000                    $UNIT:  .WORD   AUNIT            ;;I/O UNIT NUMBER
```

```
(2)  001206  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(2)  001210  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001212                        $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001212     000                $ENV:   .BYTE   AENV    .;ENVIRONMENT BYTE
(2)  001213     000                $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001214  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001216  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001220  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                ;*                      BITS 15-11=CPU TYPE
(2)                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                                ;*                              11/70=06,PDQ=07,Q=10
(2)                                ;*                      BIT 10=REAL TIME CLOCK
(2)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001222     000                $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001223     000                $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                ;*                              900 NSEC CORE=001
(2)                                ;*                              300 NSEC BIPOLAR=002
(2)                                ;*                              500 NSEC MOS=003
(2)  001224  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2)  001226     000                $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001227     000                $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001230  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001232     000                $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S BYTE
(2)  001233     000                $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001234  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001236     000                $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001237     000                $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001240  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001242  100360                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001244  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001246  170416                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001250  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001252  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001254                        $ETEND:
(2)                                .MEXIT
```

```
    (1)                                 .SBTTL  ERROR POINTER TABLE
    (1)
    (1)                                 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
    (1)                                 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
    (1)                                 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
    (1)                                 ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
    (1)                                 ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
    (1)
    (1)                                 ;*      EM              ;;POINTS TO THE ERROR MESSAGE
    (1)                                 ;*      DH              ;;POINTS TO THE DATA HEADER
    (1)                                 ;*      DT              ;;POINTS TO THE DATA
    (1)                                 ;*      DF              ;;POINTS TO THE DATA FORMAT
    (1)
    (1)
    (1)  001254                         $ERRTB:
 1195
 1197
 1198                                   ;ITEM   1
 1199  001254  013450                           EM1                     ;AA11-K STATUS REGISTER IN ERROR
 1200  001256  013756                           DH1                     ;ERRPC  IBASE   GOOD    BAD
 1201  001260  014464                           DT1                     ;$ERRPC STAT    $GDDAT  $BDDAT
 1202  001262  014600                           DF0
 1203
 1204                                   ;ITEM   2
 1205  001264  013501                           EM2                     ; DAC0 REGISTER IN ERROR
 1206  001266  013756                           DH1                     ;ERRPC  IBASE   GOOD    BAD
 1207  001270  014476                           DT2                     ;$ERRPC DAC0    $GDDAT  $BDDA1
 1208  001272  014600                           DF0
 1209
 1210                                   ;ITEM   3
 1211  001274  013530                           EM3                     ; DAC1 REGISTER IN ERROR
 1212  001276  013756                           DH1                     ;ERRPC  IBASE   GOOD    BAD
 1213  001300  014510                           DT3                     ;$ERRPC DAC1    $GDDAT  $BDDAT
 1214  001302  014600                           DF0
 1215
 1216                                   ;ITEM   4
 1217  001304  013557                           EM4                     ;DAC #2 REGISTER IN ERROR
 1218  001306  013756                           DH1                     ;ERRPC  IBASE   GOOD    BAD
 1219  001310  014522                           DT4                     ;$ERRPC DAC2    $GDDAT  $BDDAT
 1220  001312  014600                           DF0
 1221
 1222                                   ;ITEM   5
 1223  001314  013606                           EM5                     ;DAC #3 REGISTER IN ERROR
 1224  001316  013756                           DH1                     ;ERRPC  IBASE   GOOD    BAD
 1225  001320  014534                           DT5                     ;$ERRPC DAC3    $GDDAT  $BDDAT
 1226  001322  014600                           DF0
 1227
 1228                                   ;ITEM   6
 1229  001324  013735                           EM13                    ;PREVIOUSLY EXISTING AA11-K DOES NOT RESPOND NOW
 1230  001326  014026                           DH13
 1231  001330  014554                           DT13                    ;$FRRPC $BASE   EVER    $UNIT
 1232  001332  014600                           DF0
 1233
 1234                                   ;ITEM   7
 1235  001334  000000                           0
 1236  001336  000000                           0
```

```
1237  001340  000000                        0
1238  001342  000000                        0
1239
1240                                ;ITEM   10
1241  001344  013635                        EM10                    ;BUS TIME-OUT ON AA11-K ADDRESSES
1242  001346  014012                        DH6                     ;ERRPC  IBASE
1243  001350  014546                        DT6                     ;$ERRPC STAT
1244  001352  014600                        DF0
1245
1246                                ;ITEM   11
1247  001354  013706                        EM11                    ;AA11-K READY BIT ERROR
1248  001356  013756                        DH1                     ;ERRPC  IBASE   GOOD    BAD
1249  001360  014464                        DT1                     ;$ERRPC STAT    $GDDAT  $BDDAT
1250  001362  014600                        DF0
1251
1252                                        ;
 (1)                                    ;ADDRESS OF KMC-11 OF LPA-11    THE ADDR FOR KMADO MAY BE
 (1)                                    ;                               CHANGED BY THE USER TO REFLECT
 (1)                                    ;                               A DIFFERENT KMC-11 ADDR. THE
 (1)                                    ;                               REST OF THE ADDRESSES WILL
 (1)                                    ;                               BE CHANGED BY THE PROGRAM.
 (1)                                    ;
 (1)
 (1)  001364                    LPCI:
 (1)  001364  170460            KMADO:  .WORD   170460          ;BASE KMC ADDR. MAY BE PATCHED BY USER.
 (1)
 (1)  001366                    LPMR:
 (1)  001366  170461            KMAD1:  .WORD   170460+1                ;>DO NOT        <;KMC-CSR ADDR
 (1)  001370                    LPCO:
 (1)  001370  170462            KMAD2:  .WORD   170460+2                ;>PATCH         <;
 (1)  001372                    LPSO:
 (1)  001372  170463            KMAD3:  .WORD   170460+3                ;>THIS AREA     <
 (1)  001374                    LPADL:
 (1)  001374  170464            KMAD4:  .WORD   170460+4                ;
 (1)  001376                    LPADH:
 (1)  001376  170465            KMAD5:  .WORD   170460+5                ;>DO NOT        <
 (1)  001400                    LPMS1:
 (1)  001400  170466            KMAD6:  .WORD   170460+6                ;>PATCH         <
 (1)  001402                    LPMS2:
 (1)  001402  170467            KMAD7:  .WORD   170460+7                ;>THIS AREA     <
 (1)
 (1)  001404  000360            VECTOR: .WORD   AVECT1&777      ;BASE VECTOR OF KMC
 (1)  001406  000364            VECTPS: .WORD   4+AVECT1&777    ;VECOTR ADDR.+2
 (1)
 (1)  001410  000005            VERSN:  .WORD   5               ;CURRENT VERSION NUMBER OF MICROCODE.
 (1)
 (1)  001412  000000            .DVLS:  .WORD   0               ;/DEVICE LIST OF I/O ADDR. DEFINED
 (1)  001414  000020                    .BLKW   16.             ;/BY INIT.
 (1)
1253  001454  000040            VADDR:  40              ;ADJUSTMENT TO NEXT AA11-K'S ADDRESS
1254  001456  000040            VVECT:  40              ;ADJUSTMENT TO NEXT AA11-K'S VECTOR
1255  001460  000012            DELAY:  10.             ;DELAY COUNTER FOR DELAY LOOPS
1256  001462  000000            $TMDAT: 0
1257  001464  170416            STAT:   170416
1258  001466  170420            DAC0:   170420
1259  001470  170422            DAC1:   170422
```

```
  1260  001472  170424                    DAC2:   170424
  1261  001474  170426                    DAC3:   170426
  1262  001476  000350                    IV:     350
  1263  001500  000352                    IVS:    352
  1264
  1265  001502  005037  014632            BEGIN:  CLR     TEMP
  1266  001506  005037  014642                    CLR     EVER            ;RESET POINTER
  1267  001512  005037  014644                    CLR     EVER1
  1268  001516  000403                            BR      BEG
  1269  001520  012737  000001  014632    BEGIN1: MOV     #1,TEMP
  1270  001526  000240                    BEG:    NOP
  1272                                    .SBTTL  INITIALIZE THE COMMON TAGS
   (1)                                    ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
   (1)  001530  012706  001100                    MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
   (1)  001534  005026                            CLR     (R6)+           ;;CLEAR MEMORY LOCATION
   (1)  001536  022706  001140                    CMP     #SWR,R6 ;;DONE?
   (1)  001542  001374                            BNE     .-6             ;;LOOP BACK IF NO
   (1)  001544  012706  001100                    MOV     #STACK,SP       ;;SETUP THE STACK POINTER
   (1)                                    ;;INITIALIZE A FEW VECTORS
   (1)  001550  012737  015100  000020            MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
   (1)  001556  012737  000340  000022            MOV     #340,@#IOTVEC+2 ;;LEVEL 7
   (1)  001564  012737  015232  000030            MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
   (1)  001572  012737  000340  000032            MOV     #340,@#EMTVEC+2 ;;LEVEL 7
   (1)  001600  012737  017572  000034            MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
   (1)  001606  012737  000340  000036            MOV     #340,@#TRAPVEC+2;LEVEL 7
   (1)  001614  012737  015516  000024            MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
   (1)  001622  012737  000340  000026            MOV     #340,@#PWRVEC+2 ;;LEVEL 7
   (1)  001630  012737  001630  001106            MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
   (2)                                    ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   (2)                                    ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
   (2)  001636  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
   (2)  001642  012737  001676  000004            MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
   (2)  001650  012737  177570  001140            MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
   (2)  001656  012737  177570  001142            MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
   (2)  001664  022777  177777  177246            CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
   (2)  001672  001012                            BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
   (2)                                                                    ;;AND  THE HARDWARE SWR IS NOT = -1
   (2)  001674  000403                            BR      65$             ;;BRANCH IF NO TIMEOUT
   (2)  001676  012716  001704            64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
   (2)  001702  000002                            RTI
   (2)  001704  012737  000176  001140    65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
   (2)  001712  012737  000174  001142            MOV     #DISPREG,DISPLAY
   (2)  001720  012637  000004            66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
   (1)
   (2)  001724  005037  001200                    CLR     $PASS           ;;CLEAR PASS COUNT
   (2)  001730  132737  000200  001213            BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
   (2)  001736  001403                            BEQ     67$             ;;YES,USE NON-APT SWITCH
   (2)  001740  012737  001214  001140            MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
   (2)  001746                            67$:
  1273                                            ;
   (1)                                    ;THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
   (1)                                    ;
   (1)
   (1)  001746  010046                            MOV     R0,-(SP)
   (1)  001750  010146                            MOV     R1,-(SP)
   (1)  001752  013700  001364                    MOV     KMADO,R0        ;GET KMC-11 ADDRESS.
```

```
(1)  001756  012701  001366                    MOV     #KMAD1,R1       ;   ADDR. OF ADDR. LIST.
(1)
(1)  001762  005200                      68$:   INC     R0              ;UPDATE ADDR.
(1)  001764  010021                             MOV     R0,(1)+         ;WRITE ADDR.
(1)  001766  020127  001404                     CMP     R1,#KMAD7+2     ;DONE ALL ADDRESSES?
(1)  001772  001373                             BNE     68$             ;NO - DO NEXT ADDR.
(1)  001774  005037  001412                     CLR     .DVLS           ;CLR ADDR. LIST.
(1)  002000  012601                             MOV     (SP)+,R1
(1)  002002  012600                             MOV     (SP)+,R0
1274
1275  002004  005037  177776                    CLR     @#PS
1276  002010  000137  002176                    JMP     INIT1
1277                                    .SBTTL  SUBROUTINE TO LOAD A TRAP CATCHER
1278  002014  012702  000242           LDTRAP:  MOV     #242,R2                 ;LOAD R2
1279  002020  012701  000240                    MOV     #240,R1                 ;LOAD R1
1280  002024  010221                    5$:     MOV     R2,(R1)+                ;LOAD .+2
1281  002026  005021                            CLR     (R1)+                   ;LOAD HALT
1282  002030  010102                            MOV     R1,R2                   ;LOAD R2
1283  002032  005722                            TST     (R2)+                   ;BUMP R2
1284  002034  020227  001002                    CMP     R2,#1002                ;TEST FOR LAST
1285  002040  001371                            BNE     5$                      ;BR UNTIL DONE
1286                                    .SBTTL  LOAD DEVICE ADDRESSES LOCATIONS
1287  002042  012700  001464                    MOV     #STAT,R0                ;LOAD POINTER
1288  002046  013720  001246           10$:    MOV     $BASE,(R0)+             ;LOAD BASE ADDRESS
1289  002052  022700  001476                    CMP     #IV,R0                  ;TEST FOR DONE
1290  002056  001373                            BNE     10$                     ;BR
1291  002060  012700  001466                    MOV     #DACO,R0                ;LOAD 2ND ADDRESS
1292  002064  012701  000002                    MOV     #2,R1                   ;LOAD R1
1293  002070  060120                    12$:    ADD     R1,(R0)+                ;UPDATE REAL DEVICE WORD
1294  002072  005721                            TST     (R1)+                   ;BUMP R1
1295  002074  022701  000012                    CMP     #12,R1                  ;TEST FOR DONE
1296  002100  001373                            BNE     12$                     ;BR
1297  002102  013737  001242  001476            MOV     $VECT1,IV               ;LOAD INTR. VECTOR ADDRESS
1298  002110  042737  160000  001476            BIC     #160000,IV
1299  002116  013737  001476  001500            MOV     IV,IVS
1300  002124  062737  000002  001500            ADD     #2,IVS
1301  002132  113737  001243  014636            MOVB    $VECT1+1,BRLEV1         ;LOAD BR LEVEL
1302  002140  042737  177437  014636            BIC     #177437,BRLEV1
1303  002146  013737  014636  014640            MOV     BRLEV1,BRLEV2
1304  002154  162737  000040  014640            SUB     #40,BRLEV2
1305  002162  000207                            RTS     PC                      ;EXIT
1306
1307                                    .SBTTL  INITIAL HEADER TYPEOUT AND WAIT FOR OPERATOR
1308  002164  046101                    AL:     .ASCII  /AL/
1309  002166  042101                    AD:     .ASCII  /AD/
1310  002170  046115                    ML:     .ASCII  /ML/
1311  002172  042115                    MD:     .ASCII  /MD/
1312  002174  041515                    MC:     .ASCII  /MC/
1313
1314  002176  004737  002014           INIT1:  JSR     PC,LDTRAP
1315  002202  005737  014632                    TST     TEMP            ;TEST IF START OR RESTART
1316  002206  001057                            BNE     CTRLC           ;RESTART
1317  002210  005737  000042                    TST     @#42            ;TEST IF MONITOR
1318  002214  001061                            BNE     MTEST           ;BR IF NOT
1319  002216  104401                            TYPE                    ;CALL MESSAGE PRINTER VIA 'EMT'
1320  002220  012402                            TITLE                   ;TYPE PROGRAM HEADER.
```

```
1321  002222  012706  001100          WAITIN: MOV     #STACK,SP       ;RESET STACK
1322  002226  104407                          CKSWR                   ;TEST FOR CTRL G
1323  002230  104411                          RDLIN                   ;READ TWO CHARACTERS
1324  002232  013637  014646                  MOV     @(SP)+,OPRIN    ;READ THE CHARACTER
1325  002236  042737  000000  014646          BIC     #0,OPRIN        ;MASK OTHER CHARACTERS
1326
1327  002244  023737  002164  014646          CMP     AL,OPRIN        ;TEST FOR "AL"
1328  002252  001423                          BEQ     10$             ;BR IF YES
1329  002254  023737  002166  014646          CMP     AD,OPRIN        ;TEST FOR "AD"
1330  002262  001421                          BEQ     11$             ;BR IF YES
1331  002264  023737  002170  014646          CMP     ML,OPRIN        ;TEST FOR "ML"
1332  002272  001417                          BEQ     13$             ;BR FI YES
1333  002274  023737  002172  014646          CMP     MD,OPRIN        ;TEST FOR "MD"
1334  002302  001415                          BEQ     14$             ;BR IF YES
1335  002304  023737  002174  014646          CMP     MC,OPRIN        ;TEST FOR "MC"
1336  002312  001413                          BEQ     15$             ;BR IF YES
1337  002314  104401                          TYPE
1338  002316  014105                          QMARK
1339  002320  000740                          BR      WAITIN          ;WAIT FOR OPERATOR AGAIN
1340
1341  002322  000137  002360          10$:    JMP     MTEST           ;RUN THE LOGIC TEST
1342  002326  000137  007114          11$:    JMP     VISUAL          ;RUN THE VISUAL PATTERN
1343  002332  000137  011216          13$:    JMP     MANUL           ;RUN THE MANUAL LOGIC TEST
1344  002336  000137  011024          14$:    JMP     FULRMP          ;RUN THE RAMP PATTERN ON FOUR DAC'S
1345  002342  000137  011532          15$:    JMP     CALDAC          ;RUN THE MANUAL CAL OF THE DAC
1346
1347  002346  104401                  CTRLC:  TYPE
1348  002350  014113                          CONTC
1349  002352  005037  001200                  CLR     $PASS
1350  002356  000721                          BR      WAITIN
1351
1352                                  .SBTTL  DETERMINE THE NUMBER OF AA11-K ON THIS SYSTEM
1353
1354  002360  013737  001246  001126  MTEST:  MOV     $BASE,$BDDAT    ;GET THE BASE ADDRESS
1355  002366  005037  001412                  CLR     .DVLS
1356  002372  005037  001204                  CLR     $UNIT           ;CLEAR UNIT #
1357  002376                          1$:
 (1)
 (1)                                  ;*      MOV     @$BDDAT,$TMDAT  ;/READ DEVICE REG $BDDAT,PUT DATA IN $TMDAT.
1358  002406  005737  020726                  TST     $AERR
1359  002412  001006                          BNE     2$
1360  002414  063737  001454  001126          ADD     VADDR,$BDDAT    ;UPDATE THE BUS ADDRESS
1361  002422  005237  001204                  INC     $UNIT           ;UPDATE UNIT COUNT
1362  002426  000763                          BR      1$
1363  002430  000240                  2$:     NOP
1364  002432  005737  001204                  TST     $UNIT           ;TEST IF ANY EXIST
1365  002436  001002                          BNE     3$              ;BR IF SOME ARE THERE
1366  002440  104010                          ERROR   10              ;BASE ADDRESS CAUSED AN BUS TRAP
1367  002442  000440                          BR      TST1            ;;
1368  002444  005737  014642          3$:     TST     EVER            ;TEST IF # HAS BEEN REPORTED
1369  002450  100423                          BMI     4$              ;BR IF IT HAS
1370  002452  122737  000001  001134          CMPB    #1,$AUTOB       ;TEST IF AUTO MODE (XXDP CHAIN)
1371  002460  001410                          BEQ     6$              ;BR IF YES
1372  002462  104401                          TYPE
1373  002464  014121                          FOUND1                  ;TELL OPERATOR THE # OF AA11-K'S
1374  002466  013746  001204                  MOV     $UNIT,-(SP)
```

```
                                        TYPOS
                                        .BYTE   2
                                        .BYTE   0
                                        TYPE
                                        FOUND2
002502  013737  001204  014642  6$:   MOV     $UNIT,EVER      ;SAVE THE # OF AA11-K'S FOR LATER
002510  052737  100000  014642        BIS     #BIT15,EVER     ;SET "REPORTED # FLAG"
002516  000405                        BR      5$              ;;
002520  123737  014642  001204  4$:   CMPB    EVER,$UNIT      ;TEST IF ANY HAVE GONE AWAY
002526  001401                        BEQ     5$              ;;BR IF ALL ARE STILL HERE
002530  104006                        ERROR   6               ;EXISTING UNIT FAILED TO RESPOND NOW
002532  005037  001204          5$:   CLR     $UNIT           ;RESET UNIT POINTER
002536  004737  002014                JSR     PC,LDTRAP       ;LOAD TRAP CATCHER AND BUS ADDRESSES

002542  000240                LTES :  NOP
                                ;;*****************************************************************
                                ;*TEST 1         TEST THAT THE AA11-K RESPONDS TO THE CPU
                                ;;*****************************************************************
                                TST1:   SCOPE
002546  012737  002562  001106        MOV     #3$,$LPADR
002554  012737  002660  000004        MOV     #1$,ERRVEC      ;LOAD BUS TRAP RETURN
002562                          3$:

                                ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
002572  005737  001462                TST     $TMDAT

                                ;*      MOV     @DAC0,$TMDAT    ;/READ DEVICE REG DAC0,PUT DATA IN $TMDAT.
002606  005737  001462                TST     $TMDAT

                                ;*      MOV     @DAC1,$TMDAT    ;/READ DEVICE REG DAC1,PUT DATA IN $TMDAT.
002622  005737  001462                TST     $TMDAT

                                ;*      MOV     @DAC2,$TMDAT    ;/READ DEVICE REG DAC2,PUT DATA IN $TMDAT.
002636  005737  001462                TST     $TMDAT

                                ;*      MOV     @DAC3,$TMDAT    ;/READ DEVICE REG DAC3,PUT DATA IN $TMDAT.
002652  005737  001462                TST     $TMDAT
002656  000402                        BR      2$              ;;BR AND RESTORE LOC. 4
002660  022626                1$:     CMP     (SP)+,(SP)+     ;CLEAN THE STACK
002662  104010                        ERROR   10              ;ERROR, BUS TIMEOUT WHEN ADDRESSING THE AA11-K
002664  012737  000006  000004  2$:   MOV     #6,ERRVEC       ;LOAD RETURN
                                ;;*****************************************************************
                                ;*TEST 2         TEST THAT THE DAC0 REGISTER CAN BE CLEARED
                                ;;*****************************************************************
                                TST2:   SCOPE
002674  005037  001124                CLR     $GDDAT                  ;LOAD EXPECTED

                                ;*      MOV     $GDDAT,@DAC0    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0

                                ;*      MOV     @DAC0,$BDDAT    ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
002720  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
002726  001401                        BEQ     TST3            ;;BR IF EQUAL
002730  104002                        ERROR   2                       ;ERROR, DAC0 REGISTER NOT = 0

                                ;;*****************************************************************
                                ;*TEST 3         TEST THAT THE DAC0 REGISTER CAN BE LOADED WITH #7777
```

```
 (3)                                    ;;*****************************************************************
 (2)  002732  000004                    TST3:   SCOPE
1433  002734  012737  007777  001124             MOV     #7777,$GDDAT            ;LOAD EXPECTED
1434
 (1)                                    ;*      MOV     $GDDAT,@DAC0    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0
1435
 (1)                                    ;*      MOV     @DAC0,$BDDAT    ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
1436  002762  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1437  002770  001401                           BEQ     TST4            ;;BR IF EQUAL
1438  002772  104002                           ERROR   2                       ;ERRCR,  DAC0 REGISTER NOT = 7777
1439
1440                                    ;;*****************************************************************
 (3)                                    ;*TEST 4         TEST THAT THE DAC0 REGISTER CAN HOLD A FLOATING 1 PATTERN
 (3)                                    ;;*****************************************************************
 (2)  002774  000004                    TST4:   SCOPE
1441  002776  012737  004000  001124             MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
1442  003004                           1$:
 (1)
 (1)                                    ;*      MOV     $GDDAT,@DAC0    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0
1443
 (1)                                    ;*      MOV     @DAC0,$BDDAT    ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
1444  003024  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE THE DATA
1445  003032  001401                           BEQ     2$              ;;BR IF SAME
1446  003034  104002                           ERROR   2                       ;ERROR, DAC0 REGISTER FAILED TO HOLD A FLOATING
1447  003036  006237  001124           2$:     ASR     $GDDAT                  ;CHANGE THE DATA
1448  003042  001360                           BNE     1$                      ;BR AND TEST MORE DATA
1449                                    ;;*****************************************************************
 (3)                                    ;*TEST 5         TEST THAT THE DAC #1 REGISTER CAN BE CLEARED
 (3)                                    ;;*****************************************************************
 (2)  003044  000004                    TST5:   SCOPE
1450  003046  005037  001124                   CLR     $GDDAT                  ;LOAD EXPECTED
1451
 (1)                                    ;*      MOV     $GDDAT,@DAC1    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC1
1452
 (1)                                    ;*      MOV     @DAC1,$BDDAT    ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
1453  003072  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1454  003100  001401                           BEQ     TST6            ;;BR IF EQUAL
1455  003102  104003                           ERROR   3                       ;ERROR,  DAC1 REGISTER NOT = 0
1456
1457                                    ;;*****************************************************************
 (3)                                    ;*TEST 6         TEST THAT THE Y REGISTER CAN BE LOADED WITH #7777
 (3)                                    ;;*****************************************************************
 (2)  003104  000004                    TST6:   SCOPE
1458  003106  012737  007777  001124             MOV     #7777,$GDDAT            ;LOAD EXPECTED
1459
 (1)                                    ;*      MOV     $GDDAT,@DAC1    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC1
1460
 (1)                                    ;*      MOV     @DAC1,$BDDAT    ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
1461  003134  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1462  003142  001401                           BEQ     TST7            ;;BR IF EQUAL
1463  003144  104003                           ERROR   3                       ;ERROR,  DAC1 REGISTER NOT = 7777
1464
1465                                    ;;*****************************************************************
 (3)                                    ;*TEST 7         TEST THAT THE Y REGISTER CAN HOLD A FLOATING 1 PATTERN
 (3)                                    ;;*****************************************************************
 (2)  003146  000004                    TST7:   SCOPE
```

```
1466  003150  012737  004000  001124          MOV    #BIT11,$GDDAT          ;LOAD EXPECTED
1467  003156                          1$:
 (1)
 (1)                                ;*      MOV    $GDDAT,@DAC1    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC1
1468
 (1)                                ;*      MOV    @DAC1,$BDDAT    ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
1469  003176  023737  001124  001126          CMP    $GDDAT,$BDDAT          ;COMPARE THE DATA
1470  003204  001401                         BEQ    2$             ;;BR IF DATA IS SAME
1471  003206  104003                         ERROR  3              ;ERROR, DAC1 REGISTER FAILED TO HOLD A FLOATING
1472  003210  006237  001124          2$:   ASR    $GDDAT                 ;CHANGE THE DATA
1473  003214  001360                         BNE    1$                     ;BR AND TEST MORE DATA
1474
1475                                ;;*****************************************************************
 (3)                               ;*TEST 10       TEST THAT THE DAC #2 REGISTER CAN BE CLEARED
 (3)                               ;;*****************************************************************
 (2)  003216  000004               TST10:  SCOPE
1476  003220  005037  001124                 CLR    $GDDAT                 ;LOAD EXPECTED
1477
 (1)                                ;*      MOV    $GDDAT,@DAC2    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC2
1478
 (1)                                ;*      MOV    @DAC2,$BDDAT    ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
1479  003244  023737  001124  001126          CMP    $GDDAT,$BDDAT          ;COMPARE
1480  003252  001401                         BEQ    TST11          ;;BR IF EQUAL
1481  003254  104004                         ERROR  4              ;ERROR, DAC #2 REGISTER NOT = 0
1482
1483                                ;;*****************************************************************
 (3)                               ;*TEST 11       TEST THAT THE DAC #2 REGISTER CAN BE LOADED WITH #7777
 (3)                               ;;*****************************************************************
 (2)  003256  000004               TST11:  SCOPE
1484  003260  012737  007777  001124          MOV    #7777,$GDDAT           ;LOAD EXPECTED
1485
 (1)                                ;*      MOV    $GDDAT,@DAC2    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC2
1486
 (1)                                ;*      MOV    @DAC2,$BDDAT    ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
1487  003306  023737  001124  001126          CMP    $GDDAT,$BDDAT          ;COMPARE
1488  003314  001401                         BEQ    TST12          ;;BR IF EQUAL
1489  003316  104004                         ERROR  4              ;ERROR, DAC #2 REGISTER NOT = 7777
1490
1491                                ;;*****************************************************************
 (3)                               ;*TEST 12       TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN
 (3)                               ;;*****************************************************************
 (2)  003320  000004               TST12:  SCOPE
1492  003322  012737  004000  001124          MOV    #BIT11,$GDDAT          ;LOAD EXPECTED
1493  003330                          1$:
 (1)
 (1)                                ;*      MOV    $GDDAT,@DAC2    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC2
1494
 (1)                                ;*      MOV    @DAC2,$BDDAT    ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
1495  003350  023737  001124  001126          CMP    $GDDAT,$BDDAT          ;COMPARE THE DATA
1496  003356  001401                         BEQ    2$             ;;BR IF SAME
1497  003360  104004                         ERROR  4              ;ERROR, DAC #2 REGISTER FAILED TO HOLD A FLOATIN
1498  003362  006237  001124          2$:   ASR    $GDDAT                 ;CHANGE THE DATA
1499  003366  001360                         BNE    1$                     ;BR AND TEST MORE DATA
1500
1501
1502                                ;;*****************************************************************
```

```
   (3)                                  ;*TEST 13       TEST THAT THE DAC #3 REGISTER CAN BE CLEARED
   (3)                                  ;;**************************************************************
   (2)  003370  000004                  TST13:  SCOPE
  1503  003372  005037  001124                   CLR     $GDDAT                  ;LOAD EXPECTED
  1504
   (1)                                  ;*      MOV     $GDDAT,@DAC3    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC3
  1505
   (1)                                  ;*      MOV     @DAC3,$BDDAT    ;/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
  1506  003416  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
  1507  003424  001401                       BEQ     TST14           ;;BR IF EQUAL
  1508  003426  104005                       ERROR   5                       ;ERROR, DAC #3 REGISTER NOT = 0
  1509
  1510                                  ;;**************************************************************
   (3)                                  ;*TEST 14       TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777
   (3)                                  ;;************************************************************** ***
   (2)  003430  000004                  TST14:  SCOPE
  1511  003432  012737  007777  001124        MOV     #7777,$GDDAT            ;LOAD EXPECTED
  1512
   (1)                                  ;*      MOV     $GDDAT,@DAC3    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC3
  1513
   (1)                                  ;*      MOV     @DAC3,$BDDAT    ;/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
  1514  003460  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
  1515  003466  001401                       BEQ     TST15           ;;BR IF EQUAL
  1516  003470  104005                       ERROR   5                       ;ERROR, DAC #3 REGISTER NOT = 7777
  1517
  1518                                  ;;**************************************************************
   (3)                                  ;*TEST 15       TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN
   (3)                                  ;;**************************************************************
   (2)  003472  000004                  TST15:  SCOPE
  1519  003474  012737  004000  001124        MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
  1520  003502                        1$:
   (1)
   (1)                                  ;*      MOV     $GDDAT,@DAC3    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC3
  1521
   (1)                                  ;*      MOV     @DAC3,$BDDAT    ;/READ DEVILE REG DAC3,PUT DATA IN $BDDAT.
  1522  003522  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE THE DATA
  1523  003530  001401                       BEQ     2$              ;;BR IF SAME
  1524  003532  104005                       ERROR   5                       ;ERROR, DAC #3 REGISTER FAILED TO HOLD A FLOATIN
  1525  003534  006237  001124        2$:     ASR     $GDDAT                  ;CHANGE THE DATA
  1526  003540  001360                       BNE     1$                      ;BR AND TEST MORE DATA
  1527                                  ;;**************************************************************
   (3)                                  ;*TEST 16       TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA
   (3)                                  ;;**************************************************************
   (2)  003542  000004                  TST16:  SCOPE
  1528  003544  012737  001111  001462        MOV     #1111,$TMDAT            ;LOAD DAC #0
  1529
   (1)                                  ;*      MOV     $TMDAT,@DAC0    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC0
  1530  003562  012737  002222  001462        MOV     #2222,$TMDAT            ;LOAD DAC #1
  1531
   (1)                                  ;*      MOV     $TMDAT,@DAC1    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC1
  1532  003600  012737  004444  001462        MOV     #4444,$TMDAT            ;LOAD DAC #2
  1533
   (1)                                  ;*      MOV     $TMDAT,@DAC2    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC2
  1534  003616  012737  007777  001462        MOV     #7777,$TMDAT            ;LOAD DAC #3
  1535
   (1)                                  ;*      MOV     $TMDAT,@DAC3    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC3
```

```
                                                MOV     #0,$TMDAT               ;CLEAR STATUS
                                        ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
                                                MOV     #1111,$GDDAT            ;LOAD EXPECTED
                                        ;*      MOV     @DAC0,$BDDAT    ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
                                                CMP     $GDDAT,$BDDAT           ;COMPARE
                                                BEQ     1$              ;;BR IF EQUAL
                                                ERROR   2                       ;ERROR, SELECTED DAC0 IN ERROR
                                1$:             MOV     #2222,$GDDAT            ;LOAD EXPECTED
                                        ;*      MOV     @DAC1,$BDDAT    ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
                                                CMP     $GDDAT,$BDDAT           ;COMPARE
                                                BEQ     2$              ;;BR IF EQUAL
                                                ERROR   3                       ;ERROR, SELECTED DAC1 IN ERROR
                                2$:             MOV     #4444,$GDDAT            ;LOAD EXPECTED
                                        ;*      MOV     @DAC2,$BDDAT    ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
                                                CMP     $GDDAT,$BDDAT           ;COMPARE
                                                BEQ     3$              ;;BR IF SAME
                                                ERROR   4                       ;ERROR, SELECTED DAC #2 IN ERROR
                                3$:             MOV     #7777,$GDDAT            ;LOAD EXPECTED
                                        ;*      MOV     @DAC3,$BDDAT    ;/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
                                                CMP     $GDDAT,$BDDAT           ;COMPARE
                                                BEQ     TST17           ;;BR IF SAME
                                                ERROR   5                       ;ERROR, SELECTED DAC #3 IN ERROR
                                        ;;******************************************************************
                                        ;*TEST 17       TEST THAT RESET SETS READY BIT
                                        ;;******************************************************************
                                TST17:  SCOPE
                                        MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
                                        JSR     PC,$RESET
                                        ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
                                        CMP     $GDDAT,$BDDAT           ;COMPARE
                                        BEQ     TST20           ;;BR IF SET
                                        ERROR   1                       ;RESET FAILED TO SET READY
                                        ;;******************************************************************
                                        ;*TEST 20       TEST THAT FAST INTENSIFY CAN BE SET AND CLEARED
                                        ;;******************************************************************
                                TST20:  SCOPE
                                        MOV     #BIT1,$TMDAT            ;LOAD BIT 1
                                        .*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
                                        MOV     #BIT7!BIT1,$GDDAT       ;LOAD EXPECTED
                                        ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
                                        CMP     $GDDAT,$BDDAT           ;COMAPRE
                                        BEQ     1$              ;;BR IF SAME
                                        ERROR   1                       ;ERROR, STATUS NOT - 202
                                1$:     MOV     #0,$TMDAT               ;CLEAR STATUS REGISTER
```

```
1578
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1579  004136  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1580
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1581  004154  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1582  004162  001401                            BEQ     TST21           ;;BR IF SAME
1583  004164  104001                            ERROR   1                       ;ERROR, STATUS NOT = 200
1584
1585                                    ;;*****************************************************************
 (3)                                    ;*TEST 21       TEST THAT MODE BIT 2 CAN BE SET AND CLEARED
 (3)                                    ;;*****************************************************************
 (2)  004166  000004                    TST21:  SCOPE
1586  004170  012737  000004  001462            MOV     #BIT2,$TMDAT            ;LOAD DISPLAY STATUS
1587
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1588  004206  012737  000204  001124            MOV     #BIT7!BIT2,$GDDAT       ;LOAD EXPECTED
1589
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1590  004224  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1591  004232  001401                            BEQ     1$              ;;BR IF EQUAL
1592  004234  104001                            ERROR   1                       ;ERROR, STATUS NOT = 204
1593  004236  012737  000000  001462  1$:       MOV     #0,$TMDAT               ;CLEAR STATUS
1594
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1595  004254  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1596
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1597  004272  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1598  004300  001401                            BEQ     TST22           ;;BR IF CLEARED
1599  004302  104001                            ERROR   1                       ;MODE FAILED TO CLEAR
1600
1601
1602                                    ;;*****************************************************************
 (3)                                    ;*TEST 22       TEST AT MODE BIT 3 CAN BE SET
 (3)                                    ;;*****************************************************************
 (2)  004304  000004                    TST22:  SCOPE
1603  004306  012737  000010  001462            MOV     #BIT3,$TMDAT            ;LOAD
1604
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1605  004324  012737  000210  001124            MOV     #BIT7!BIT3,$GDDAT       ;LOAD EXPECTED
1606
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1607  004342  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1608  004350  001401                            BEQ     1$              ;;BR IF EQUAL
1609  004352  104001                            ERROR   1                       ;ERROR, STATUS NOT = 210
1610
1611  004354  012737  000000  001462  1$:       MOV     #0,$TMDAT               ;CLEAR REG.
1612
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1613  004372  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1614
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1615  004410  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1616  004416  001401                            BEQ     TST23           ;;BR IF EQUAL
1617  004420  104001                            ERROR   1                       ;ERROR, STATUS NOT - 200
```

```
1618
1619                                    ;;*****************************************************************
 (3)                                    ;*TEST 23       TEST THAT EXT. DELAY (BIT 4) CAN BE SET AND CLEARED
 (3)                                    ;;*****************************************************************
 (2) 004422  000004                     TST23:  SCOPE
1620 004424  012737  000020  001462             MOV     #BIT4,$TMDAT            ;LOAD STATUS
1621
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1622 004442  012737  000220  001124             MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPCTED
1623
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1624 004460  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1625 004466  001401                             BEQ     1$              ;;BR IF SAME
1626 004470  104001                             ERROR   1               ;ERROR, EXT. DELAY BIT FAILED TO SET
1627 004472  012737  000000  001462  1$:       MOV     #0,$TMDAT               ;CLEAR BIT 4
1628
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1629 004510  012737  000200  001124             MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1630
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1631 004526  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1632 004534  001401                             BEQ     TST24           ;;BR IF SAME
1633 004536  104001                             ERROR   1               ;ERROR, EXT. DELAY BIT FAILED TO CLEAR
1634                                    ;;*****************************************************************
 (3)                                    ;*TEST 24       TEST THAT INTERRUPT ENABLE (BIT 6) CAN BE SET
 (3)                                    ;;*****************************************************************
 (2) 004540  000004                     TST24:  SCOPE
1635 004542  012737  000100  001462             MOV     #BIT6,$TMDAT            ;LOAD DISPLAY STATUS
1636
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1637 004560  012737  000300  001124             MOV     #BIT7.BIT6,$GDDAT       ;LOAD EXPECTED
1638
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1639 004576  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1640 004604  001401                             BEQ     TST25           ;;BR IF EQUAL
1641 004606  104001                             ERROR   1               ;ERROR, STATUS NOT = 300
1642
1643                                    ;;*****************************************************************
 (3)                                    ;*TEST 25       TEST THAT CHANNEL (BIT 9) CAN BE SET
 (3)                                    ;;*****************************************************************
 (2) 004610  000004                     TST25:  SCOPE
1644 004612  012737  001000  001462             MOV     #BIT9,$TMDAT            ;LOAD DISPLAY STATUS
1645
 (1)                                    ,*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1646 004630  012737  001200  001124             MOV     #BIT9!BIT7,$GDDAT       ;LOAD EXPECTED
1647
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1648 004646  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1649 004654  001401                             BEQ     TST26           ;;BR IF EQUAL
1650 004656  104001                             ERROR   1               ;ERROR, STATUS NOT = 1200
1651
1652                                    ;;*****************************************************************
 (3)                                    ;*TEST 26       TEST THAT STORE (BIT 10) CAN BE SET
 (3)                                    ;;*****************************************************************
 (2) 004660  000004                     TST26:  SCOPE
1653 004662  012737  002000  001462             MOV     #BIT10,$TMDAT           ;LOAD DISPLAY STATUS
```

```
1654
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1655  004700  012737  002200  001124            MOV     #BIT10!BIT7,$GDDAT      ;LOAD EXPECTED
1656
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1657  004716  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1658  004724  001401                            BEQ     TST27           ;;BR IF EQUAL
1659  004726  104001                            ERROR   1                       ;ERROR, STATUS NOT = 2200
1660
1661                                    ;;*******************************************************************
 (3)                                    ;*TEST 27       TEST THAT WRITE THRU (BIT 11) CAN BE SET
 (3)                                    ;;*******************************************************************
 (2)  004730  000004                    TST27:  SCOPE
1662  004732  012737  004000  001462            MOV     #BIT11,$TMDAT           ;LOAD DISPLAY STATUS
1663
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1664  004750  012737  004200  001124            MOV     #BIT11!BIT7,$GDDAT      ;LOAD EXPECTED
1665
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1666  004766  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1667  004774  001401                            BEQ     TST30           ;;BR IF EQUAL
1668  004776  104001                            ERROR   1                       ;ERROR, STATUS NOT = 4200
1669
1670                                    ;;*******************************************************************
 (3)                                    ;*TEST 30       TEST THAT  INTENSIFY BIT  SETS THAT THE READY BIT
 (3)                                    ;;*******************************************************************
 (2)  005000  000004                    TST30:  SCOPE
1671                                    ; AND THEN SETS AFTER A DELAY
1672  005002  012700  001000                    MOV     #1000,R0
1673  005006  005037  001124                    CLR     $GDDAT                  ;LOAD EXPECTED
1674  005012  012737  000001  001462            MOV     #BIT0,$TMDAT            ;INTENSIFY
1675
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1676  005030  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1677
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1678  005046  023737  001124  001126            CMP     $GDDAT,$BDDAT
1679  005054  001401                            BEQ     TST31           ;;BR IF EQUAL
1680  005056  104011                            ERROR   11                      ;READY FAILED TO SET AFTER A DELAY
1681                                                            ;IS STORAGE SCOPE CONNECTED BUT NOT TURNED ON ??
1682                                    ;;*******************************************************************
 (3)                                    ;*TEST 31       TEST THAT MODE 1 (INTENSIFY ON DAC0) SETS THE READY FLAG
 (3)                                    ;;*******************************************************************
 (2)  005060  000004                    TST31:  SCOPE
1683                                    ; AND THEN SETS IT
1684  005062  012700  001000                    MOV     #1000,R0                ;SET UP DELAY
1685  005066  012737  000204  001124            MOV     #BIT7!BIT2,$GDDAT       ;LOAD EXPECTED
1686  005074  012737  000004  001462            MOV     #BIT2,$TMDAT            ;LOAD MODE 1
1687
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1688
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1689  005122  105737  001126                    TSTB    $BDDAT                  ;TEST READY
1690  005126  100402                            BMI     2$              ;;BR IF READY STILL SET
1691  005130  104011                            ERROR   11                      ;ERROR, IN MODE 1 READY SHOULD NOT
1692                                                                            ; CLEAR UNTIL DAC0 IS LOADED
```

```
1693  005132  000425                          BR      1ST32           ;;BR TO SCOPE
1694
1695  005134  012737  000004  001124  2$:     MOV     #BIT2,$GDDAT            ;LOAD EXPECTED
1696  005142  012737  000000  001462          MOV     #0,$TMDAT               ;ADDRESS DAC 0
1697
 (1)                                  ;*      MOV     $TMDAT,@DAC0    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC0
1698
 (1)                                  ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1699
1700  005170  105737  001126                  TSTB    $BDDAT
1701  005174  100404                          BMI     TST32           ;;NEXT TEST
1702  005176  012737  000204  001124          MOV     #BIT7!BIT2,$GDDAT       ;LOAD EXPECTED
1703  005204  104011                          ERROR   11                      ;ERROR, READY FAILED TO SET
1704                                                                          ; AFTER MODE 1 OPERATION
1705
1706                                  ;;**********************************************************************
 (3)                                  ;*TEST 32        TEST THAT MODE 2 (INTENSIFY ON DAC1) SETS THE READY FLAG
 (3)                                  ;;**********************************************************************
 (2)  005206  000004                  TST32:  SCOPE
1707                                  ; AND THEN SETS IT
1708  005210  012700  001000                  MOV     #1000,R0                ;SET UP DELAY
1709  005214  012737  000210  001124          MOV     #BIT7!BIT3,$GDDAT       ;LOAD EXPECTED
1710  005222  012737  000010  001462          MOV     #BIT3,$TMDAT            ;LOAD MODE 2
1711
 (1)                                  ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1712
 (1)                                  ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1713  005250  105737  001126                  TSTB    $BDDAT                  ;TEST READY
1714  005254  100402                          BMI     2$              ;;BR IF SET
1715  005256  104011                          ERROR   11                      ;ERROR, IN MODE 2 READY SHOULD NOT CLEAR
1716                                                                          ;UNTIL DAC1 IS LOADED
1717  005260  000425                          BR      TST33           ;;BR TO SCOPE
1718  005262  012737  000010  001124  2$:     MOV     #BIT3,$GDDAT            ;LOAD EXPECTED
1719  005270  012737  000000  001462          MOV     #0,$TMDAT               ;ADDRESS DAC1
1720
 (1)                                  ;*      MOV     $TMDAT,@DAC1    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC1
1721
 (1)                                  ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1722  005316  105737  001126                  TSTB    $BDDAT
1723  005322  100404                          BMI     TST33           ;;NEXT TEST
1724  005324  012737  000210  001124          MOV     #BIT7!BIT3,$GDDAT       ;LOAD EXPECTED
1725  005332  104011                          ERROR   11                      ;ERROR, READY FAILED TO SET
1726                                                                          ; AFTER MODE 2 OPERATION
1727                                  ;;**********************************************************************
 (3)                                  ;*TEST 33        TEST WHEN ERASE IS SET, READY BIT CLEARS AND SET AFTER DELAY    (SW12=1)
 (3)                                  ;;**********************************************************************
 (2)  005334  000004                  TST33:  SCOPE
1728  005336  032777  010000  173574          BIT     #BIT12,@SWR             ;TEST BIT 12
1729  005344  001437                          BEQ     TST34           ;;BYPASS IF NO STORAGE SCOPE
1730  005346  012700  000010                  MOV     #10,R0
1731  005352  005037  014632                  CLR     TEMP                    ;CLEAR DELAY
1732  005356  012737  002000  001462          MOV     #BIT10,$TMDAT           ;SET STORE MODE
1733
 (1)                                  ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1734  005374  052737  010000  001462          BIS     #BIT12,$TMDAT           ;SET ERASE BIT
1735
```

```
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1736  005412                            1$:
 (1)
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1737  005422  105737  001462                    TSTB    $TMDAT
1738  005426  100406                            BMI     TST34           ;;BR IF SET
1739  005430  005337  014632                    DEC     TEMP                    ;DELAY
1740  005434  001366                            BNE     1$              ;;BR IF NOT READY
1741  005436  005300                            DEC     R0                      ;DECREMENT COUNTER
1742  005440  001364                            BNE     1$              ;;BR IF NOT DONE
1743  005442  104011                            ERROR   11                      ;ERROR, ERASE CLEARED READY AND FAILED
1744                                                                             ;TO SET READY AFTER A DELAY
1745                                                    ;SWR 12 = 1 AND NO STORAGE SCOPE CONNECTED ??
1746                                  ;;*****************************************************************
 (3)                                  ;*TEST 34       TEST THAT RESET CLEARS MODE, EXT. DELAY AND FAST INTENISIFY BITS
 (3)                                  ;;*****************************************************************
 (2)  005444  000004                  TST34:  SCOPE
1747  005446  012737  000036  001462            MOV     #BIT4!BIT3.BIT2!BIT1,$TMDAT
1748
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1749  005464  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1750  005472  004737  021700                    JSR     PC,$RESET
1751
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1752  005506  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1753  005514  001401                            BEQ     TST35           ;;BR IF EQUAL
1754  005516  104001                            ERROR   1                       ;ERROR, RESET FAILED TO CLEAR STATUS REG
1755                                  ;;*****************************************************************
 (3)                                  ;*TEST 35       TEST THAT RESET CLEARS INTERRUPT ENABLE, CHANNEL, STORE, WRITE THRU
 (3)                                  ;;*****************************************************************
 (2)  005520  000004                  TST35:  SCOPE
1756  005522  012737  007100  001462            MOV     #BIT11!BIT10!BIT9!BIT6,$TMDAT
1757
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1758  005540  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1759  005546  004737  021700                    JSR     PC,$RESET
1760
 (1)                                    ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1761  005562  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1762  005570  001401                            BEQ     TST36           ;;BR IF EQUAL
1763  005572  104001                            ERROR   1                       ;ERROR, RESET FAILED TO CLEAR STATUS
1764                                  ;;*****************************************************************
 (3)                                  ;*TEST 36       TEST THAT RESET CLEARS DAC0 REGISTER    (SW09=0)
 (3)                                  ;;*****************************************************************
 (2)  005574  000004                  TST36:  SCOPE
1765  005576  032777  001000  173334            BIT     #SW09,@SWR      ;TEST IF BIT 9 IS SET
 (3)  005604  001024                            BNE     TST37           ;;BR IF SET
1766  005606  012737  177777  001462            MOV     #-1,$TMDAT
1767
 (1)                                    ;*      MOV     $TMDAT,@DAC0    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC0
1768  005624  005037  001124                    CLR     $GDDAT                  ;LOAD EXPECTED
1769  005630  004737  021700                    JSR     PC,$RESET
1770
 (1)                                    ;*      MOV     @DAC0,$BDDAT    ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
1771  005644  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1772  005652  001401                            BEQ     TST37           ;;BR IF EQUAL
```

```
 1773  005654  104002                          ERROR   2                       ;ERROR, RESET FAILED TO CLEAR  DAC 0 REGISTER
 1774                                  ;;*****************************************************************
  (3)                                  ;*TEST 37       TEST THAT RESET CLEARS DAC1 REGISTER    (SW09=0)
  (3)                                  ;;*****************************************************************
  (2)  005656  000004                  TST37:  SCOPE
 1775  005660  032777  001000  173252          BIT     #SW09,@SWR      ;TEST IF BIT 9 IS SET
  (3)  005666  001024                          BNE     TST40           ;;BR IF SET
 1776  005670  012737  177777  001462          MOV     #-1,$TMDAT
 1777
  (1)                                  ;*      MOV     $TMDAT,@DAC1    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC1
 1778  005706  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED
 1779  005712  004737  021700                  JSR     PC,$RESET
 1780
  (1)                                  ;*      MOV     @DAC1,$BDDAT    ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
 1781  005726  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1782  005734  001401                          BEQ     TST40           ;;BR IF EQUAL
 1783  005736  104003                          ERROR   3                       ;ERROR, RESET FAILED TO CLEAR DAC1 REGISTER
 1784                                  ;;*****************************************************************
  (3)                                  ;*TEST 40       TEST THAT RESET CLEARS DAC #2 REGISTER  (SW09=0)
  (3)                                  ;;*****************************************************************
  (2)  005740  000004                  TST40:  SCOPE
 1785  005742  032777  001000  173170          BIT     #SW09,@SWR      ;TEST IF BIT 9 IS SET
  (3)  005750  001023                          BNE     TST41           ;;BR IF SET
 1786  005752  012737  177777  001462          MOV     #-1,$TMDAT              ;LOAD THE REGISTER
 1787
  (1)                                  ;*      MOV     $TMDAT,@DAC2    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC2
 1788  005770  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
 1789  005774  004737  021700                  JSR     PC,$RESET
 1790
  (1)                                  ;*      MOV     @DAC2,$BDDAT    ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
 1791  006010  005737  001126                  TST     $BDDAT
 1792  006014  001401                          BEQ     TST41           ;;BR IF CLEARED
 1793  006016  104004                          ERROR   4                       ;ERROR, RESET FAILED TO CLEAR DAC #2
 1794                                  ;;*****************************************************************
  (3)                                  ;*TEST 41       TEST THAT RESET CLEARS DAC #3 REGISTER  (SW09=0)
  (3)                                  ;;*****************************************************************
  (2)  006020  000004                  TST41:  SCOPE
 1795  006022  032777  001000  173110          BIT     #SW09,@SWR      ;TEST IF BIT 9 IS SET
  (3)  006030  001023                          BNE     TST42           ;;BR IF SET
 1796  006032  012737  177777  001462          MOV     #-1,$TMDAT              ;LOAD THE REGISTER
 1797
  (1)                                  ;*      MOV     $TMDAT,@DAC3    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC3
 1798  006050  005037  001124                  CLR     $GDDAT                  ;CLEAR THE EXPECTED
 1799  006054  004737  021700                  JSR     PC,$RESET
 1800
  (1)                                  ;*      MOV     @DAC3,$BDDAT    ;/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
 1801  006070  005737  001126                  TST     $BDDAT
 1802  006074  001401                          BEQ     TST42           ;;BR IF CLEARED
 1803  006076  104005                          ERROR   5                       ;ERROR, RESET FAILED TO CLEAR DAC #3
 1817                                  ;;*****************************************************************
  (4)                                  ;*TEST 42       TEST THAT RESET SET DAC0 TO 4000        (SW09=1)
  (4)                                  ;;*****************************************************************
  (3)  006100  000004                  TST42:  SCOPE
  (1)  006102  032777  001000  173030          BIT     #SW09,@SWR              ;TEST BIT 9
  (3)  006110  001425                          BEQ     TST43           ;;BR IF CLEARED
  (1)  006112  012737  003777  001462          MOV     #3777,$TMDAT            ;LOAD DAC0
```

```
 (2)
 (2)                                    ;*      MOV     $TMDAT,@DAC0    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC0
 (1)  006130  012737  004000  001124             MOV     #4000,$GDDAT            ;LOAD EXPTECTED
 (1)  006136  004737  021700                     JSR     PC,$RESET
 (2)
 (2)                                    ;*      MOV     @DAC0,$BDDAT    ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
 (1)  006152  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
 (3)  006160  001401                             BEQ     TST43           ;;BR IF SAME
 (1)  006162  104002                             ERROR   2                       ;RESET FAILED TO SET DAC0 TO 4000
1818                                    ;;*******************************************************************
 (4)                                    ;*TEST 43       TEST THAT RESET SET DAC1 TO 4000        (SW09=1)
 (4)                                    ;;*******************************************************************
 (3)  006164  000004                    TST43:  SCOPE
 (1)  006166  032777  001000  172744             BIT     #SW09,@SWR              ;TEST BIT 9
 (3)  006174  001425                             BEQ     TST44           ;;BR IF CLEARED
 (1)  006176  012737  003777  001462             MOV     #3777,$TMDAT            ;LOAD DAC1
 (2)
 (2)                                    ;*      MOV     $TMDAT,@DAC1    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC1
 (1)  006214  012737  004000  001124             MOV     #4000,$GDDAT            ;LOAD EXPTECTED
 (1)  006222  004737  021700                     JSR     PC,$RESET
 (2)
 (2)                                    ;*      MOV     @DAC1,$BDDAT    ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
 (1)  006236  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
 (3)  006244  001401                             BEQ     TST44           ;;BR IF SAME
 (1)  006246  104003                             ERROR   3                       ;RESET FAILED TO SET DAC1 TO 4000
1819                                    ;;*******************************************************************
 (4)                                    ;*TEST 44       TEST THAT RESET SET DAC2 TO 4000        (SW09=1)
 (4)                                    ;;*******************************************************************
 (3)  006250  000004                    TST44:  SCOPE
 (1)  006252  032777  001000  172660             BIT     #SW09,@SWR              ;TEST BIT 9
 (3)  006260  001425                             BEQ     TST45           ;;BR IF CLEARED
 (1)  006262  012737  003777  001462             MOV     #3777,$TMDAT            ;LOAD DAC2
 (2)
 (2)                                    ;*      MOV     $TMDAT,@DAC2    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC2
 (1)  006300  012737  004000  001124             MOV     #4000,$GDDAT            ;LOAD EXPTECTED
 (1)  006306  004737  021700                     JSR     PC,$RESET
 (2)
 (2)                                    ;*      MOV     @DAC2,$BDDAT    ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
 (1)  006322  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
 (3)  006330  001401                             BEQ     TST45           ;;BR IF SAME
 (1)  006332  104004                             ERROR   4                       ;RESET FAILED TO SET DAC2 TO 4000
1820                                    ;;*******************************************************************
 (4)                                    ;*TEST 45       TEST THAT RESET SET DAC3 TO 4000        (SW09=1)
 (4)                                    ;;*******************************************************************
 (3)  006334  000004                    TST45:  SCOPE
 (1)  006336  032777  001000  172574             BIT     #SW09,@SWR              ;TEST BIT 9
 (3)  006344  001425                             BEQ     TST46           ;;BR IF CLEARED
 (1)  006346  012737  003777  001462             MOV     #3777,$TMDAT            ;LOAD DAC3
 (2)
 (2)                                    ;*      MOV     $TMDAT,@DAC3    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC3
 (1)  006364  012737  004000  001124             MOV     #4000,$GDDAT            ;LOAD EXPTECTED
 (1)  006372  004737  021700                     JSR     PC,$RESET
 (2)
 (2)                                    ;*      MOV     @DAC3,$BDDAT    ;/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
 (1)  006406  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
 (3)  006414  001401                             BEQ     TST46           ;;BR IF SAME
```

```
 (1)  006416  104005                          ERROR   5                       ;RESET FAILED TO SET DAC3 TO 4000
1821                                  ;;*****************************************************************
 (3)                                  ;*TEST 46       TEST THAT EXTERNAL DELAY DOES NOT SET DISPLAY READY     SW10=0
 (3)                                  ;;*****************************************************************
 (2)  006420  000004                  TST46:  SCOPE
1822  006422  032777  002000  172510          BIT     #SW10,@SWR              ;TEST IF SW 10 IS SET
1823  006430  001051                          BNE     TST47           ;;BR IF EXT. DELAY SWITCH IS SET
1824  006432  012737  000020  001462          MOV     #BIT4,$TMDAT            ;LOAD EXT. DELAY ENABLE
1825
 (1)                                  .*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1826
 (1)                                  ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1827  006460  105237  001462                  INCB    $TMDAT
1828
 (1)                                  ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1829  006474  012700  000000                  MOV     #0,R0                   ;LOAD DELAY
1830  006500                               1$:
 (1)
 (1)                                  ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1831  006510  105737  001462                  TSTB    $TMDAT
1832  006514  100403                          BMI     2$                      ;BR IF SET
1833  006516  005300                          DEC     R0                      ;DELAY
1834  006520  001367                          BNE     1$                      ;BR IF NOT FINISHED
1835  006522  000414                          BR      TST47           ;;
1836  006524  012737  000020  001124  2$.     MOV     #BIT4,$GDDAT            ;LOAD EXPECTED
1837
 (1)                                  ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1838  006542  023737  001124  001126          CMP     $GDDAT,$BDDAT
1839  006550  001401                          BEQ     TST47           ;;BRANCH IF EQ
1840  006552  104011                          ERROR   11                      ;EXT. DELAY CAUSED READY TO SET
1841                                                                  ;WHEN THE OPERATOR (VIA SW10) SAID NO EXT. DELAY WAS CON
1842
1843                                  ;;*****************************************************************
 (3)                                  ;*TEST 47       TEST THE EXTERNAL DELAY DOES SET DISPLAY READY  SW10=1
 (3)                                  ;;*****************************************************************
 (2)  006554  000004                  TST47:  SCOPE
1844  006556  032777  002000  172354          BIT     #SW10,@SWR              ;TEST SR BIT 10
1845  006564  001442                          BEQ     TST50           ;;BR IF CLEARED
1846  006566  012737  000020  001462          MOV     #BIT4,$TMDAT            ;ENABLE EXT. DELAY
1847
 (1)                                  ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1848  006604  012700  000000                  MOV     #0,R0                   ;LOAD COUNTER
1849  006610  005237  001462                  INC     $TMDAT                  ;ENABLE DISPLAY
1850
 (1)                                  ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1851  006624                               1$:
 (1)
 (1)                                  ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1852  006634  105737  001464                  TSTB    STAT
1853  006640  100412                          BMI     2$                      ;BR IF DONE
1854  006642  005300                          DEC     R0                      ;DELAY
1855  006644  001367                          BNE     1$                      ;BR IF NOT DONE
1856  006646  012737  000220  001124          MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPECTED VALUE
1857
 (1)                                  ;*      MOV     @STAT,$BDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $BDDAT.
1858  006664  104011                          ERROR   11                      ;EXT. DELAY FAILED TO SET READY
```

```
1859                                                    ;WHEN THE OPERATOR (VIA SW 10=1) SAID AN EXT. DELAY WAS
1860  006666  004737  021700          2$:     JSR     PC,$RESET                       ;ENSURE CLEARED AA11-K
1861
1862                                  ;;*********************************************************************
 (3)                                  ;*TEST 50       DETERMINE IF MORE AA11-K'S REMAIN TO BE TESTED
 (3)                                  ;;*********************************************************************
 (2)  006672  000004                  TST50:  SCOPE
1863  006674  004737  021700                  JSR     PC,$RESET
1864  006700  005237  001204                  INC     $UNIT                   ;MORE UNITS?
1865  006704  123737  001204  014642          CMPB    $UNIT,EVER
1866  006712  001431                          BEQ     $EOP                    ;;BR IF NONE
1867  006714  063737  001454  001464          ADD     VADDR,STAT              ;UPDATE ADDRESS
1868  006722  063737  001454  001466          ADD     VADDR,DAC0
1869  006730  063737  001454  001470          ADD     VADDR,DAC1
1870  006736  063737  001454  001472          ADD     VADDR,DAC2
1871  006744  063737  001454  001474          ADD     VADDR,DAC3
1872  006752  063737  001456  001476          ADD     VVECT,IV                ;UPDATE VECTOR
1873  006760  063737  001456  001500          ADD     VVECT,IVS
1874  006766  005037  001102                  CLR     $TSTNM
1875  006772  000137  002542                  JMP     LTEST                   ;TEST ANOTHER AA11-K
1876
1877                                  .SBTTL  END OF PASS ROUTINE
 (1)
 (2)                                  ;;*********************************************************************
 (1)                                  ;*INCREMENT THE PASS NUMBER ($PASS)
 (1)                                  ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
 (1)                                  ;*IF THERES A MONITOR GO TO IT
 (1)                                  ;*IF THERE ISN'T JUMP TO MTEST
 (1)
 (1)  006776                          $EOP:
 (1)  006776  000004                          SCOPE
 (1)  007000  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
 (1)  007004  005237  001200                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
 (1)  007010  042737  100000  001200          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
 (1)  007016  005327                          DEC     (PC)+           ;;LOOP?
 (1)  007020  000001                  $EOPCT: .WORD   1
 (1)  007022  003022                          BGT     $DOAGN          ;;YES
 (1)  007024  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
 (1)  007026  000001                  $ENDCT: .WORD   1
 (1)  007030  007020                          $EOPCT
 (1)  007032  104401  007077                  TYPE    ,$ENDMG         ;;TYPE "END PASS #"
 (2)  007036  013746  001200                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
 (2)  007042  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
 (1)  007044  104401  007074                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
 (1)  007050  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
 (1)  007054  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
 (1)  007056  000005                          RESET                   ;;CLEAR THE WORLD
 (1)  007060  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
 (1)  007062  000240                          NOP                     ;;SAVE ROOM
 (1)  007064  000240                          NOP                     ;;FOR
 (1)  007066  000240                          NOP                     ;;ACT11
 (1)  007070                          $DOAGN:
 (1)  007070  000137                          JMP     @(PC)+          ;;RETURN
 (1)  007072  002360                  $RTNAD: .WORD   MTEST
 (1)  007074     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
 (1)  007077     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
```

```
   (1)  007104  050040  051501  020123
   (1)  007112  000043
1886  007114  012706  001100          VISUAL: MOV     #STACK,SP               ;LOAD SP
1887  007120  005737  014644                  TST     EVER1                   ;TEST IF TYPED ONCE
1888  007124  001006                          BNE     PIC0                    ;BR IF YES
1889  007126  104401                          TYPE
1890  007130  013103                          MES6                    ;HEADER ABOUT SCOPE ADJ.
1891  007132  104401                          TYPE
1892  007134  013133                          MES15                   ;TEXT ABOUT SWR
1893  007136  005237  014644                  INC     EVER1                   ;SET FLAG
1894                                          .SBTTL  DISPLAY HORIZONTAL LINE
1895  007142  013737  001466  007230  PIC0:   MOV     DAC0,R0X
1896  007150  013737  001470  007232          MOV     DAC1,R1Y
1897  007156  032777  000010  171754          BIT     #SW03,@SWR              ;TEST SR BIT 3
1898  007164  001406                          BEQ     2$                      ;BR IF DAC #0 AND #1
1899  007166  013737  001472  007230          MOV     DAC2,R0X                ;USE DAC #2 AND 3
1900  007174  013737  001474  007232          MOV     DAC3,R1Y
1901  007202  012737  000005  014630  2$:     MOV     #5,TICKS                ;LOAD TIMER FOR CP TYPE
1902  007210  004737  012354                  JSR     PC,CHTIME               ;CHANGE TIMER FOR CP TYPE
1903  007214  004737  007330          1$:     JSR     PC,PBB
1904  007220  004737  012246                  JSR     PC,TIMER                ;DONE ?
1905  007224  000773                          BR      1$
1906  007226  000405                          BR      PIC1
1907  007230  000000          R0X:    .WORD   0
1908  007232  000000          R1Y:    .WORD   0
1909  007234  000000          R0Y:    .WORD   0
1910  007236  000000          R1X:    .WORD   0
1911  007240  000000          R2Y:    .WORD   0
1912                                          .SBTTL  DISPLAY A VERTICAL LINE
1913  007242  013737  001470  007230  PIC1:   MOV     DAC1,R0X
1914  007250  013737  001466  007232          MOV     DAC0,R1Y
1915  007256  032777  000010  171654          BIT     #SW03,@SWR              ;TEST SR BIT 3
1916  007264  001406                          BEQ     2$                      ;BR IF DAC #0 AND 1
1917  007266  013737  001474  007230          MOV     DAC3,R0X                ;USE DAC #2 AND 3
1918  007274  013737  001472  007232          MOV     DAC2,R1Y
1919  007302  012737  000005  014630  2$:     MOV     #5,TICKS                ;LOAD TIME
1920  007310  004737  012354                  JSR     PC,CHTIME               ;CHANGE TIMER FOR CP TYPE
1921  007314  004737  007330          1$:     JSR     PC,PBB
1922  007320  004737  012246                  JSR     PC,TIMER                ;DONE ?
1923  007324  000773                          BR      1$
1924  007326  000516                          BR      PIC3
1925  007330  012737  000000  001462  PBB:    MOV     #0,$TMDAT
1926
   (1)                                  ;*     MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1927  007346  032777  002000  171564          BIT     #SW10,@SWR              ;TEST IS SW10 =1
1928  007354  001413                          BEQ     10$                     ;BR IF NOT
1929
   (1)                                  ;*     MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1930  007366  052737  000020  001462          BIS     #BIT4,$TMDAT
1931
   (1)                                  ;*     MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1932  007404  013704  001464          10$:    MOV     STAT,R4
1933  007410  012703  007777                  MOV     #7777,R3                ;SET HIGH LIMIT
1934  007414  012702  000001                  MOV     #1,R2                   ;INITIALIZE INCREMENTS BETWEEN POINTS
1935  007420  032777  001000  171512          BIT     #SW09,@SWR              ;TEST SW 9 = 1
1936  007426  001042                          BNE     2$                      ;BR IF YES
```

```
1937  007430  012737  000400  001462          MOV     #400,$TMDAT
1938
 (1)                                ;*      MOV     $TMDAT,@R1Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1Y
1939  007446  012737  000000  001462          MOV     #0,$TMDAT
1940
 (1)                                ;*      MOV     $TMDAT,@R0X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R0X
1941  007464  060237  001462          1$:     ADD     R2,$TMDAT               ;INCREMENT
1942
 (1)                                ;*      MOV     $TMDAT,@R0X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R0X
1943
 (1)                                ;*      MOV     @STAT,$GDDAT    ;/READ DEVICE REG STAT,PUT DATA IN $GDDAT.
1944  007510  005237  001124                  INC     $GDDAT
1945
 (1)                                ;*      MOV     $GDDAT,@STAT    ;/ PUT DATA FROM $GDDAT TO DEVICE REG STAT
1946  007524  023703  001462                  CMP     $TMDAT,R3               ;DONE ALL POINTS?
1947  007530  001355                          BNE     1$              ;NO
1948  007532  000207                          RTS     PC
1949  007534  012737  000000  001462  2$:     MOV     #0,$TMDAT                       ;LOAD TWO'S COMP. ORIGIN
1950
 (1)                                ;*      MOV     $TMDAT,@R1Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1Y
1951
 (1)                                ;*      MOV     $TMDAT,@R0X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R0X
1952  007562  000740                          BR      1$
1953                                          .SBTTL  PINCUSHION TEST (DISPLAY SQUARE)
1954
1955  007564  012737  000000  001462  PIC3:   MOV     #0,$TMDAT
1956
 (1)                                ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1957  007602  012737  000010  014630          MOV     #10,TICKS
1958  007610  004737  012354                  JSR     PC,CHTIME
1959  007614  013737  001466  007236          MOV     DAC0,R1X
1960  007622  013737  001470  007240          MOV     DAC1,R2Y
1961  007630  032777  000010  171302          BIT     #SW03,@SWR      ;TEST SR BIT 3
1962  007636  001406                          BEQ     1$              ;BR IF DAC #0 AND 1
1963  007640  013737  001472  007236          MOV     DAC2,R1X                ;USE DAC #2 AND 3
1964  007646  013737  001474  007240          MOV     DAC3,R2Y
1965  007654  013703  001464          1$:     MOV     STAT,R3
1966  007660  012704  000020                  MOV     #20,R4
1967  007664  032777  001000  171246  P3:     BIT     #SW09,@SWR      ;TEST BIT 9
1968  007672  001414                          BEQ     1$
1969  007674  012737  004000  001462          MOV     #4000,$TMDAT    ;LOAD TWO'S COMP ORIGIN
1970
 (1)                                ;*      MOV     $TMDAT,@R1X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
1971
 (1)                                ;*      MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
1972  007722  000412                          BR      2$
1973  007724  005037  001462          1$:     CLR     $TMDAT          ;CLEAR AXIS
1974
 (1)                                ;*      MOV     $TMDAT,@R1X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
1975
 (1)                                ;*      MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
1976                                ;DRAW BOTTOM LINE
1977  007750  012700  000377          2$:     MOV     #377,R0
1978  007754  032777  002000  171156          BIT     #SW10,@SWR              ;TEST IF SW10 =1
1979  007762  001413                          BEQ     P3A                     ;BR IF NOT
1980
```

```
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1981  007774  052737  040000  001462            BIS     #BIT14,$TMDAT
1982
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1983  010012                           P3A:
 (1)
 (1)                                    ;*      MOV     @R1X,$TMDAT     ;/READ DEVICE REG R1X,PUT DATA IN $TMDAT.
1984  010022  060437  001462                    ADD     R4,$TMDAT
1985
 (1)                                    ;*      MOV     $TMDAT,@R'X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
1986
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1987  010046  005237  001462                    INC     $TMDAT
1988
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
1989                                                                    ;WAIT FOR READY
1990  010062  005300                           DEC     R0
1991  010064  001352                           BNE     P3A             ;NO
1992                                   ;DRAW RIGHT LINE
1993  010066  012700  000377                   MOV     #377,R0
1994  010072                           P3B:
 (1)
 (1)                                    ;*      MOV     @R2Y,$TMDAT     ;/READ DEVICE REG R2Y,PUT DATA IN $TMDAT.
1995  010102  060437  001462                    ADD     R4,$TMDAT
1996
 (1)                                    ;*      MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
1997
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
1998  010126  005237  001462                    INC     $TMDAT
1999
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2000                                                                    ;WAIT FOR READY
2001  010142  005300                           DEC     R0
2002  010144  001352                           BNE     P3B             ;NO
2003                                   ;DRAW TOP LINE
2004  010146  012700  000377                   MOV     #377,R0
2005  010152                           P3C:
 (1)
 (1)                                    ;*      MOV     @R1X,$TMDAT     ;/READ DEVICE REG R1X,PUT DATA IN $TMDAT.
2006  010162  160437  001462                    SUB     R4,$TMDAT
2007
 (1)                                    ;*      MOV     $TMDAT,@R'X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
2008
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
2009  010206  005237  001462                    INC     $TMDAT
2010
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2011                                                                    ;WAIT FOR READY
2012  010222  005300                           DEC     R0
2013  010224  001352                           BNE     P3C             ;NO
2014                                   ;DRAW LEFT LINE
2015  010226  012700  000377                   MOV     #377,R0
2016  010232                           P3D:
 (1)
 (1)                                    ;*      MOV     @R2Y,$TMDAT     ;/READ DEVICE REG R2Y,PUT DATA IN $TMDAT.
2017  010242  160437  001462                    SUB     R4,$TMDAT
```

```
 2018
  (1)                                   ;*      MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
 2019
  (1)                                   ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
 2020  010266  005237  001462                   INC     $TMDAT
 2021
  (1)                                   ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
 2022
 2023                                                                   ;WAIT FOR READY
 2024  010302  005300                           DEC     R0
 2025  010304  001352                           BNE     P3D             ;NO
 2026  010306  004737  012246                   JSR     PC,TIMER
 2027  010312  000401                           BR      1$
 2028  010314  000402                           BR      PIC4
 2029  010316  000137  007664           1$:     JMP     P3
 2030                                          .SBTTL  PLOT AN X
 2031
 2032  010322  012737  000000  001462  PIC4:   MOV     #0,$TMDAT
 2033
  (1)                                   ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
 2034  010340  012737  000010  014630           MOV     #10,TICKS
 2035  010346  004737  012354                   JSR     PC,CHTIME       ;CHANGE TIMER FOR CP TYPE
 2036  010352  013737  001466  007236  PIC4B:  MOV     DAC0,R1X
 2037  010360  013737  001470  007240           MOV     DAC1,R2Y
 2038  010366  032777  000010  170544           BIT     #SW03,@SWR      ;TEST SWR BIT 3
 2039  010374  001406                           BEQ     1$              ;BR IF DAC #0 AND 1
 2040  010376  013737  001472  007236           MOV     DAC2,R1X                ;USE DAC #2 AND 3
 2041  010404  013737  001474  007240           MOV     DAC3,R2Y
 2042  010412  013703  001464           1$:     MOV     STAT,R3
 2043  010416  012704  000004                   MOV     #4,R4
 2044  010422  032777  001000  170510  P4:     BIT     #SW09,@SWR      ;TEST BIT 9
 2045  010430  001414                           BEQ     1$
 2046  010432  012737  004000  001462           MOV     #4000,$TMDAT
 2047
  (1)                                   ;*      MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
 2048
  (1)                                   ;*      MOV     $TMDAT,@R1X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
 2049  010460  000412                           BR      2$
 2050  010462  005037  001462           1$:     CLR     $TMDAT
 2051
  (1)                                   ;*      MOV     $TMDAT,@R1X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
 2052
  (1)                                   ;*      MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
 2053
 2054                                   ;PLOT LINE BEGINNING IN LOWER LEFT CORNER
 2055  010506  012700  001777           2$:     MOV     #1777,R0
 2056  010512  032777  002000  170420           BIT     #SW10,@SWR              ;TEST IF SW10 -1
 2057  010520  001413                           BEQ     P4A                     ;BR IF NOT
 2058
  (1)                                   ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
 2059  010532  052737  000020  001462           BIS     #BIT4,$TMDAT
 2060
  (1)                                   ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
 2061  010550                           P4A:
  (1)
  (1)                                   ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
```

```
2062  010560  005237  001462              INC     $TMDAT
2063
 (1)                                ;*     MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2064
 (1)                                ;*     MOV     @R2Y,$TMDAT     ;/READ DEVICE REG R2Y,PUT DATA IN $TMDAT.
2065  010604  060437  001462              ADD     R4,$TMDAT
2066
 (1)                                ;*     MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
2067
 (1)                                ;*     MOV     $TMDAT,@R1X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
2068                                                                ;+4 TO X
2069  010630  005300                       DEC     R0
2070  010632  001346                       BNE     P4A             ;NO
2071                                ;PLOT LINE BEGINNING IN UPPER LEFT CORNER
2072  010634  012737  007774  001462       MOV     #7774,$TMDAT
2073
 (1)                                ;*     MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
2074  010652  005037  001462              CLR     $TMDAT
2075
 (1)                                ;*     MOV     $TMDAT,@R1X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
2076  010666  012700  001777              MOV     #1777,R0
2077  010672                       P4B:
 (1)
 (1)                                ;*     MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
2078  010702  005237  001462              INC     $TMDAT
2079
 (1)                                ;*     MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2080
 (1)                                ;*     MOV     @R2Y,$TMDAT     ;/READ DEVICE REG R2Y,PUT DATA IN $TMDAT.
2081  010726  160437  001462              SUB     R4,$TMDAT
2082
 (1)                                ;*     MOV     $TMDAT,@R2Y     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R2Y
2083
 (1)                                ;*     MOV     @R1X,$TMDAT     ;/READ DEVICE REG R1X,PUT DATA IN $TMDAT.
2084  010752  060437  001462              ADD     R4,$TMDAT
2085
 (1)                                ;*     MOV     $TMDAT,@R1X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1X
2086  010766  005300                       DEC     R0
2087  010770  001340                       BNE     P4B             ;NO
2088  010772  004737  012246              JSR     PC,TIMER
2089  010776  000402                       BR      1$
2090  011000  000137  007142              JMP     PICO
2091  011004  000137  010422       1$:    JMP     P4
2092
```

```
2094
2095  011010  000000                    VCXTMP: 0
2096  011012  000000                    VCYTMP: 0
2097  011014  000000                    YPOS:   0                       ;CONTAINS Y POSITION AT ANY GIVEN TIME
2098  011016  000000                    XPOS:   0                       ;CONTAINS X POSITION AT ANY GIVEN TIME
2099  011020  000000                    CHRCOL: 0
2100  011022  000000                    YPT:    0
2101
2102
2103                                    .SBTTL  MANUAL DISPLAY ROUTINE
2104
2105  011024                            PIC6:
2106  011024  012706  001100            FULRMP: MOV     #STACK,SP               ;LOAD POINTER
2107  011030  004737  002014                    JSR     PC,LDTRAP               ;LOAD BUS ADDRESS
2108  011034  032777  002000  170076            BIT     #SW10,@SWR      ;TEST SR 10=1
2109  011042  001413                            BEQ     1$              ;BR IF NOT
2110
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
2111  011054  052737  000020  001462            BIS     #BIT4,$TMDAT
2112
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2113  011072  013737  001466  007230  1$:       MOV     DAC0,R0X                ;GET BUS ADDRESS
2114  011100  004737  011144                    JSR     PC,10$                  ;LOAD THE RAMP ON DAC #1
2115  011104  013737  001470  007230            MOV     DAC1,R0X                ;GET BUS ADDRESS
2116  011112  004737  011144                    JSR     PC,10$                  ;LOAD THE RAMP ON DAC #1
2117  011116  013737  001472  007230            MOV     DAC2,R0X                ;GET BUS ADDRESS
2118  011124  004737  011144                    JSR     PC,10$                  ;LOAD THE RAMP ON DAC #2
2119  011130  013737  001474  007230            MOV     DAC3,R0X                ;GET THE BUS ADDRESS
2120  011136  004737  011144                    JSR     PC,10$                  ;LOAD THE RAMP ON DAC #3
2121  011142  000753                            BR      1$                      ;BR BACK
2122
2123  011144  005037  001462           10$:     CLR     $TMDAT
2124
 (1)                                    ;*      MOV     $TMDAT,@R0X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R0X
2125  011160  062737  000010  001462    11$:     ADD     #10,$TMDAT
2126
 (1)                                    ;*      MOV     $TMDAT,@R0X     ;/ PUT DATA FROM $TMDAT TO DEVICE REG R0X
2127  011176  005737  001462                    TST     $TMDAT                  ;TEST IF DONE
2128  011202  001366                            BNE     11$                     ;BR IF NOT
2129  011204  004737  011410                    JSR     PC,ACTRLC               ;TEST FOR CTRL C
2130  011210  000207                            RTS     PC                      ;EXIT
2131  011212  000240                            NOP
2132  011214  000240                            NOP
2133
2134                                    .SBTTL  MANUAL LOGIC TEST
2135
2136  011216  012706  001100           MANUL:   MOV     #STACK,SP               ;LOAD STACK
2137  011222  004737  002014                    JSR     PC,LDTRAP               ;LOAD BUS ADDRESS
2138  011226  104401                            TYPE
2139  011230  014343                            MANHED
2140  011232  004737  011410           1$:      JSR     PC,ACTRLC               ;TEST FOR CTRL C
2141  011236  104407                            CKSWR
2142  011240  017700  167674                    MOV     @SWR,R0                 ;READ THE SWITCHES
2143  011244  010001                            MOV     R0,R1                   ;COPY
2144  011246  042700  017777                    BIC     #17777,R0               ;MASK OFF BITS
2145  011252  001003                            BNE     2$                      ;BR IF NOT DAC 0
```

```
2146  011254  013702  001466              MOV     DAC0,R2                 ;LOAD DAC 0 BUS ADDRESS
2147  011260  000424                      BR      10$
2148  011262  022700  020000      2$:     CMP     #BIT13,R0               ;TEST FOR DAC 1
2149  011266  001003                      BNE     3$                      ;BR IF NOT
2150  011270  013702  001470              MOV     DAC1,R2                 ;LOAD DAC 1 BUS ADDRESS
2151  011274  000416                      BR      10$
2152  011276  022700  040000      3$:     CMP     #BIT14,R0               ;TEST FOR DAC 2
2153  011302  001003                      BNE     4$                      ;BR IF NOT
2154  011304  013702  001472              MOV     DAC2,R2                 ;LOAD DAC 2 BUS ADDRESS
2155  011310  000410                      BR      10$
2156  011312  022700  060000      4$:     CMP     #BIT14!BIT13,R0         ;TEST FOR DAC 3
2157  011316  001003                      BNE     5$                      ;BR IF NOT
2158  011320  013702  001474              MOV     DAC3,R2                 ;LOAD DAC 3 BUS ADDRESS
2159  011324  000402                      BR      10$
2160  011326  013702  001464      5$:     MOV     STAT,R2                 ;LOAD STATUS REG. BUS ADDRESS
2161
2162  011332  010137  001124      10$:    MOV     R1,$GDDAT
2163  011336  010237  001126              MOV     R2,$BDDAT
2164
 (1)                                  ;*  MOV     $GDDAT,@$BDDAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG $BDDAT
2165  011352  013703  001460              MOV     DELAY,R3                ;PRESET THE DELAY
2166  011356  005303              11$:    DEC     R3                      ;DELAY
2167  011360  001376                      BNE     11$
2168  011362  005037  001124              CLR     $GDDAT                  ;CLEAR THE REGISTER
2169
 (1)                                  ;*  MOV     $GDDAT,@$BDDAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG $BDDAT
2170  011366  013703  001460              MOV     DELAY,R3                ;LOAD THE PRESET
2171  011402    5303              12$:    DEC     R3                      ;DELAY
2172  011404  001376                      BNE     12$
2173  011406  000711                      BR      1$
2174
2175                                  ;TEST FOR CTRL C  -- DONT WAIT
2176
2177  011410  105777  167530      ACTRLC: TSTB    @$TKS           ;INPUT FLAG ?
2178  011414  100014                      BPL     1$                      ;NO
2179  011416  017737  167524  011450      MOV     @$TKB,10$               ;READ THE CHARACTER
2180  011424  042737  177600  011450      BIC     #177600,10$
2181  011432  022737  000003  011450      CMP     #3,10$                  ;TEST FOR CTRL C
2182  011440  001002                      BNE     1$                      ;BR IF NOT
2183  011442  000137  002346              JMP     CTRLC
2184  011446  000207              1$:     RTS     PC                      ;EXIT
2185  011450  000000              10$:    0
2186
2187                                  ;WAIT FOR A "SPACE" CHARACTER
2188
2189  011452  105777  167466      CSPACE: TSTB    @$TKS           ;WAIT FOR CHAR
2190  011456  100375                      BPL     CSPACE
2191  011460  017737  167462  011530      MOV     @$TKB,10$               ;READ CHAR
2192  011466  042737  177600  011530      BIC     #177600,10$             ;MASK
2193  011474  022737  000003  011530      CMP     #3,10$                  ;TEST FOR CTRL C
2194  011502  001002                      BNE     1$
2195  011504  000137  002346              JMP     CTRLC
2196  011510  022737  000040  011530  1$: CMP     #40,10$                 ;TEST FOR SPACE
2197  011516  001001                      BNE     2$
2198  011520  000207                      RTS     PC                      ;EXIT
2199  011522  104401              2$:     TYPE
```

```
                                        QMARK
                                        BR      CSPACE

                        10$:            0

                                        .SBTTL  MANUAL DAC CALIBRATION

                        CALDAC: MOV     #STACK,SP               ;LOAD STACK POINTER
                                JSR     PC,LDTRAP               ;LOAD BUS ADDRESSES
                        1$:     CKSWR                           ;TEST FOR CTRL G
                                JSR     PC,ACTRLC               ;TEST FOR CTRL C
                                MOV     @SWR,R0                 ;READ SWITCHES
                                MOV     R0,$TMDAT               ;LOAD DAC #0

                        ;*      MOV     $TMDAT,@DAC0    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC0

                        ;*      MOV     $TMDAT,@DAC1    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC1

                        ;*      MOV     $TMDAT,@DAC2    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC2

                        ;*      MOV     $TMDAT,@DAC3    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC3
                                BR      1$

                                .SBTTL  DYNAMIC DAC CALIBRATION

                        DYNCAL: MOV     #STACK,SP               ;LOAD STACK POINTER
                                JSR     PC,LDTRAP               ;LOAD BUS ADDRESSES
                        1$:     CKSWR                           ;TEST FOR CTRL G
                                JSR     PC,ACTRLC               ;TEST FOR CTRL C
                                MOV     @SWR,R0                 ;READ SWR
                                JSR     PC,10$                  ;LOAD THE SWR VALUE TO ALL DACS
                                CLR     R0                      ;CLEAR R0
                                JSR     PC,10$                  ;LOAD ALL DAC'S WITH 0
                                BR      1$

                        10$:

                        ;*      MOV     $TMDAT,@DAC0    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC0

                        ;*      MOV     $TMDAT,@DAC1    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC1

                        ;*      MOV     $TMDAT,@DAC2    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC2

                        ;*      MOV     $TMDAT,@DAC3    ;/ PUT DATA FROM $TMDAT TO DEVICE REG DAC3
                                MOV     #20,R0                  ;LOAD DELAY COUNTER
                        11$:    DEC     R0                      ;DELAY
                                BPL     11$                     ;WAIT
                                RTS     PC                      ;EXIT

                                .SBTTL  SUBROUTINE TO ERASE STORAGE SCOPE SCREEN
                        CLRVC:  CKSWR
                                BIT     #BIT5,@SWR      ;TEST SR BIT 5
                                BEQ     3$              ;BR IF NOT A STORAGE SCOPE
                                MOV     #BIT10,$TMDAT   ;ERASE THE SCREEN
```

```
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2252
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
2253  011772  052737  010000  001462             BIS     #BIT12,$TMDAT
2254
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2255  012010  012700  000020                     MOV     #20,R0          ;SET UP DELAY
2256  012014  005001                             CLR     R1
2257  012016  032777  000040  167114  1$:       BIT     #BIT5,@SWR      ;TEST IF NOT STORAGE SCOPE
2258  012024  001430                             BEQ     3$
2259
 (1)                                    ;*      MOV     @STAT,$TMDAT    ;/READ DEVICE REG STAT,PUT DATA IN $TMDAT.
2260  012036  105737  001462                     TSTB    $TMDAT
2261  012042  100421                             BMI     3$              ;BRANCH IF SET
2262  012044  005301                             DEC     R1              ;DELAY
2263  012046  001363                             BNE     1$
2264  012050  004737  011410                     JSR     PC,ACTRLC       ;TEST FOR CTRL C
2265  012054  005300                             DEC     R0              ;DELAY
2266  012056  001357                             BNE     1$
2267  012060  037727  167054  010000             BIT     @SWR,#SW12      ;TEST INHIBIT PRINTOUT
2268  012066  001002                             BNE     2$
2269  012070  104401                             TYPE
2270  012072  013036                             MES3
2271  012074  104407                     2$:     CKSWR                   ;TEST FOR CTRL C
2272  012076  005777  167036                     TST     @SWR            ;TEST @SWR
2273  012102  100001                             BPL     3$
2274  012104  000000                             HALT                    ;ERASE RETURN FAILED TO SET READY
2275  012106  000207                     3$:     RTS     PC
2276
2277                                            .SBTTL  SUBROUTINE TO DRAW A HORIZONTAL LINE
2278  012110  012737  000000  001462  LOADVC:   MOV     #0,$TMDAT       ;CLEAR STATUS
2279
 (1)                                    ;*      MOV     $TMDAT,@STAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG STAT
2280  012126  012737  007777  014634             MOV     #7777,TEMP1
2281  012134  013700  001464                     MOV     STAT,R0
2282  012140  032777  000010  166772             BIT     #SW03,@SWR      ;TEST SR BIT 3
2283  012146  001405                             BEQ     4$              ;BR IF DAC #0 AND 1
2284  012150  013701  001472                     MOV     DAC2,R1 ;LOAD X AXIS
2285  012154  013702  001474                     MOV     DAC3,R2 ;LOAD Y AXIS
2286  012160  000404                             BR      5$
2287  012162  013701  001466             4$:     MOV     DAC0,R1
2288  012166  013702  001470                     MOV     DAC1,R2
2289  012172  012710  002000             5$:     MOV     #BIT10,(0)      ;SET STORE MODE
2290  012176  013712  014634                     MOV     TEMP1,(2)
2291  012202  012711  007777             1$:     MOV     #7777,(1)
2292  012206  000402                             BR      3$
2293  012210  162711  000010             2$:     SUB     #10,(1)
2294  012214  005210                     3$:     INC     (0)
2295  012216  105710                             TSTB    (0)
2296  012220  100376                             BPL     .-2
2297  012222  022711  000007                     CMP     #7,(1)
2298  012226  001370                             BNE     2$
2299  012230  104407                             CKSWR                   ;TEST FOR CTRL G
2300  012232  004737  011410                     JSR     PC,ACTRLC               ;TEST FOR CTRL C
2301  012236  162712  000010                     SUB     #10,(2)
2302  012242  100357                             BPL     1$
```

```
2303  012244  000207                        RTS     PC
2304
2305                                        .SBTTL  TIMER ROUTINE FOR VISUAL TEST PATTERNS
2306                                ; ENTER VIA JSR PC,TIMER
2307
2308  012246  104407                TIMER:  CKSWR
2309  012250  004737  011410               JSR     PC,ACTRLC               ;TEST FOR CTRL C
2310  012254  017737  166660  014626       MOV     @SWR,TIMSV
2311
2312  012262  032737  000400  014626 TIMERA: BIT     #BIT8,TIMSV
2313  012270  001006                       BNE     TIMER2          ;BIT 8 SET ?
2314  012272  005337  014630               DEC     TICKS           ;NO, DECREMENT TICKS
2315  012276  001002                       BNE     TIMER1
2316  012300  062716  000002               ADD     #2,(6)          ;ADD 2 TO STACK POINTER
2317  012304  000207                TIMER1: RTS     PC              ;RETURN
2318
2319                                ; SWR 8=1 SELECT TEST TO LOCK ON
2320                                ; SWR 2-0= TEST NUMBER
2321
2322  012306  042737  177770  014626 TIMER2: BIC     #177770,TIMSV
2323  012314  006337  014626               ASL     TIMSV
2324  012320  062737  012342  014626       ADD     #ROUTPT,TIMSV
2325  012326  017737  002274  014626       MOV     @TIMSV,TIMSV
2326  012334  022600                       CMP     (SP)+,R0
2327  012336  000177  002264        TIMER4: JMP     @TIMSV
2328
2329  012342  007142                ROUTPT: PIC0                    ;DISPLAY A HORIZONTAL LINE
2330  012344  007242                        PIC1                    ;DISPLAY A VERTICAL LINE
2331  012346  007554                        PIC3                    ;DISPLAY A SQUARE
2332  012350  010322                        PIC4                    ;DISPALY A "X"
2333  012352  011024                        PIC6                    ;DISPLAY CHARACTER SET
2334
2335  012354  013737  012400  014636 CHTIME: MOV     CPTYPE,BRLEV1
2336  012362  005337  014636        1$:    DEC     BRLEV1
2337  012366  001403                       BEQ     2$
2338  012370  006337  014630               ASL     TICKS
2339  012374  000772                       BR      1$
2340  012376  000207                2$:    RTS     PC
2341
2342  012400  000001                CPTYPE: 1
2343
2344
2345                                        .SBTTL  ASCII MESSAGES
2346
2347  012402  005015  041412  046122 TITLE:  .ASCII  <15><12><12>'CRLPBB0   LPA/AA11-K DIAGNOSTIC TEST  '<15><12>
      012410  041120  030102  020040
      012416  046040  040520  040457
      012424  030501  026461  020113
      012432  044504  043501  047516
      012440  052123  041511  052040
      012446  051505  020124  006440
      012454     012
2348  012455     123  046105  041505         .ASCII  /SELECT TESTS BY TYPING A TWO LETTER I.D./<15><12>
      012462  020124  042524  052123
      012470  020123  054502  052040
      012476  050131  047111  020107
```

```
                012504  020101  053524  020117
                012512  042514  052124  051105
                012520  044440  042056  006456
                012526     012
2349            012527     101  004514  040472          .ASCII  /AL     :AUTO LOGIC TEST/<15><12>
                012534  052125  020117  047514
                012542  044507  020103  042524
                012550  052123  005015
2350            012554  042101  035011  052501          .ASCII  /AD     :AUTO DISPLAY TEST (DISPLAY SCOPE CONNECTED)/<15><12>
                012562  047524  042040  051511
                012570  046120  054501  052040
                012576  051505  020124  042050
                012604  051511  046120  054501
                012612  051440  047503  042520
                012620  041440  047117  042516
                012626  052103  042105  006451
                012634     012
2351            012635     115  004514  046472          .ASCII  /ML     :MANUAL LOGIC TEST (SWR 15-13 = REGISTER  12-0 = DATA)/<15><12>
                012642  047101  040525  020114
                012650  047514  044507  020103
                012656  042524  052123  024040
                012664  053523  020122  032461
                012672  030455  020063  020075
                012700  042522  044507  052123
                012706  051105  020040  031061
                012714  030055  036440  042040
                012722  052101  024501  005015
2352            012730  042115  035011  040515          .ASCII  /MD     :MANUAL DISPLAY (NO DISPLAY SCOPE)/<15><12>
                012736  052516  046101  042040
                012744  051511  046120  054501
                012752  024040  047516  042040
                012760  051511  046120  054501
                012766  051440  047503  042520
                012774  006451     012
2353            012777     115  004503  046472          .ASCII  /MC     :MANUAL CALIBRATION LOOP/<15><12>
                013004  047101  040525  020114
                013012  040503  044514  051102
                013020  052101  047511  020116
                013026  047514  050117  005015
2354            013034  000056                          .ASCIZ  /./
2355            013036  005015  051105  051501  MES3:   .ASCIZ  <15><12>''ERASE RETURN FAILED TO SET READY''<15><12>
                013044  020105  042522  052524
                013052  047122  043040  044501
                013060  042514  020104  047524
                013066  051440  052105  051040
                013074  040505  054504  005015
                013102     000
2356            013103     015  051412  047503  MES6:   .ASCIZ  <15><12>'SCOPE ADJUSTMENT TEST'
                013110  042520  040440  045104
                013116  051525  046524  047105
                013124  020124  042524  052123
                013132     000
2357            013133     015  051412  051127  MES15:  .ASCII  <15><12>/SWR8 AND 0 THRU 2 CONTROL PATTERN/<15><12>
                013140  020070  047101  020104
                013146  020060  044124  052522
                013154  031040  041440  047117
```

```
                013162  051124  046117  050040
                013170  052101  042524  047122
                013176  005015
 2358   013200  053523  020122  044502          .ASCII  'SWR BIT 3 SELECTS DAC 0+1 OR DAC 2+3'<15><12>
                013206  020124  020063  042523
                013214  042514  052103  020123
                013222  040504  020103  025460
                013230  020061  051117  042040
                013236  041501  031040  031453
                013244  005015
 2359   013246  053523  020122  040502          .ASCII  /SWR BIT 5 SELECTS STORAGE SCOPE MODE/<15><12>
                013254  020124  020065  042523
                013262  042514  052103  020123
                013270  052123  051117  043501
                013276  020105  041523  050117
                013304  020105  047515  042504
                013312  005015
 2360   013314  053523  020122  044502          .ASCII  /SWR BIT 7 ENABLES VERTICAL SETTLING TEST/<15><12>
                013322  020124  020067  047105
                013330  041101  042514  020123
                013336  042526  052122  041511
                013344  046101  051440  052105
                013352  046124  047111  020107
                013360  042524  052123  005015
 2361   013366  053523  020122  044502          .ASCIZ  /SWR BIT 9 SELECTS TWO'S COMPLEMENT DISPLAY MODE/<15><12>
                013374  020124  020071  042523
                013402  042514  052103  020123
                013410  053524  023517  020123
                013416  047503  050115  042514
                013424  042515  052116  042040
                013432  051511  046120  054501
                013440  046440  042117  006505
                013446  000012
 2362   013450  052123  052101  051525  EM1:    .ASCIZ  /STATUS REGISTER IN ERROR/
                013456  051040  043505  051511
                013464  042524  020122  047111
                013472  042440  051122  051117
                013500     000
 2363   013501     104  041501  020060  EM2:    .ASCIZ  /DAC0 REGISTER IN ERROR/
                013506  042522  044507  052123
                013514  051105  044440  020116
                013522  051105  047522  000122
 2364   013530  040504  030503  051040  EM3:    .ASCIZ  /DAC1 REGISTER IN ERROR/
                013536  043505  051511  042524
                013544  020122  047111  042440
                013552  051122  051117     000
 2365   013557     104  041501  020062  EM4:    .ASCIZ  /DAC2 REGISTER IN ERROR/
                013564  042522  044507  052123
                013572  051105  044440  020116
                013600  051105  047522  000122
 2366   013606  040504  031503  051040  EM5:    .ASCIZ  /DAC3 REGISTER IN ERROR/
                013614  043505  051511  042524
                013622  020122  047111  042440
                013630  051122  051117     000
 2367   013635     102  051525  052040  EM10:   .ASCIZ  /BUS TIME-OUT WHEN REFERENCING THE AA11-K/
                013642  046511  026505  052517
```

```
                013650  020124  044127  047105
                013656  051040  043105  051105
                013664  047105  044503  043516
                013672  052040  042510  040440
                013700  0305C1  026461  000113
  2368          013706  040501  030461  045455  EM11:   .ASCIZ  /AA11-K READY BIT ERROR/
                013714  051040  040505  054504
                013722  041040  052111  042440                ;
                013730  051122  051117     000
  2369          013735     117  042516  041040  EM13:   .ASCIZ  /ONE BIT THE DUST/
                013742  052111  052040  042510
                013750  042040  051525  000124
  2370          013756  051105  050122  004503  DH1:    .ASCIZ  /ERRPC  IBASE   EXPECT    BAD/
                013764  041111  051501  020105
                013772  020040  054105  042520
                014000  052103  020040  020040
                014006  040502  000104
  2371          014012  051105  050122  004503  DH6:    .ASCIZ  /ERRPC  IBASE/
                014020  041111  051501  000105
  2372          014026  051105  050122  004503  DH13:   .ASCIZ  /ERRPC  IBASE   # FIRST # NOW/
                014034  041111  051501  004505
                014042  020043  044506  051522
                014050  004524  020043  047516
                014056  000127
  2373          014060  051105  050122  004503  DH14:   .ASCIZ  /ERRPC  IBASE   GOOD    BAD/
                014066  041111  051501  004505
                014074  047507  042117  041011
                014102  042101     000
  2374
  2375          014105     015     012     077  QMARK:  .BYTE   15,12,77,15,12,0
                014110     015     012     000
  2376          014113     136     103     015  CONTC:  .BYTE   136,103,15,12,56,0
                014116     012     056     000
  2377          014121     015     012          FOUND1: .BYTE   15,12
  2378          014123     120  047522  051107          .ASCIZ  /PROGRAM DETECTED /
                014130  046501  042040  052105
                014136  041505  042524  020104
                014144     000
  2379          014145     050  024470  020040  FOUND2: .ASCIZ  /(8)   AA11-K(S)  /
                014152  040440  030501  026461
                014160  024113  024523  020040
                014166     000
  2380          014167     015     012     000          .BYTE   15,12,0
  2381          014172     015     012          SELD01: .BYTE   15,12
  2382          014174  042523  020124  053523          .ASCIZ  /SET SWITCH TO SELECT DAC 0 AND 1/<15><12>
                014202  052111  044103  052040
                014210  020117  042523  042514
                014216  052103  042040  041501
                014224  030040  040440  042116
                014232  030440  005015     000
  2383          014237     015     012          SELD23: .BYTE   15,12
  2384          014241     123  052105  051440          .ASCIZ  /SET SWITCH TO SELECT DAC 2 AND 3/<15><12>
                014246  044527  041524  020110
                014254  047524  051440  046105
                014262  041505  020124  040504
                014270  020103  020062  047101
```

```
                014276  020104  006463  000012
2385            014304     015     012          CALDON: .BYTE   15,12
2386            014306  052501  047524  041440          .ASCIZ  /AUTO CALIBRATION COMPLETED/<15><12>
                014314  046101  041111  040522
                014322  044524  047117  041440
                014330  046517  046120  052105
                014336  042105  005015     000
2387            014343     015     012          MANHED: .BYTE   15,12
2388            014345     123  020127  032461          .ASCII  /SW 15-13 SELECT THE REGISTER/<15><12>
                014352  030455  020063  042523
                014360  042514  052103  052040
                014366  042510  051040  043505
                014374  051511  042524  006522
                014402     012
2389            014403     123  020127  031061          .ASCII  /SW 12-00 ARE LOADED INTO THE SELECTED REGISTER/<15><12>
                014410  030055  020060  051101
                014416  020105  047514  042101
                014424  042105  044440  052116
                014432  020117  044124  020105
                014440  042523  042514  052103
                014446  042105  051040  043505
                014454  051511  042524  006522
                014462     012
2390            014463     000                          .BYTE   0
2391                                                    .EVEN
2392            014464  001116  001464  001124  D 1.    $ERRPC,STAT,$GDDAT,$BDDAT,0
                014472  001126  000000
2393            014476  001116  001466  001124  DT2:    $ERRPC,DAC0,$GDDAT,$BDDAT,0
                014504  001126  000000
2394            014510  001116  001470  001124  DT3:    $ERRPC,DAC1,$GDDAT,$BDDAT,0
                014516  001126  000000
2395            014522  001116  001472  001124  DT4:    $ERRPC,DAC2,$GDDAT,$BDDAT,0
                014530  001126  000000
2396            014534  001116  001474  001124  DT5:    $ERRPC,DAC3,$GDDAT,$BDDAT,0
                014542  001126  000000
2397            014546  001116  001464  000000  DT6:    $ERRPC,STAT,0
2398            014554  001116  001464  014642  DT13:   $ERRPC,STAT,EVER,$UNIT,0
                014562  001204  000000
2399            014566  001116  001464  001124  DT14:   $ERRPC,STAT,$GDDAT,$BDDAT,0
                014574  001126  000000
2400            014600  000000  000000  000000  DF0:    0,0,0,0,0,0,0,0
                014606  000000  000000  000000
                014614  000000  000000
2401            014620  000000                  LOW:    0
2402            014622  000000                  HIGH:   0
2403            014624  000010                  INCR:   10
2404            014626  000000                  TIMSV:  0
2405            014630  000000                  TICKS:  0
2406            014632  000000                  TEMP:   0
```

```
 2408  014634  000000                  TEMP1:  0                       ;TEMPORARY STORAGE
 2409  014636  000000                  BRLEV1: 0
 2410  014640  000000                  BRLEV2: 0
 2411  014642  000000                  EVER:   0
 2412  014644  000000                  EVER1:  0
 2413  014646  000000                  OPRIN:  0
 2414  014650  000000                  P7CNT:  0
 2415  014652  000000                  P7PNT:  0
 2416
 2417                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
  (1)
  (2)                                  ;;**********************************************************************
  (1)                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
  (1)                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
  (1)                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
  (1)                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
  (1)                                  ;*REPLACED WITH SPACES.
  (1)                                  ;*CALL:
  (1)                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
  (1)                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
  (1)
  (1)  014654                          $TYPDS:
  (3)  014654  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (3)  014656  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (3)  014660  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (3)  014662  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (3)  014664  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
  (1)  014666  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
  (1)  014672  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
  (1)  014676  100004                          BPL     1$              ;;BR IF INPUT IS POS.
  (1)  014700  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
  (1)  014702  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
  (1)  014710  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
  (1)  014712  012703  015070                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
  (1)  014716  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
  (1)  014722  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
  (1)  014724  016001  015060                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
  (1)  014730  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
  (1)  014732  002402                          BLT     4$              ;;BR IF DONE
  (1)  014734  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
  (1)  014736  000774                          BR      3$
  (1)  014740  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
  (1)  014742  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
  (1)  014744  001002                          BNE     5$              ;;FALL THROUGH IF 0
  (1)  014746  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
  (1)  014750  100407                          BMI     7$              ;;BR IF YES
  (1)  014752  106316                  5$:     ASLB    (SP)            ;;MSD?
  (1)  014754  103003                          BCC     6$              ;;BR IF NO
  (1)  014756  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
  (1)  014764  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
  (1)  014770  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
  (1)  014774  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
  (1)  014776  005720                          TST     (R0)+           ;;JUST INCREMENTING
  (1)  015000  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
  (1)  015004  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
  (1)  015006  003002                          BGT     8$              ;;GO TO EXIT
```

```
(1)  015010  010502                        MOV     R5,R2           ;;GET THE LSD
(1)  015012  000764                        BR      6$              ;;GO CHANGE TO ASCII
(1)  015014  105726                 8$:    TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  015016  100003                        BPL     9$              ;;BR IF NO
(1)  015020  116663  177777  177776       MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  015026  105013                 9$:    CLRB    (R3)            ;;SET THE TERMINATOR
(3)  015030  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  015032  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  015034  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  015036  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  015040  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  015042  104401  015070                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  015046  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  015054  012616                        MOV     (SP)+,(SP)
(1)  015056  000002                        RTI                     ;;RETURN TO USER
(1)  015060  023420                 $DTBL: 10000.
(1)  015062  001750                        1000.
(1)  015064  000144                        100.
(1)  015066  000012                        10.
(1)  015070  000004                 $DBLK: .BLKW   4
2418                                .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)                                 ;;**********************************************************************
(1)                                 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)                                 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)                                 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                 ;*SW14=1        LOOP ON TEST
(1)                                 ;*SW08=1        LOOP ON TEST IN SWR<7:0>
(1)                                 ;*CALL
(1)                                 ;*      SCOPE           ;;SCOPE=IOT
(1)
(1)  015100                         $SCOPE:
(1)  015100  104407                        CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
(2)  015102  004737  011410                JSR     PC,ACTRLC
(1)  015106  032777  040000  164024 1$:    BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
(1)  015114  001040                        BNE     $OVER           ;;YES IF SW14=1
(1)                                 ;#####START OF CODE FOR THE XOR TESTER#####
(1)  015116  000416                 $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
(1)                                                                 ;;THIS INSTRUCTION TO A "NOP" (NOP-240)
(1)  015120  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  015124  012737  015144  000004        MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
(1)  015132  005737  177060                TST     @#177060        ;;TIME OUT ON XOR?
(1)  015136  012637  000004                MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  015142  000414                        BR      $SVLAD          ;;GO TO THE NEXT TEST
(1)  015144  022626                 5$:    CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
(1)  015146  012637  000004                MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  015152  000421                        BR      $OVER           ;;LOOP ON THE PRESENT TEST
(1)  015154                         6$:;#####END OF CODE FOR THE XOR TESTER#####
(1)  015154  032777  000400  163756        BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
(1)  015162  001404                        BEQ     $SVLAD          ;;BR IF NO
(1)  015164  127737  163750  001102        CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
(1)  015172  001411                        BEQ     $OVER           ;;BR IF YES
(1)  015174  105237  001102         $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
(1)  015200  113737  001102  001176        MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1)  015206  011637  001106                MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
```

```
(1)  015212  105037  001103                    CLRB    $ERFLG           ;;ZERO THE ERROR FLAG
(1)  015216  013777  001102  163716  $OVER:    MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER
(1)  015224  013716  001106                    MOV     $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
(1)  015230  000002                            RTI                      ;;FIXES PS
2419                                .SBTTL  ERROR HANDLER ROUTINE
(1)
(2)                                 ;;**************************************** ********************
(1)                                 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)                                 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)                                 ;*AND GO TO $ERRTYP ON ERROR
(1)                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                 ;*SW15=1        HALT ON ERROR
(1)                                 ;*SW13=1        INHIBIT ERROR TYPEOUTS
(1)                                 ;*CALL
(1)                                 ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1)  015232                         $ERROR:
(1)  015232  104407                            CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
(2)  015234  004737  011410                    JSR     PC,ACTRLC
(1)  015240  105237  001103            7$:     INCB    $ERFLG           ;;SET THE ERROR FLAG
(1)  015244  001775                            BEQ     7$               ;;DON'T LET THE FLAG GO TO ZERO
(1)  015246  013777  001102  163666            MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  015254  005237  001112                    INC     $ERTTL           ;;INC THE ERROR COUNT
(1)  015260  011637  001116                    MOV     (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  015264  162737  000002  001116            SUB     #2,$ERRPC
(1)  015272  117737  163620  001114            MOVB    @$ERRPC,$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  015300  032777  020000  163632            BIT     #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
(1)  015306  001004                            BNE     20$              ;;SKIP TYPEOUTS
(1)  015310  004737  015362                    JSR     PC,$ERRTYP       ;;GO TO USER ERROR ROUTINE
(1)  015314  104401  001167                    TYPE    ,$CRLF
(1)  015320                         20$:
(1)  015320  122737  000001  001212            CMPB    #APTENV,$ENV     ;;RUNNING IN APT MODE
(1)  015326  001007                            BNE     2$               ;;NO,SKIP APT ERROR REPORT
(1)  015330  113737  001114  015342            MOVB    $ITEMB,21$       ;;SET ITEM NUMBER AS ERROR NUMBER
(1)  015336  004737  017342                    JSR     PC,$ATY4         ;;REPORT FATAL ERROR TO APT
(1)  015342     000                 21$:       .BYTE   0
(1)  015343     000                            .BYTE   0
(1)  015344  000777                 22$:       BR      22$              ;;APT ERROR LOOP
(1)  015346  005777  163566          2$:       TST     @SWR             ;;HALT ON ERROR
(1)  015352  100002                            BPL     3$               ;;SKIP IF CONTINUE
(1)  015354  000000                            HALT                     ;;HALT ON ERROR!
(1)  015356  104407                            CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
(1)  015360                         3$:
(1)  015360  000002                            RTI                      ;;RETURN
2420                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                 ;;**************************************************************
(1)                                 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1)                                 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1)                                 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  015362                         $ERRTYP:
(1)  015362  104401  001167                    TYPE    ,$CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  015366  010046                            MOV     R0,-(SP)         ;;SAVE R0
(1)  015370  005000                            CLR     R0               ;;PICKUP THE ITEM INDEX
(1)  015372  153700  001114                    BISB    @#$ITEMB,R0
```

```
(1)  015376  001004                    BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                             ;;TYPE THE PC OF THE ERROR
(2)  015400  013746  001116            MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                             ;;ERROR ADDRESS
(2)  015404  104402                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  015406  000426                    BR      6$              ;;GET OUT
(1)  015410  005300            1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1)  015412  006300                    ASL     R0              ;;      WORK FOR THE ERROR TABLE
(1)  015414  006300                    ASL     R0
(1)  015416  006300                    ASL     R0
(1)  015420  062700  001254            ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(1)  015424  012037  015434            MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
(1)  015430  001404                    BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1)  015432  104401                    TYPE                    ;;TYPE THE "ERROR MESSAGE"
(1)  015434  000000            2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
(1)  015436  104401  001167            TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEFD"
(1)  015442  012037  015452    3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
(1)  015446  001404                    BEQ     5$              ;;SKIP TYPEOUT IF 0
(1)  015450  104401                    TYPE                    ;;TYPE THE "DATA HEADER"
(1)  015452  000000            4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
(1)  015454  104401  001167            TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  015460  011000            5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
(1)  015462  001004                    BNE     7$              ;;GO TYPE THE DATA
(1)  015464  012600            6$:     MOV     (SP)+,R0        ;;RESTORE R0
(1)  015466  104401  001167            TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  015472  000207                    RTS     PC              ;;RETURN
(1)  015474                    7$:
(2)  015474  013046                    MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
(2)  015476  104402                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  015500  005710                    TST     (R0)            ;;IS THERE ANOTHER NUMBER?
(1)  015502  001770                    BEQ     6$              ;;BR IF NO
(1)  015504  104401  015512            TYPE    ,8$             ;;TYPE TWO(2) SPACES
(1)  015510  000771                    BR      7$              ;;LOOP
(1)  015512  020040      000   8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
(1)          015516                    .EVEN
```

```
                        .SBTTL  POWER DOWN AND UP ROUTINES

                        ;;*****************************************************************
                        ;POWER DOWN ROUTINE
015516  012737  015662  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
015524  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
015532  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
015534  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015536  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
015540  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
015542  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
015544  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
015546  017746  163366                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
015552  010637  015666                  MOV     SP,$SAVR6       ;;SAVE SP
015556  012737  015570  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
015564  000000                          HALT
015566  000776                          BR      .-2             ;;HANG UP

                        ;;*****************************************************************
                        ;POWER UP ROUTINE
015570  012737  015662  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
015576  013706  015666                  MOV     $SAVR6,SP       ;;GET SP
015602  005037  015666                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
015606  005237  015666          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
015612  001375                          BNE     1$              ;;OF  WORD
015614  012677  163320                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
015620  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
015622  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
015624  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
015626  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
015630  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
015632  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
015634  012737  015516  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
015642  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
015650  104401                          TYPE                    ;;REPORT THE POWER FAILURE
015652  015670          $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
015654  012716                  MOV     (PC)+,(SP)      ;;RESTART AT BEGIN1
015656  001520          $PWRAD: .WORD   BEGIN1          ;;RESTART ADDRESS
015660  000002                  RTI
015662  000000          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
015664  000776                  BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
015666  000000          $SAVR6: 0                       ;;PUT THE SP HERE
015670  005015  042522  052123  PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>
015676  051101  044524  043516
015704  040440  052106  051105
015712  040440  050040  053517
015720  051105  043040  044501
015726  052514  042522  005015
015734  000012

                        .EVEN
```

```
2426                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
 (1)                                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
 (1)                                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
 (1)                                    ;*CALL:
 (1)                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 (1)                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
 (1)                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
 (1)                                    ;*      .BYTE   M               ;;M=1 OR 0
 (1)                                    ;*                                      ;;1=TYPE LEADING ZEROS
 (1)                                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
 (1)                                    ;*
 (1)                                    ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
 (1)                                    ;*$TYPOS OR $TYPOC
 (1)                                    ;*CALL:
 (1)                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 (1)                                    ;*      TYPON                   ;;CALL FOR TYPEOUT
 (1)                                    ;*
 (1)                                    ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
 (1)                                    ;*CALL:
 (1)                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 (1)                                    ;*      TYPOC                   ;;CALL FOR TYPEOUT
 (1)
 (1)  015736  017646  000000            $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
 (1)  015742  116637  000001  016161            MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
 (1)  015750  112637  016163                    MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
 (1)  015754  062716  000002                    ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
 (1)  015760  000406                            BR      $TYPON
 (1)  015762  112737  000001  016161  $TYPOC:  MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
 (1)  015770  112737  000006  016163            MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
 (1)  015776  112737  000005  016160  $TYPON:  MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
 (1)  016004  010346                            MOV     R3,-(SP)        ;;SAVE R3
 (1)  016006  010446                            MOV     R4,-(SP)        ;;SAVE R4
 (1)  016010  010546                            MOV     R5,-(SP)        ;;SAVE R5
 (1)  016012  113704  016163                    MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
 (1)  016016  005404                            NEG     R4
 (1)  016020  062704  000006                    ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
 (1)  016024  110437  016162                    MOVB    R4,$OMODE       ;;SAVE IT FOR USE
 (1)  016030  113704  016161                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 (1)  016034  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 (1)  016040  005003                            CLR     R3              ;;CLEAR THE OUTPUT WORD
 (1)  016042  006105                    1$:     ROL     R5              ;;ROTATE MSB INTO "C"
 (1)  016044  000404                            BR      3$              ;;GO DO MSB
 (1)  016046  006105                    2$:     ROL     R5              ;;FORM THIS DIGIT
 (1)  016050  006105                            ROL     R5
 (1)  016052  006105                            ROL     R5
 (1)  016054  010503                            MOV     R5,R3
 (1)  016056  006103                    3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
 (1)  016060  105337  016162                    DECB    $OMODE          ;;TYPE THIS DIGIT?
 (1)  016064  100016                            BPL     7$              ;;BR IF NO
 (1)  016066  042703  177770                    BIC     #177770,R3      ;;GET RID OF JUNK
 (1)  016072  001002                            BNE     4$              ;;TEST FOR 0
 (1)  016074  005704                            TST     R4              ;;SUPPRESS THIS 0?
 (1)  016076  001403                            BEQ     5$              ;;BR IF YES
```

```
(1)  016100  005204                    4$:   INC   R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  016102  052703  000060              BIS   #'0,R3             ;;MAKE THIS DIGIT ASCII
(1)  016106  052703  000040              5$:   BIS   #' ,R3             ;;MAKE ASCII IF NOT ALREADY
(1)  016112  110337  016156              MOVB  R3,8$              ;;SAVE FOR TYPING
(1)  016116  104401  016156              TYPE  ,8$                ;;GO TYPE THIS DIGIT
(1)  016122  105337  016160              7$:   DECB  $OCNT            ;;COUNT BY 1
(1)  016126  003347                      BGT   2$                 ;;BR IF MORE TO DO
(1)  016130  002402                      BLT   6$                 ;;BR IF DONE
(1)  016132  005204                      INC   R4                 ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  016134  000744                      BR    2$                 ;;GO DO THE LAST DIGIT
(1)  016136  012605                      6$:   MOV   (SP)+,R5           ;;RESTORE R5
(1)  016140  012604                      MOV   (SP)+,R4           ;;RESTORE R4
(1)  016142  012603                      MOV   (SP)+,R3           ;;RESTORE R3
(1)  016144  016666  000002  000004      MOV   2(SP),4(SP)        ;;SET THE STACK FOR RETURNING
(1)  016152  012616                      MOV   (SP)+,(SP)
(1)  016154  000002                      RTI                      ;;RETURN
(1)  016156     000                      8$:   .BYTE 0                ;;STORAGE FOR ASCII DIGIT
(1)  016157     000                      .BYTE 0                  ;;TERMINATOR FOR TYPE ROUTINE
(1)  016160     000                      $OCNT:  .BYTE 0              ;;OCTAL DIGIT COUNTER
(1)  016161     000                      $OFILL: .BYTE 0              ;;ZERO FILL SWITCH
(1)  016162  000000                      $OMODE: .WORD 0              ;;NUMBER OF DIGITS TO TYPE
2427                                     .SBTTL TYPE ROUTINE
(1)
(2)                                      ;;**************************************************************
(1)                                      ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)                                      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)                                      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)                                      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)                                      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)                                      ;*
(1)                                      ;*CALL:
(1)                                      ;*1) USING A TRAP INSTRUCTION
(1)                                      ;*      TYPE    ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)                                      ;*OR
(1)                                      ;*      TYPE
(1)                                      ;*      MESADR
(1)                                      ;*
(1)
(1)  016164  105737  001157              $TYPE:  TSTB  $TPFLG          ;;IS THERE A TERMINAL?
(1)  016170  100002                      BPL   1$                 ;;BR IF YES
(1)  016172  000000                      HALT                     ;;HALT HERE IF NO TERMINAL
(1)  016174  000430                      BR    3$                 ;;LEAVE
(1)  016176  010046                      1$:   MOV   R0,-(SP)           ;;SAVE R0
(1)  016200  017600  000002              MOV   @2(SP),R0          ;;GET ADDRESS OF ASCIZ STRING
(1)  016204  122737  000001  001212      CMPB  #APTENV,$ENV       ;;RUNNING IN APT MODE
(1)  016212  001011                      BNE   62$                ;;NO,GO CHECK FOR APT CONSOLE
(1)  016214  132737  000100  001213      BITB  #APTSPOOL,$ENVM    ;;SPOOL MESSAGE TO APT
(1)  016222  001405                      BEQ   62$                ;;NO,GO CHECK FOR CONSOLE
(1)  016224  010037  016234              MOV   R0,61$             ;;SETUP MESSAGE ADDRESS FOR APT
(1)  016230  004737  017332              JSR   PC,$ATY3           ;;SPOOL MESSAGE TO APT
(1)  016234  000000                      61$:  .WORD 0                ;;MESSAGE ADDRESS
(1)  016236  132737  000040  001213      62$.  BITB  #APTCSUP,$ENVM    ;;APT CONSOLE SUPPRESSED
(1)  016244  001003                      BNE   60$                ;;YES,SKIP TYPE OUT
(1)  016246  112046                      2$:   MOVB  (R0)+,-(SP)        ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1)  016250  001005                      BNE   4$                 ;;BR IF IT ISN'T THE TERMINATOR
(1)  016252  005726                      TST   (SP)+              ;;IF TERMINATOR POP IT OFF THE STACK
```

```
(1)  016254  012600                  60$:   MOV     (SP)+,R0          ;;RESTORE R0
(1)  016256  062716  000002          3$:    ADD     #2,(SP)           ;;ADJUST RETURN PC
(1)  016262  000002                         RTI                       ;;RETURN
(1)  016264  122716  000011          4$:    CMPB    #HT,(SP)          ;;BRANCH IF <HT>
(1)  016270  001430                         BEQ     8$
(1)  016272  122716  000200                 CMPB    #CRLF,(SP)        ;;BRANCH IF NOT <CRLF>
(1)  016276  001006                         BNE     5$
(1)  016300  005726                         TST     (SP)+             ;;POP  <CR><LF> EQUIV
(1)  016302  104401                         TYPE                      ;;TYPE A CR AND LF
(1)  016304  001167                         $CRLF
(1)  016306  105037  016442                 CLRB    $CHARCNT          ;;CLEAR CHARACTER COUNT
(1)  016312  000755                         BR      2$                ;;GET NEXT CHARACTER
(1)  016314  004737  016376          5$:    JSR     PC,$TYPEC         ;;GO TYPE THIS CHARACTER
(1)  016320  123726  001156          6$:    CMPB    $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
(1)  016324  001350                         BNE     2$                ;;IF NO GO GET NEXT CHAR.
(1)  016326  013746  001154                 MOV     $NULL,-(SP)       ;;GET # OF FILLER CHARS. NEEDED
(1)                                                                    ;;AND THE NULL CHAR.
(1)  016332  105366  000001          7$:    DECB    1(SP)             ;;DOES A NULL NEED TO BE TYPED?
(1)  016336  002770                         BLT     6$                ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1)  016340  004737  016376                 JSR     PC,$TYPEC         ;;GO TYPE A NULL
(1)  016344  105337  016442                 DECB    $CHARCNT          ;;DO NOT COUNT AS A COUNT
(1)  016350  000770                         BR      7$                ;;LOOP
(1)
(1)                                  ;HORIZONTAL TAB PROCESSOR
(1)
(1)  016352  112716  000040          8$:    MOVB    #' ,(SP)          ;;REPLACE TAB WITH SPACE
(1)  016356  004737  016376          9$:    JSR     PC,$TYPEC         ;;TYPE A SPACE
(1)  016362  132737  000007  016442        BITB    #7,$CHARCNT       ;;BRANCH IF NOT AT
(1)  016370  001372                         BNE     9$                ;;TAB STOP
(1)  016372  005726                         TST     (SP)+             ;;POP SPACE OFF STACK
(1)  016374  000724                         BR      2$                ;;GET NEXT CHARACTER
(1)  016376  105777  162546          $TYPEC: TSTB    @$TPS             ;;WAIT UNTIL PRINTER IS READY
(1)  016402  100375                         BPL     $TYPEC
(1)  016404  116677  000002  162540        MOVB    2(SP),@$TPB       ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  016412  122766  000015  000002        CMPB    #CR,2(SP)         ;;IS CHARACTER A CARRIAGE RETURN?
(1)  016420  001003                         BNE     1$                ;;BRANCH IF NO
(1)  016422  105037  016442                 CLRB    $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
(1)  016426  000406                         BR      $TYPEX            ;;EXIT
(1)  016430  122766  000012  000002  1$:    CMPB    #LF,2(SP)         ;;IS CHARACTER A LINE FEED?
(1)  016436  001402                         BEQ     $TYPEX            ;;BRANCH IF YES
(1)  016440  105227                         INCB    (PC)+             ;;COUNT THE CHARACTER
(1)  016442  000000                  $CHARCNT:.WORD  0                 ;;CHARACTER COUNT STORAGE
(1)  016444  000207                  $TYPEX: RTS     PC
(1)
2428                                 .SBTTL  TTY INPUT ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)                                  .ENABL  LSB
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1)                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1)                                  ;*WHEN OPERATING IN TTY FLAG MODE.
(1)  016446  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR        ;;IS THE SOFT-SWR SELECTED?
(1)  016454  001074                         BNE     15$               ;;BRANCH IF NO
```

```
(1)  016456  105777  162462                  TSTB    @$TKS           ;;CHAR THERE?
(1)  016462  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
(1)  016464  117746  162456                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
(1)  016470  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
(1)  016474  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
(1)  016500  001062                          BNE     15$             ;;NO, RETURN TO USER
(1)  016502  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  016510  001456                          BEQ     15$             ;;BRANCH IF YES
(1)
(1)  016512  104401  017173                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  016516  104401  017200          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  016522  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  016526  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016530  104401  017211                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  016534  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
(1)  016536  005046                          CLR     -(SP)           ;;THE NEW SWR
(1)  016540  105777  162400          7$:     TSTB    @$TKS           ;;CHAR THERE?
(1)  016544  100375                          BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  016546  117746  162374                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  016552  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1)  016556  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  016562  001005                          BNE     10$             ;;BRANCH IF NOT
(1)  016564  104401  017166                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  016570  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  016574  000757                          BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  016576  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
(1)  016602  001022                          BNE     16$             ;;BRANCH IF NO
(1)  016604  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  016610  001403                          BEQ     11$             ;;BRANCH IF YES
(1)  016612  016677  000002  162320          MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  016620  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
(1)  016624  104401  001167          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  016630  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  016636  001003                          BNE     15$             ;;BRANCH IF NOT
(1)  016640  012777  000100  162276          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  016646  000002                  15$:    RTI                     ;;RETURN
(1)  016650  004737  016376          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  016654  021627  000060                  CMP     (SP),#'0        ;;CHAR < 0?
(1)  016660  002420                          BLT     18$             ;;BRANCH IF YES
(1)  016662  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
(1)  016666  003015                          BGT     18$             ;;BRANCH IF YES
(1)  016670  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  016674  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  016700  001403                          BEQ     17$             ;;BRANCH IF YES
(1)  016702  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  016704  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
(1)  016706  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
(1)  016710  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  016714  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  016720  000707                          BR      7$              ;;GET THE NEXT ONE
```

```
(1)  016722  104401  001166          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  016726  000720                          BR      20$             ;;SIMULATE CONTROL-U
(1)                                  .DSABL  LSB
(1)
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                  ;*CALL:
(1)                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                  ;
(1)
(1)  016730  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  016732  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  016740  105777  162200          1$:     TSTB    @$TKS           ;;WAIT FOR
(1)  016744  100375                          BPL     1$              ;;A CHARACTER
(1)  016746  117766  162174  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  016754  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  016762  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  016770  001013                          BNE     3$              ;;BRANCH IF NO
(1)  016772  105777  162146          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  016776  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  017000  117746  162142                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  017004  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  017010  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  017014  001366                          BNE     2$              ;;IF NOT DISCARD IT
(1)  017016  000750                          BR      1$              ;;YES, RESUME
(1)  017020  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  017026  002407                          BLT     4$              ;;BRANCH IF YES
(1)  017030  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  017036  003003                          BGT     4$              ;;BRANCH IF YES
(1)  017040  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
(1)  017046  000002                  4$:     RTI                     ;;GO BACK TO USER
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                  ;*CALL:
(1)                                  ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                  ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                  ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  017050  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  017052  012703  017156          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  017056  022703  017166          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
(1)  017062   01405                          BLOS    4$              ;;BR IF YES
(1)  017064  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  017066  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  017070  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  017074  001003                          BNE     3$              ;;SKIP IF NOT
(1)  017076  104401  001166          4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  017102  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  017104  111337  017154          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  017110  104401  017154                  TYPE    ,9$
(1)  017114  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  017120  001356                          BNE     2$              ;;LOOP IF NOT RETURN
(1)  017122  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
```

```
   (1)  017126  104401  001170                    TYPE    ,$LF            ;;TYPE A LINE FEED
   (1)  017132  012603                            MOV     (SP)+,R3        ;;RESTORE R3
   (1)  017134  011646                            MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
   (1)  017136  016666  000004  000002            MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
   (1)  017144  012766  017156  000004            MOV     #$TTYIN,4(SP)
   (1)  017152  000002                            RTI                     ;;RETURN
   (1)  017154     000                 9$:        .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
   (1)  017155     000                            .BYTE   0               ;;TERMINATOR
   (1)  017156  000010              $TTYIN:       .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
   (1)  017166  052536  005015  000  $CNTLU:       .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
   (1)  017173     136  006507  000012 $CNTLG:     .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
   (1)  017200  005015  053523  020122 $MSWR:     .ASCIZ  <15><12>/SWR = /
   (1)  017206  020075     000
   (1)  017211     040  047040  053505 $MNEW:     .ASCIZ  /  NEW = /
   (1)  017216  036440  000040
  2429                                            .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
   (1)
   (2)                                            ;;**************************************************************
   (1)                                            ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
   (1)                                            ;*CHANGE IT TO BINARY.
   (1)                                            ;*CALL:
   (1)                                            ;*      RDOCT                   ;;READ AN OCTAL NUMBER
   (1)                                            ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
   (1)                                            ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
   (1)
   (1)  017222  011646              $RDOCT:       MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
   (1)  017224  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
   (3)  017232  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
   (3)  017234  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
   (3)  017236  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
   (1)  017240  104411                 1$:        RDLIN                   ;;READ AN ASCIZ LINE
   (1)  017242  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
   (1)  017244  005001                            CLR     R1              ;;CLEAR DATA WORD
   (1)  017246  005002                            CLR     R2
   (1)  017250  112046                 2$:        MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
   (1)  017252  001412                            BEQ     3$              ;;IF ZERO GET OUT
   (1)  017254  006301                            ASL     R1              ;;*2
   (1)  017256  006102                            ROL     R2
   (1)  017260  006301                            ASL     R1              ;;*4
   (1)  017262  006102                            ROL     R2
   (1)  017264  006301                            ASL     R1              ;;*8
   (1)  017266  006102                            ROL     R2
   (1)  017270  042716  177770                    BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
   (1)  017274  062601                            ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
   (1)  017276  000764                            BR      2$              ;;LOOP
   (1)  017300  005726                 3$:        TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
   (1)  017302  010166  000012                    MOV     R1,12(SP)       ;;SAVE THE RESULT
   (1)  017306  010267  017322                    MOV     R2,$HIOCT
   (3)  017312  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
   (3)  017314  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
   (3)  017316  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
   (1)  017320  000002                            RTI                     ;;RETURN
   (1)  017322  000000              $HIOCT:       .WORD   0               ;;HIGH ORDER BITS GO HERE
  2430                                            .SBTTL  APT COMMUNICATIONS ROUTINE
   (1)
   (2)                                            ;;**************************************************************
```

```
(1)  017324  112737  000001  017570  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  017332  112737  000001  017566  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  017340  000403                          BR      $ATYC
(1)  017342  112737  000001  017570  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  017350                          $ATYC:
(3)  017350  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  017352  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  017354  105737  017566                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  017360  001450                          BEQ     5$              ;;IF NOT:  BR
(1)  017362  122737  000001  001212          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  017370  001031                          BNE     3$              ;;IF NOT:  BR
(1)  017372  132737  000100  001213          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  017400  001425                          BEQ     3$              ;;IF NOT:  BR
(1)  017402  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  017406  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  017414  005737  001172          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  017420  001375                          BNE     1$              ;;IF NOT:  WAIT
(1)  017422  010037  001206                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  017426  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  017430  001376                          BNE     2$
(1)  017432  163700  001206                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  017436  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  017440  010037  001210                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  017444  012737  000004  001172          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  017452  000413                          BR      5$
(1)  017454  017637  000004  017500  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  017462  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  017470  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  017474  004737  016164                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  017500  000000                  4$:     .WORD   0
(1)  017502                          5$:
(1)  017502  105737  017570          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  017506  001416                          BEQ     12$             ;;IF NOT:  BR
(1)  017510  005737  001212                  TST     $ENV            ;;RUNNING UNDER APT?
(1)  017514  001413                          BEQ     12$             ;;IF NOT:  BR
(1)  017516  005737  001172          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  017522  001375                          BNE     11$             ;;IF NOT:  WAIT
(1)  017524  017637  000004  001174          MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  017532  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  017540  005237  001172                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  017544  105037  017570          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  017550  105037  017567                  CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  017554  105037  017566                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  017560  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  017562  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  017564  000207                          RTS     PC              ;;RETURN
(1)  017566     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  017567     000          $LFLG:  .BYTE   0               ;;LOG FLAG
(1)  017570     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          017572                  .EVEN
(1)          000200          APTSIZE=200
(1)          000001          APTENV=001
(1)          000100          APTSPOOL=100
(1)          000040          APTCSUP=040
2431
2432                         .SBTTL  TRAP DECODER
```

```
(1)
(2)                                     ;;**************************************** ****************************
(1)                                     ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)                                     ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)                                     ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)                                     ;*GO TO THAT ROUTINE.
(1)
(1)  017572  010046                     $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
(1)  017574  016600  000002                     MOV     2(SP),R0        ;;GET TRAP ADDRESS
(1)  017600  005740                             TST     -(R0)           ;;BACKUP BY 2
(1)  017602  111000                             MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
(1)  017604  006300                             ASL     R0              ;;POSITION FOR INDEXING
(1)  017606  016000  017626                     MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
(1)  017612  000200                             RTS     R0              ;;GO TO ROUTINE
(1)
(1)
(1)                                     ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1)  017614  011646                     $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
(1)  017616  016666  000004  000002             MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
(1)  017624  000002                             RTI                     ;;RESTORE THE PSW
(1)
(3)                                     .SBTTL  TRAP TABLE
(3)
(3)                                     ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)                                     ;*BY THE "TRAP" INSTRUCTION.
(3)
(3)                                     ;       ROUTINE
(3)                                     ;       -------
(3)  017626  017614                     $TRPAD: .WORD   $TRAP2
(3)  017630  016164                             $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3)  017632  015762                             $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)  017634  015736                             $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)  017636  015776                             $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)  017640  014654                             $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
(1)
(3)  017642  016516                             $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
(1)
(3)  017644  016446                             $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
(3)  017646  016730                             $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3)  017650  017050                             $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3)  017652  017222                             $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
2433
2434  017654  000074                    BUF2:   .BLKW   60.
2435  020044  000074                    BUF3:   .BLKW   60.
2436  020234  000074                    BUFFER: 60.
2437
2438  020236  000240                            NOP
2439
 (2)                                            ;*
 (2)                                            ;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
 (2)                                            ;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
 (2)                                            ;*NEXT WE WILL INIT BOTH UPROCESSORS
 (2)                                            ;*THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
 (2)                                            ;*THE ORDER OF LOAD IS DETERMINED BY THE USER.
 (2)                                            ;*
```

```
(2)                                         ;*      CALL=   JSR     R5,$LPAI
(2)                                         ;*              .WORD   0               ;ADDR. OF DEVICE ADDRESS.
(2)                                         ;* ROUTINES REQUIRED:   .LOADLP
(2)                                         ;* PROGRAMS REQUIRED:   DRLPX2
(2)                                         ;*
(2)
(2)                                         ;*              ;RETURNS WITH $AERR=1 IF SLAVE
(2)                                         ;*              ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
(2)                                         ;*
(2)  020240                         $LPAI:
(2)  020240  013746  000004                 MOV     4,-(SP)
(2)
(2)  020244  000413                 BR      31$             ;FIELD DOES NOT HAVE A BUS SWITCH TO
(2)                                                         ;WORRY ABOUT,SO WE WILL UNCONDITIONALLY
(2)                                                         ;BRANCH ARROUD THE NEXT CODE THAT
(2)                                                         ;WORKS BASED ON A BUS SWITCH.
(2)                                                         ;CODE LEFT IN HERE FOR IN HOUSE
(2)                                                         ;PERSONAL WHO MAY PATCH THIS BRANCH
(2)                                                         ;INSTRUCTION TO A <NOP> OCTAL <240>
(2)                                                         ;IN ORDER TO RUN PROGRAM WITH A SWITCH.
(2)
(2)                                                         ;NOTE THIS ''SWITCH'' IS A PIECE OF INHOUSE
(2)                                                         ;TEST EQUIPMENT ONLY IT CONNECTS
(2)                                                         ;THE UNIBUS TO THE I/O BUS FOR
(2)                                                         ;CERTAIN TESTING.
(2)  020246  012737  020272  000004         MOV     #30$,4
(2)  020254  005237  170000                 INC     170000
(3)  020260  104401  020266                 TYPE    ,65$            ;;TYPE ASCIZ STRING
(3)  020264  000401                 BR      64$             ;;GET OVER THE ASCIZ
(3)                                 ;;65$:  .ASCIZ  <7>##
(3)  020270                         64$:
(2)  020270  000401                 BR      31$
(2)  020272  022626                 30$:    CMP     (SP)+,(SP)+
(2)  020274  012637  000004         31$:    MOV     (SP)+,4         ;ALL THIS JUNK MUST BE REMOVED!!
(2)  020300  005037  020726                 CLR     $AERR
(2)  020304  004537  020730                 JSR     R5,$LOAD        ;LOAD MICRO-CODE.
(2)  020310  000000G                        .WORD   DRLPX2          ;FILE ''DRLPX2.OBJ''
(2)
(2)  020312  052777  040000  161044         BIS     #BIT14,@KMADO   ;ISSUE KMC+DMC INIT.
(2)
(2)  020320                         1$:
(2)                                                         ;''HANGS'' HERE THEN KMC-11 ERROR.
(2)  020320  010146                 MOV     R1,-(SP)
(2)  020322  005001                 CLR     R1
(2)  020324  005201                 2$:     INC     R1              ;STALL FOR DMC-UP
(2)  020326  001376                 BNE     2$
(2)  020330  012777  104000  161026         MOV     #BIT15!BIT11,@KMADO     ;SET RUN, AND ENABLE ARBITRATION.
(2)  020336  105201                 25$:    INCB    R1
(2)  020340  001376                 BNE     25$
(2)
(2)  020342  032777  000040  161014         BIT     #BIT5,@KMAD0    ;SLAVE READY? (READING IPBM SR)
(2)  020350  001401                 BEQ     3$
(2)                                                         ;FATAL LPA-11 ERROR SLAVE NOT READY.
(2)  020352  104000                 ERROR
(2)
(2)  020354  012777  000004  161006  3$:    MOV     #4,@KMAD2       ;READ FAST PATH
```

```
(2)  020362                          4$:
(3)  020362  004537  021640                 JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  020366  104000                         ERROR                   ;/TIME-OUT ERROR
(3)                                                                 ;/WE FAILED TO COMPLETE
(3)                                                                 ;/CURRENT OPERATION.
(3)                                                                 ;/CONTINUES IN THIS LOOP
(3)                                                                 ;/WOULD MAKE US "HANG" HERE
(3)
(3)  020370  000774                         BR              4$
(3)
(3)                                                                 ;/RETURNS HERE-FROM-TIMED OUT.
(2)  020372  122777  000377  160770         CMPB    #377,@KMAD2     ;WAIT TILL KMC DONE COMMAND.
(2)  020400  001370                         BNE     4$
(2)  020402  122777  000377  160764         CMPB    #377,@KMAD4     ;IF FAST PATH=377 THEN ERROR.
(2)  020410  001001                         BNE     35$
(2)  020412  104000                         ERROR                   ;IPBM ERROR (SLAVE SIDE)
(2)                                                                 ;YOU MUST RUN IPBM DIAGNOSTIC.
(2)
(2)  020414  117737  160754  020674  35$:  MOVB    @KMAD4,11$      ;GET THE VERSION NUMBER FROM DMC-11
(2)  020422  005227  177777               INC     #-1
(2)  020426  001045                         BNE     5$
(2)  020430  005227  177777               INC     #-1
(2)  020434  001042                         BNE     5$
(3)  020436  104401  020444                 TYPE    ,67$            ;;TYPE ASCIZ STRING
(3)  020442  000426                         BR      66$             ;;GET OVER THE ASCIZ
(3)                                ;;67$:   .ASCIZ  <200>"M8200-YC (DMC) MICROCODE VERSION NUMBER - "
(3)  020520                          66$:
(2)  020520  013746  020674                 MOV     11$,-(SP)
(2)  020524  104403                         TYPOS
(2)  020526     002     000                 .BYTE   2,0
(3)  020530  104401  020536                 TYPE    ,69$            ;;TYPE ASCIZ STRING
(3)  020534  000402                         BR      68$             ;;GET OVER THE ASCIZ
(3)                                ;;69$:   .ASCIZ  <200>"  "
(3)  020542                          68$:
(2)
(2)  020542  112737  177777  020674  5$:   MOVB    #0-1,11$        ;DAC CODE FOR SLAVE.
(2)  020550  012501                         MOV     (5)+,R1         ;GET NEXT DEVICE ADDR.
(2)  020552  021127  000000          6$:    CMP     (R1),#0         ;TERM REACHED?
(2)  020556  001444                         BEQ     10$
(2)  020560  105237  020674                 INCB    11$
(2)  020564  113777  020674  160602         MOVB    11$,@KMAD4      ;FIFO DATA
(2)  020572  004737  020676                 JSR     PC,20$          ;ISSUE SEND
(2)  020576  112177  160572                 MOVB    (R1)+,@KMAD4    ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
(2)  020602  004737  020676                 JSR     PC,20$          ;ISSUE SEND
(2)  020606  112177  160562                 MOVB    (R1)+,@KMAD4    ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
(2)  020612  004737  020676                 JSR     PC,20$
(2)
(2)  020616  032777  000002  160540  7$:   BIT     #BIT1,@KMAD0    ;WAIT FOR FIFO DATA
(2)  020624  001374                         BNE     7$              ;=1 NO DATA. =0 DATA.
(2)  020626  112777  000002  160534         MOVB    #2,@KMAD2       ;READ FIFO.
(2)
(2)  020634                          8$:
(3)  020634  004537  021640                 JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  020640  104000                         ERROR                   ;/TIME-OUT ERROR
```

```
(3)                                                     ;/WE FAILED TO COMPLETE
(3)                                                     ;/CURRENT OPERATION.
(3)                                                     ;/CONTINUES IN THIS LOOP
(3)                                                     ;/WOULD MAKE US "HANG" HERE
(3)
(3)  020642  000774                     BR              8$
(3)
(3)                                                     ;/RETURNS HERE-FROM-TIMED OUT.
(2)  020644  122777  000377  160516     CMPB    #377,@KMAD2     ;WAIT FOR READ.
(2)  020652  001370                     BNE     8$
(2)  020654  105777  160514             TSTB    @KMAD4          ;WAS A ZERO RETURNED?
(2)  020660  001734                     BEQ     6$              ;YES GET NEXT ADDR.
(2)                                                     ;SLAVE WILL RETURN CODE 0 IF
(2)  020662  005237  020726             INC     $AERR           ;DEV PRESENT.   ELSE
(2)                                                     ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
(2)  020666  005041                     CLR     -(1)            ;GET RID OF REFERENCE TO BAD ADDR.
(2)  020670  012601             10$:    MOV     (SP)+,R1
(2)  020672  000205                     RTS     R5              ;RETURN ALL ADDR. CHECKED.
(2)
(2)  020674  000000             11$:    .WORD   0               ;HOLDS DAC CODE PLUS OFFSET
(2)                                                     ;TO SLAVES ADDR. TABLE.
(2)
(2)  020676  112777  000003  160464 20$:    MOVB    #3,@KMAD2       ;ISSUE FIFO WRITE
(2)  020704                      21$:
(3)  020704  004537  021640             JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  020710  104000                     ERROR                   ;/TIME-OUT ERROR
(3)                                                     ;/WE FAILED TO COMPLETE
(3)                                                     ;/CURRENT OPERATION.
(3)                                                     ;/CONTINUES IN THIS LOOP
(3)                                                     ;/WOULD MAKE US "HANG" HERE
(3)
(3)  020712  000774                     BR              21$
(3)
(3)                                                     ;/RETURNS HERE-FROM-TIMED OUT.
(2)  020714  122777  000377  160446     CMPB    #377,@KMAD2     ;KMC CODE WILL RETURN A "377"
(2)  020722  001370                     BNE     2'$             ;WHEN DONE COMMAND.
(2)  020724  000207                     RTS     PC
(2)
(2)  020726  000000             $AERR:  .WORD   0               ;=0 IF ADDR. LIST OK,-1 IF BAD.
(2)
(2)                                             ;*
(2)                                             ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
(2)                                             ;*      CALL =  JSR     R5,$LOAD
(2)                                             ;*              .WORD   XX              ;ADDR. OF MICRO CODE.
(2)                                             ;*              ;RETURNS HERE
(2)                                             ;*      NOTE:   MICRO CODE FILE MUST END IN -1 DATA.
(2)                                             ;*
(2)
(2)  020730  010446             $LOAD:  MOV     R4,-(SP)        ;SAVE R4.
(2)  020732  010046                     MOV     R0,-(SP)        ;SAVE R0.
(2)  020734  012500             1$:     MOV     (5)+,R0         ;GET PROG. ADDR.
(2)  020736  005077  160422             CLR     @KMAD0          ;CLEAR CSR
(2)  020742  005077  160426             CLR     @KMAD4          ;CLEAR CRAM ADDR.
(2)  020746  052777  002000  160410 2$:     BIS     #2000,@KMAD0    ;SELECT CRAM.
(2)  020754  012077  160420             MOV     (0)+,@KMAD6     ;WRITE DATA.
```

```
(2)  020760  052777  020000  160376          BIS     #20000,@KMAD0   ;SET CRAM WRITE
(2)  020766  005077  160372                  CLR     @KMAD0          ;DISABLE CRAM.
(2)  020772  005277  160376                  INC     @KMAD4          ;UPDATE CRAM ADDR.
(2)  020776  021027  177777                  CMP     (0),#-1         ;ALL DONE?
(2)  021002  001361                          BNE     2$              ;NO LOOP.
(2)  021004  005077  160364                  CLR     @KMAD4          ;CLEAR CRAM ADDR.
(2)  021010  016500  177776                  MOV     -2(5),R0        ;GET MICRO CODE ADDR.
(2)
(2)  021014  052777  002000  160342  3$:     BIS     #2000,@KMAD0    ;SELECT CRAM
(2)  021022  022077  160352                  CMP     (R0)+,@KMAD6    ;DATA OK?
(2)  021026  001013                          BNE     5$              ;NO - REPORT AN ERROR.
(2)  021030  021027  177777                  CMP     (0),#-1         ;ALL DONE?
(2)  021034  001405                          BEQ     4$              ;YES - EXIT
(2)  021036  005077  160322                  CLR     @KMAD0          ;NO - DESELECT CRAM.
(2)  021042  005277  160326                  INC     @KMAD4          ;UPDATE CRAM ADDR.
(2)  021046  000762                          BR      3$
(2)
(2)  021050  012600                  4$:     MOV     (SP)+,R0        ;RESTORE R0
(2)  021052  012604                          MOV     (SP)+,R4        ;RESTORE R4
(2)  021054  000205                          RTS     R5              ;EXIT
(2)
(2)  021056                          5$:                             ;COME HERE ON LOAD ERROR
(2)  021056  005745                          TST     -(5)
(2)  021060  105204                          INCB    R4              ;UPDATE ERROR COUNTER.
(2)  021062  100324                          BPL     1$              ;IF NOT TOO MANY, TRY AGAIN.
(2)  021064  000000                          HALT                    ;MICRO CODE LOAD ERROR.
(2)                                                                  ;KMC-11 FAULT. YOU COULD TRY
(2)  021066  000722                  1$      BR      1$              ;TO PRESS CONTINUE TO GIVE IT
(2)                                                                  ;ANOTHER CHANCE, BUT I DOUBT
(2)                                                                  ;THAT THAT WOULD WORK. SINCE I'VE
(2)                                                                  ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
(2)                                                                  ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
(2)
(2)
(2)                                                  ;
(2)                                                  ;*THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
(2)                                                  ;*
(2)                                                  ;*      CALL =  JSR     R5,$TLKW
(2)                                                  ;*              .WORD   0               ;OFFSET OF DEVICE ADDR.
(2)                                                  ;*              .WORD   0               ;DATA TO BE WRITTEN
(2)                                                  ;*
(2)  021070  010046                  $TLKW:  MOV     R0,-(SP)        ;SAVE R0
(2)  021072  012500                          MOV     (5)+,R0         ;GET DEVICE OFFSET
(2)  021074  052700  000340                  BIS     #340,R0         ;ADD WRITE CODE.
(2)  021100  004737  021352                  JSR     PC,$LPW         ;WAIT FOR FAST PATH READY
(2)  021104  010037  021176                  MOV     R0,@#1
(2)  021110  010077  160260                  MOV     R0,@KMAD4
(2)  021114  112777  000005  160246          MOVB    #5,@KMAD2       ;ISSUE FAST PATH WRITE
(2)  021122  004737  021352                  JSR     PC,$LPW         ;WAIT FOR RDY
(2)  021126  011537  021200                  MOV     (5),@#2
(2)  021132  112577  160236                  MOVB    (5)+,@KMAD4     ;WRITE LOW BYTE DATA.
(2)
(2)  021136  112777  000005  160224          MOVB    #5,@KMAD2       ;FP WRITE
(2)  021144  004737  021352                  JSR     PC,$LPW
(2)  021150  111537  021202                  MOVB    (5),@#3
(2)  021154  112577  160214                  MOVB    (5)+,@KMAD4     ;WRITE HIGH BYTE
```

```
(2)  021160  112777  000005  160202          MOVB    #5,@KMAD2
(2)  021166  004737  021352                  JSR     PC,$LPW
(2)  021172  012600                          MOV     (SP)+,R0
(2)  021174  000205                          RTS     R5              ;EXIT DONE.
(2)  021176  000000                  W1:     0
(2)  021200  000000                  W2:     0
(2)  021202  000000                  W3:     0
(2)
(2)                                          ;*
(2)                                          ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
(2)                                          ;*
(2)                                          ;*      CALL =  JSR     R5,$TLKR
(2)                                          ;*              .WORD   0               ;OFFSET OF DEVICE
(2)                                          ;*              ;RETURNS HERE
(2)                                          ;*DATA IN WORD $DATR
(2)                                          ;*
(2)
(2)  021204  010046                  $TLKR:  MOV     R0,-(SP)        ;SAVE R0
(2)  021206  012500                          MOV     (5)+,R0         ;GET OFFSET
(2)  021210  052700  000300                  BIS     #300,R0         ;ADD READ CODE
(2)  021214  004737  021352                  JSR     PC,$LPW         ;WAIT TILL READY
(2)  021220  110077  160150                  MOVB    R0,@KMAD4
(2)  021224  112777  000005  160136          MOVB    #5,@KMAD2       ;ISSUE WRITE FP
(2)  021232  004737  021352                  JSR     PC,$LPW
(2)  021236  010037  021346                  MOV     R0,RD1
(2)  021242                          1$:
(3)  021242  004537  021640                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  021246  104000                          ERROR                   ;/TIME-OUT ERROR
(3)                                          ;/WE FAILED TO COMPLETE
(3)                                          ;/CURRENT OPERATION.
(3)                                          ;/CONTINUES IN THIS LOOP
(3)                                          ;/WOULD MAKE US "HANG" HERE
(3)
(3)  021250  000774                          BR              1$
(3)
(3)                                          ;/RETURNS HERE-FROM-TIMED OUT.
(2)  021252  032777  000040  160104          BIT     #BIT5,@KMAD0    ;FAST PATH GOT DATA?
(2)  021260  001370                          BNE     1$
(2)  021262  112777  000004  160100          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
(2)  021270  004737  021352                  JSR     PC,$LPW
(2)  021274  117737  160074  021350          MOVB    @KMAD4,$DATR    ;GET LOW BYTE
(2)  021302                          2$:
(3)  021302  004537  021640                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  021306  104000                          ERROR                   ;/TIME-OUT ERROR
(3)                                          ;/WE FAILED TO COMPLETE
(3)                                          ;/CURRENT OPERATION.
(3)                                          ;/CONTINUES IN THIS LOOP
(3)                                          ;/WOULD MAKE US "HANG" HERE
(3)
(3)  021310  000774                          BR              2$
(3)
(3)                                          ;/RETURNS HERE-FROM-TIMED OUT.
(2)  021312  032777  000040  160044          BIT     #BIT5,@KMAD0    ;FAST PATH READY?
(2)  021320  001370                          BNE     2$
```

```
(2)  021322  112777  000004  160040          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
(2)  021330  004737  021352                  JSR     PC,$LPW
(2)  021334  117737  160034  021351          MOVB    @KMAD4,$DATR+1  ;SAVE HIGH BYTE
(2)  021342  012600                          MOV     (SP)+,R0
(2)  021344  000205                          RTS     R5
(2)  021346  000000                  RD1:    0
(2)  021350  000000                  $DATR:  .WORD   0
(2)                                          ;
(2)                                          ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
(2)                                          ;AS FAST PATH TO BE READ.
(2)                                          ;
(2)                                          ;       CALL =  JSR     PC,$LPW
(2)                                          ;
(2)                                          ;IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
(2)                                          ;THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
(2)                                          ;
(2)
(2)  021352  010146                  $LPW:   MOV     R1,-(SP)        ;SAVE R1
(2)  021354  005001                          CLR     R1
(2)  021356  122777  000377  160004  1$:     CMPB    #377,@KMAD2     ;FINISHED INSTRUCTION?
(2)  021364  001403                          BEQ     2$
(2)  021366  005201                          INC     R1              ;TIME OUT?
(2)  021370  001372                          BNE     1$
(2)  021372  000411                          BR      10$
(2)
(2)  021374  032777  000020  157762  2$:     BIT     #BIT4,@KMAD0    ;FAST PATH READ?
(2)  021402  001403                          BEQ     3$
(2)  021404  005201                          INC     R1              ;NO - TIME OUT?
(2)  021406  001372                          BNE     2$
(2)  021410  000402                          BR      10$             ;YES - REPORT AN ERROR
(2)
(2)  021412  012601                  3$:     MOV     (SP)+,R1        ;RESTORE R1
(2)  021414  000207                          RTS     PC              ;EXIT
(2)
(2)  021416                          10$:
(3)  021416  104401  021424                  TYPE    ,65$            ;;TYPE ASCIZ STRING
(3)  021422  000407                          BR      64$             ;;GET OVER THE ASCIZ
(3)                                  ;;65$:  .ASCIZ  <200>#LPA-11 FAULT#
(3)  021442                          64$:
(2)
(2)  021442  000000                  11$:    HALT                    ;LPA-11 FAULT RUN LPA-11
(2)  021444  000776                          BR      11$             ;DIAGNOSTICS.
(2)
(2)
(2)
(2)                                          ;*
(2)                                          ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
(2)                                          ;*A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.
(2)                                          ;*
(2)                                          ;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED
(2)                                          ;* BEFORE. IF NOT WE HAVE TO INITIALIZE THE LPA WITH
(2)                                          ;* THAT ADDRESS.
(2)                                          ;* WHEN THE ADDR. IS KNOWN BY THE LPA, DO THE OUTPUT BY
(2)                                          ;* $TLKW
(2)                                          ;*
(2)
```

```
(2)  021446  010046              $OUTLP: MOV     R0,-(SP)        ;SAVE R0
(2)  021450  010146                      MOV     R1,-(SP)        ;SAVE R1
(2)
(2)  021452  012700  001412              MOV     #.DVLS,R0       ;PROGRAM DEFINED LIST.
(2)  021456  005001                      CLR     R1
(2)  021460  005710              1$:     TST     (0)             ;TERMINATOR REACHED?
(2)  021462  001421                      BEQ     10$             ;YES NEXT STEP.
(2)  021464  027520  000000              CMP     @(5),(0)+       ;MATCH WITH ADDR IN LIST?
(2)  021470  001402                      BEQ     2$
(2)  021472  005201                      INC     R1
(2)  021474  000771                      BR      1$
(2)
(2)  021476  010137  021514      2$:     MOV     R1,3$           ;SAVE OFFSET, DEVICE KNOWN.
(2)  021502  005725                      TST     (5)+
(2)  021504  013537  021516              MOV     @(5)+,4$        ;GET DATA TO BE WRITTEN
(2)  021510  004537  021070              JSR     R5,$TLKW        ;DO WRITE
(2)  021514  000000              3$:     .WORD   0               ;DEVICE OFFSET
(2)  021516  000000              4$:     .WORD   0               ;DATA TO BE WRITTEN.
(2)  021520  012601                      MOV     (SP)+,R1
(2)  021522  012600                      MOV     (SP)+,R0
(2)  021524  000205                      RTS     R5
(2)  021526  017520  000000      10$:    MOV     @(5),(0)+       ;SAVE ADDR.
(2)  021532  005010                      CLR     (0)
(2)  021534  004537  020240              JSR     R5,$LPAI
(2)  021540  001412                      .WORD   .DVLS
(2)  021542  000755                      BR      2$
(2)
(2)                                      ;*
(2)                                      ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE
(2)                                      ;*TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
(2)                                      ;*
(2)                                      ;*FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN
(2)                                      ;*USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
(2)                                      ;*WITH THE NEW ADDR.
(2)                                      ;*WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH
(2)                                      ;*$TLKR
(2)                                      ;*      CALL THROUGH    MOVEI   DATA,ADDR.
(2)                                      ;*              WHICH EQUALS:
(2)                                      ;*                      JSR     R5,$INLP
(2)                                      ;*                      .WORD   XX      ADDR OF DEVICE
(2)                                      ;*                      .WORD   YY      ADDR TO STORE READ DATA.
(2)
(2)  021544  010046              $INLP:  MOV     R0,-(SP)        ;SAVE R0
(2)  021546  010146                      MOV     R1,-(SP)        ;SAVE R1
(2)
(2)  021550  012700  001412              MOV     #.DVLS,R0       ;PROG DEFINED ADDR. LIST.
(2)  021554  005001                      CLR     R1
(2)  021556  005710              1$:     TST     (0)             ;EOL REACHED?
(2)  021560  001420                      BEQ     10$             ;YES - DEFINE NEW ADDR.
(2)
(2)  021562  027520  000000              CMP     @(5),(0)+       ;ADDR. MATCH?
(2)  021566  001402                      BEQ     2$
(2)  021570  005201                      INC     R1
(2)  021572  000771                      BR      1$
(2)
(2)  021574  010137  021606      2$:     MOV     R1,3$           ;SAVE LIST OFFSET
```

```
(2)  021600  005725                        TST     (5)+
(2)  021602  004537  021204                JSR     R5,$TLKR        ;GO READ DEVICE
(2)          021606                $OFS=.
(2)  021606  000000                3$:     .WORD   0               ;OFFSET OF DEVICE
(2)
(2)  021610  013735  021350                MOV     $DATR,@(5)+     ;STORE DATA.
(2)  021614  012601                        MOV     (SP)+,R1        ;RESTORE R1
(2)  021616  012600                        MOV     (SP)+,R0        ;RESTORE R2
(2)  021620  000205                        RTS     R5              ;EXIT
(2)
(2)  021622  017520  000000        10$:    MOV     @(5),(0)+
(2)  021626  005010                        CLR     (0)
(2)  021630  004537  020240                JSR     R5,$LPAI
(2)  021634  001412                        .WORD   .DVLS
(2)  021636  000756                        BR      2$
(2)                                         ;*
(2)                                         ;*$TOUT ROUTINE USED TO WATCH IF
(2)                                         ;*      WE'RE IN A LOOP TOO-LONG
(2)                                         ;*      CALL=   JSR R5, $TOUT
(2)                                         ;*              ERROR X     ;RETURNS HERE ON TIMEOUT
(2)                                         ;*              BR
(2)                                         ;*              ;RETURNS HERE NO ERROR
(2)                                         ;*
(2)
(2)  021640  020537  021674        $TOUT:  CMP     R5,$SAD         ;SAME ADDR?
(2)  021644  001405                        BEQ     1$
(2)  021646  010537  021674                MOV     R5,$SAD         ;NO-SAVE THIS ADDR.
(2)  021652  005037  021676                CLR     $CNT            ;CLR CNT AT ADDR.
(2)  021656  000403                        BR      2$
(2)  021660  005237  021676        1$:     INC     $CNT            ;OVERFLOW?
(2)  021664  100402                        BMI     3$              ;YES-ERROR RETURN
(2)  021666  062705  000004        2$:     ADD     #4,R5           ;NO-NON ERROR RETURN
(2)  021672  000205                3$:     RTS     R5              ;RETURN.
(2)
(2)  021674  000000                $SAD:   .WORD   0               ;CONTAINS LOOP ADDR.
(2)  021676  000000                $CNT:   .WORD   0               ;# OF TIMES AT ADDR.
(2)
(2)                                         ;*
(2)                                         ;* THIS ROUTINE REPLACES WHAT THE USER WOULD ORDINARILY
(2)                                         ;*USE FOR A RESET.  FIRST,WE DO A RESET INSTRUCTION.
(2)                                         ;*THEN WE CLR ".DVLST" WHICH FORCES US TO RESET BOTH THE
(2)                                         ;*KMC AND DMC AS SOON AS A DEVICE IS REFERENCED.
(2)                                         ;*
(2)                                         ;*      CALL=JSR        PC,$RESET       ;REPLACES "RESET INSTRUCTION
(2)                                         ;*                  ;RETURNS HERE.
(2)                                         ;*
(2)  021700  000005                $RESET: RESET                   ;RESET THE WORLD.
(3)
(3)                                 ;*      MOV     @2$,1$  ;/READ DEVICE REG 2$,PUT DATA IN 1$.
(2)  021712  005737  020726                TST     $AERR           ;IF NO ERROR,LOOP
(2)  021716  001004                        BNE     10$             ;THERE WAS AN ERROR.
(2)  021720  062737  000002  021734        ADD     #2,2$           ;UPDATE DEVICE ADDR.
(2)                                                                 ;YOU SEE ,WE HAVE TO PROTECT OUR SELF!
(2)                                                                 ;IF 2$ CONTAINED A VALID ADDR,WE
(2)                                                                 ;MUST KEEP TRYING UNTIL WE GENERATE
(2)                                                                 ;AN INVALID ADDR.
```

```
(2)  021726  000764                        BR      $RESET
(2)  021730                          10$:
(2)  021730  000207                        RTS     PC
(2)  021732  000000                  1$:   .WORD   0               ;JUNK LOC.
(2)  021734  160000                  2$:   .WORD   160000          ;DUMB ADDR. FORCES INIT OF DMC/KMC.
(2)
(2)
(2)                                         ;
(2)                                         ;SDELAY- ROUTINE TO GIVE A MINOR DELAY.
(2)                                         ;        IS NOT TIME DEPENDENT CODE SENCE
(2)                                         ;        NOT USED TO GET SPECIFIC TIME BUT
(2)                                         ;        JUST A LITTLE DELAY.
(2)                                         ;
(2)                                         ;       THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
(2)                                         ;       THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
(2)                                         ;
(2)                                         ;
(2)                                         ;       CALL=   JSR PC,  SDELAY
(2)                                         ;
(2)  021736                          SDELAY:
(2)  021736  005737  022020                 TST     RTCCSR          ;CLOCK PRESENT?
(2)  021742  100016                         BPL     10$
(2)  021744  012737  000002  022010         MOV     #2,TIME
(2)  021752  052777  000115  000040         BIS     #115,@RTCCSR    ;START CLOCK
(2)  021760  005037  177776                 CLR     PS
(2)  021764  005737  022010          1$:    TST     TIME
(2)  021770  001375                         BNE     1$
(2)  021772  005077  000022                 CLR     @RTCCSR         ;STOP CLOCK
(2)
(2)  021776  000207                         RTS PC
(2)  022000  105237  022010          10$:   INCB    TIME
(2)  022004  001375                         BNE     10$
(2)  022006  000207                         RTS     PC
(2)
(2)  022010  000000                  TIME:  .WORD   0
(2)
(2)  022012  005337  022010          CLKINT: DEC    TIME
(2)  022016  000002                         RTI
(2)  022020  000000                  RTCCSR: .WORD  0               ;CLOCK CSR IF USED.
(2)                                         ;
(2)                                         ;*
(2)                                         ;*THIS MACRO ALLOWS THE OPERATOR TO TALK TO
(2)                                         ;*ANY DEVICE ON THE I/O BUS
(2)                                         ;*USER MUST START AT THIS ADDR.
(2)                                         ;*HE MUST SAY EITHER "E" FOR EXAMINE, OR "D" FOR DEPOSIT.
(2)                                         ;*"E" IS DEFAULT.
(2)                                         ;*NEXT, HE MUST SUPPLY AN ADDR.
(2)                                         ;*NOTE IF ADDR. IS NOT FOUND ON I/O BUS, A HALT
(2)                                         ;*WILL OCCUR.
(2)
(2)  022022                          $UTK:
(2)  022022  005037  001412                 CLR     .DVLS
(2)  022026                          21$:
(3)  022026  104401  022034                 TYPE    ,65$            ;;TYPE ASCIZ STRING
(3)  022032  000405                         BR      64$             ;;GET OVER THE ASCIZ
(3)                                  ;;65$: .ASCIZ  <200>#E OR D?#
```

```
(3)  022046                          64$:
(2)  022046  105777  157072          1$:     TSTB    @$TKS
(2)  022052  100375                          BPL     1$
(2)  022054  117737  157066  022176          MOVB    @$TKB,20$       ;GET INPUT
(2)  022062  104401  022176                  TYPE,   20$             ;ECHO, NEXT MESSAGE.
(2)  022066  142737  000240  022176          BICB    #240,20$        ;STRIP PARITY, LC
(2)  022074  104412                          RDOCT                   ;GET ADDR.
(2)  022076  012637  022174                  MOV     (SP)+,14$
(2)  022102  123727  022176  000104          CMPB    20$,#'D         ;DEPOSIT?
(2)  022110  001411                          BEQ     10$
(2)
(2)  022112  004537  021544                  JSR     R5,$INLP        ;GET DATA
(2)  022116  022174                  2$:     .WORD   14$
(2)  022120  022132                          .WORD   5$
(2)
(3)  022122  013746  022132                  MOV     5$,-(SP)        ;;SAVE 5$ FOR TYPEOUT
(3)  022126  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(2)  022130  000736                          BR      21$             ;LOOP.
(2)  022132  000000                  5$:     .WORD   0
(2)
(2)  022134                          10$:
(3)  022134  104401  022142                  TYPE    ,67$            ;;TYPE ASCIZ STRING
(3)  022140  000404                          BR      66$             ;;GET OVER THE ASCIZ
(3)                                  ;;67$:  .ASCIZ  <200>#DATA= #
(3)  022152                          66$:
(2)  022152  104412                          RDOCT
(2)  022154  012637  022172                  MOV     (SP)+,13$
(2)
(2)  022160  004537  021446          11$:    JSR     R5,$OUTLP       ;OUTPUT ROUTINE.
(2)  022164  022174                  12$:    .WORD   14$             ;DEVICE ADDR.
(2)  022166  022172                          .WORD   13$             ;DATA
(2)  022170  000716                          BR      21$
(2)
(2)  022172  000000                  13$:    .WORD   0
(2)  022174  000000                  14$:    .WORD   0
(2)  022176  10^001  042504  044526  20$:    .ASCIZ  <1><200>#DEVICE ADDR= #
(2)  022204  042503  040440  042104
(2)  022212  036522  000040
(2)                                          .EVEN
(2)
(1)
(1)
(2)                                  ;
(2)                                  ;THIS ROUTINE LOOKS THROUGH CURENT .DVLS FOR A/D ADDR.
(2)                                  ;IF UNFOUND,GENERATES IT. THIS ROUTINE'S WHOLE PURPOSE IS
(2)                                  ;TO SET UP THE USER PROGRAM TO LINK TO FILE "DRLPX2" FOR
(2)                                  ;SAMPLE TAKEING PURPOSES.
(2)                                  ;       TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
(2)                                  ;       A/D CSR IN BSEL 4,AND 5.
(2)                                  ;       (2) HE MUST CALL THIS ROUTINE:
(2)                                  ;               JSR     R5,$PUTS        ;CALL SET UP ROUTINE.
(2)                                  ;               .WORD   ADCSR           ;ADDR. OF A/D CSR.
(2)                                  ;               ;RETURNS HERE ;KMC BSEL 3,6,7 PERMINENTLY SET UP
(2,                                  ;                               ;(UNTILL ONE DOES A RESET)
(2)                                  ;
(2)                                  ;       (3)THE USER MUST PUT CODE 006 INTO KMC REG 2 TO
```

```
   (2)                                          ;          START CONVERSION  CAUTION•DO WITH MOVB INSTR.!
   (2)                                          ;        (4)MONITOR KMC REG 2 FOR CODE 377 (DRLPX2 IS DONE)
   (2)                                          ;        (5)READ KMC REG 4,5 FOR A/D RESULT.
   (2)                                          ;        (6) TO TAKE MORE SAMPLES,SIMPLY PUT A/D CSR INTO
   (2)                                          ;           BSEL 4,5 AND CODE 6 INTO BSEL 2.
   (2)                                          ;
   (2)  022216  012537  022226          $PUTS:  MOV     (5)+,1$                 ;GET ADDR OF ADDR. OF A/D
   (2)  022222  004537  021544                  JSR     R5,$INLP
   (2)  022226  000000                  1$:     .WORD   0
   (2)  022230  022324                          .WORD   10$
   (2)  022232  113777  021606  157140          MOVB    $OFS,@#KMAD6
   (2)  022240  113777  021606  157134          MOVB    $OFS,@#KMAD7
   (2)  022246  013737  022226  022266          MOV     1$,2$
   (2)  022254  062737  000002  022266          ADD     #2,2$
   (2)  022262  004537  021544                  JSR     R5,$INLP
   (2)  022266  000000                  2$:     .WORD   0
   (2)  022270  022324                          .WORD   10$
   (2)  022272  113777  021606  157072          MOVB    $OFS,@#KMAD3
   (2)  022300  152777  000340  157072          BISB    #340,@#KMAD6
   (2)  022306  152777  000300  157066          BISB    #300,@#KMAD7
   (2)  022314  152777  000300  157050          BISB    #300,@#KMAD3
   (2)  022322  000205                          RTS     R5
   (2)  022324  000000                  10$:    .WORD   0
   (2)
  2440          000001                          .END
```

```
ABASE = 170416           1179#   1194
ACDW1 = 000000           1194
ACDW2 = 000000           1194
ACPUOP= 000000           1194
ACTRLC  011410           2129    2140    2177#   2210    2224    2264    2300    2309    2418    2419
AD      002166           1309#   1329
ADDW0 = 000000           1194
ADDW1 = 000000           1194
ADDW10= 000000           1194
ADDW11= 000000           1194
ADDW12= 000000           1194
ADDW13= 000000           1194
ADDW14= 000000           1194
ADDW15= 000000           1194
ADDW2 = 000000           1194
ADDW3 = 000000           1194
ADDW4 = 000000           1194
ADDW5 = 000000           1194
ADDW6 = 000000           1194
ADDW7 = 000000           1194
ADDW8 = 000000           1194
ADDW9 = 000000           1194
ADEVCT= 000000           1194
ADEVM = 000000           1194
AENV  = 000000           1194
AENVM = 000000           1194
AFATAL= 000000           1194
AL      002164           1308#   1327
AMADR1= 000000           1194
AMADR2= 000000           1194
AMADR3= 000000           1194
AMADR4= 000000           1194
AMAMS1= 000000           1194
AMAMS2= 000000           1194
AMAMS3= 000000           1194
AMAMS4= 000000           1194
AMSGAD= 000000           1194
AMSGLG= 000000           1194
AMSGTY= 000000           1194
AMTYP1= 000000           1194
AMTYP2= 000000           1194
AMTYP3= 000000           1194
AMTYP4= 000000           1194
APASS = 000000           1194
APRIOR= 000200           1181#   1194
APTCSU= 000040           2427    2430#
APTENV= 000001           2419    2427    2430#
APTSIZ= 000200           1272    2430#
APTSPO= 000100           2427    2430#
ASWREG= 000000           1194
ATESTN= 000000           1194
AUNIT = 000000           1194
AUSWR = 000000           1194
AVECT1= 100360           1180#   1194    1252
AVECT2= 000000           1194
BEG     001526           1268    1270#
```

```
BEGIN   001502          1185    1265#
BEGIN1  001520          1186    1269#   2422
BIT0  = 000001          1177#   1674
BIT00 = 000001          1177#
BIT01 = 000002          1177#
BIT02 = 000004          1177#
BIT03 = 000010          1177#
BIT04 = 000020          1177#
BIT05 = 000040          1177#
BIT06 = 000100          1177#
BIT07 = 000200          1177#
BIT08 = 000400          1177#   2418
BIT09 = 001000          1177#
BIT1  = 000002          1177#   1570    1572    1747    2439
BIT10 = 002000          1177#   1653    1655    1732    1756    2250    2289
BIT11 = 004000          1177#   1441    1466    1492    1519    1662    1664    1756    2439
BIT12 = 010000          1177#   1728    1734    2253
BIT13 = 020000          1177#   2148    2156    2419
BIT14 = 040000          1177#   1981    2152    2156    2418    2439
BIT15 = 100000          1177#   1381    2439
BIT2  = 000004          1177#   1586    1588    1685    1686    1695    1702    1747
BIT3  = 000010          1177#   1603    1605    1709    1710    1718    1724    1747
BIT4  = 000020          1177#   1620    1622    1747    1824    1836    1846    1856    1930    2059    2111    2439
BIT5  = 000040          1177#   2248    2257    2439
BIT6  = 000100          1177#   1635    1637    1756
BIT7  = 000200          1177#   1562    1572    1579    1588    1595    1605    1613    1622    1629    1637    1646    1655
                        1664    1676    1685    1702    1709    1724    1749    1758    1856
BIT8  = 000400          1177#   2312
BIT9  = 001000          1177#   1644    1646    1756
BPTVEC= 000014          1177#
BRLEV1  014636          1301*   1302*   1303    2335*   2336*   2409#
BRLEV2  014640          1303*   1304*   2410#
BUFFER  020234          2436#
BUF2    017654          2434#
BUF3    020044          2435#
CALDAC  011532          1345    2207#
CALDON  014304          2385#
CHRCOL  011020          2099#
CHTIME  012354          1902    1920    1958    2035    2335#
CKSWR = 104407          1322    2141    2209    2223    2247    2271    2299    2308    2418    2419    2432#
CLKINT  022012          2439#
CLRVC   011732          2247#
CONTC   014113          1348    2376#
CPTYPE  012400          2335    2342#
CR    = 000015          1177#   2427
CRLF  = 000200          1177#   2427
CSPACE  011452          2189#   2190    2201
CTRLC   002346          1316    1347#   2183    2195
DAC0    001466          1258#   1291    1412    1426    1427    1434    1435    1442    1443    1529    1539    1697    1767
                        1770    1817    1868*   1895    1914    1959    2036    2113    2146    2213    2231    2287    2393
DAC1    001470          1259#   1414    1451    1452    1459    1460    1467    1468    1531    1545    1720    1777    1780
                        1818    1869*   1896    1913    1960    2037    2115    2150    2214    2232    2288    2394
DAC2    001472          1260#   1416    1477    1478    1485    1486    1493    1494    1533    1551    1787    1790    1819
                        1870*   1899    1918    1963    2040    2117    2154    2215    2233    2284    2395
DAC3    001474          1261#   1418    1504    1505    1512    1513    1520    1521    1535    1557    1797    1800    1820
                        1871*   1900    1917    1964    2041    2119    2158    2216    2234    2285    2396
```

```
DDISP = 177570          1177#   1194    1272
DELAY   001460          1255#   2165    2170
DF0     014600          1202    1208    1214    1220    1226    1232    1244    1250    2400#
DH1     013756          1200    1206    1212    1218    1224    1248    2370#
DH13    014026          1230    2372#
DH14    014060          2373#
DH6     014012          1242    2371#
DISPLA  001142          1194#   1272*   2418*   2419*
DISPRE  000174          1185#   1272
DRLPX?= ****** G          52#   2439
DSWR  = 177570          1177#   1194    1272
DT1     014464          120:    1249    2392#
DT13    014554          1231    2398#
DT14    014566          2399#
DT2     014476          1207    2393#
DT3     014510          1213    2394#
DT4     014522          1219    2395#
DT5     014534          1225    2396#
DT6     014546          1243    2397#
DYNCAL  011622          2221#
EMTVEC= 000030          1177#   1272*
EM1     013450          1199    2362#
EM10    013635          1241    2367#
EM11    013706          1247    2368#
EM13    013735          1229    2369#
EM2     013501          1205    2363#
EM3     013530          1211    2364#
EM4     013557          1217    2365#
EM5     013606          1223    2366#
ERRVEC= 000004          1177#   1272*   1409*   1423*   2418*
EVER    014642          1266*   1368    1380*   1381*   1383    1865    2398    2411#
EVER1   014644          1267*   1887    1893*   2412#
FOUND1  014121          1373    2377#
FOUND2  014145          1379    2379#
FULRMP  011024          1344    2106#
GNS   = ****** U        1185    2432    2439
GTSWR = 104406          2432#
HIGH    014622          2402#
HT    = 000011          1177#   2427
INCR    014624          2403#
INIT:   002176          1276    1314#
IOTVEC= 000020          1177#   1272*
IV      001476          1262#   1289    1297*   1298*   1299    1872*
IVS     001500          1263#   1299*   1300*   1873*
KMAD0   001364          1252#   1273    2439*
KMAD1   001366          1252#   1273
KMAD2   001370          1252#   2439*
KMAD3   001372          1252#   2439*
KMAD4   001374          1252#   2439*
KMAD5   001376          1252#
KMAD6   001400          1252#   2439*
KMAD7   001402          1252#   1273    2439*
LDTRAP  002014          1278#   1314    1387    2107    2137    2208    2222
LF    = 000012          1177#   2427
LOADVC  012110          2278#
LOW     014620          2401#
```

```
LPADH   001376          1252#
LPADL   001374          1252#
LPCI    001364          1252#
LPCO    001370          1252#
LPMR    001366          1252#
LPMS1   001400          1252#
LPMS2   001402          1252#
LPSO    001372          1252#
LTEST   002542          1389#   1875
MANHED  014343          2139    2387#
MANUL   011216          1343    2136#
MC      002174          1312#   1335
MD      002172          1311#   1333
MES15   013133          1892    2357#
MES3    013036          2270    2355#
MES6    013103          1890    2356#
ML      002170          1310#   1331
MTEST   002360          1318    1341    1354#   1877
OPRIN   014646          1324*   1325*   1327    1329    1331    1333    1335    2413#
PBB     007330          1903    1921    1925#
PICO    007142          1888    1895#   2090    2329
PIC1    007242          1906    1913#   2330
PIC3    007564          1924    1955#   2331
PIC4    010322          2028    2032#   2332
PIC4B   010352          2036#
PIC6    011024          2105#   2333
PIRQ  = 177772          1177#
PIRQVE= 000240          1177#
PR0   = 000000          1177#
PR1   = 000040          1177#
PR2   = 000100          1177#
PR3   = 000140          1177#
PR4   = 000200          1177#
PR5   = 000240          1177#
PR6   = 000300          1177#
PR7   = 000340          1177#
PS    = 177776          1177#   1275*   2439*
PSW   = 177776          1177#
PWRMSG  015670          2422    2423#
PWRVEC= 000024          1177#   1272*   2422*
P3      007664          1967#   2029
P3A     010012          1979    1983#   1991
P3B     010072          1994#   2002
P3C     010152          2005#   2013
P3D     010232          2016#   2025
P4      010422          2044#   2091
P4A     010550          2057    2061#   2070
P4B     010672          2077#   2087
P7CNT   014650          2414#
P7PNT   014652          2415#
QMARK   014105          1338    2200    2375#
RDCHR = 104410          2428    2432#
RDLIN = 104411          1323    2429    2432#
RDOCT = 104412          2432#   2439
RD1     021346          2439#*
RESVEC= 000010          1177#
```

```
ROUTP1  012342          2324    2329#
RTCCSR  022020          2439#*
ROX     007230          1895*   1899*   1907#   1913*   1917*   1940    1942    1951    2113*   2115*   2117*   2119*   2124
                        2126
ROY     007234          1909#
R1X     007236          1910#   1959*   1963*   1970    1974    1983    1985    2005    2007    2036*   2040*   2048    2051
                        2067    2075    2083    2085
R1Y     007232          1896*   1900*   1908#   1914*   1918*   1938    1950
R2Y     007240          1911#   1960*   1964*   1971    1975    1994    1996    2016    2018    2037*   2041*   2047    2052
                        2064    2066    2073    2080    2082
SDELAY  021736          2439#
SELD01  014172          2381#
SELD23  014237          2383#
STACK = 001100          1177#   1272    1321    1886    2106    2136    2207    2221
STAT    001464          1257#   1287    1410    1537    1564    1571    1573    1578    1580    1587    1589    1594    1596
                        1604    1606    1612    1614    1621    1623    1628    1630    1636    1638    1645    1647    1654
                        1656    1663    1665    1675    1677    1687    1688    1698    1711    1712    1721    1733    1735
                        1736    1748    1751    1757    1760    1825    1826    1828    1830    1837    1847    1850    1851
                        1852    1857    1867*   1926    1929    1931    1932    1943    1945    1956    1965    1980    1982
                        1986    1988    1997    1999    2008    2010    2019    2021    2033    2042    2058    2060    2061
                        2063    2077    2079    2110    2112    2160    2251    2252    2254    2259    2279    2281    2392
                        2397    2398    2399
STKLMT= 177774          1177#
SWR     001140          1194#   1272*   1728    1765    1775    1785    1795    1817    1818    1819    1820    1822    1844
                        1897    1915    1927    1935    1961    1967    1978    2038    2044    2056    2108    2142    2211
                        2225    2248    2257    2267    2272    2282    2310    2418    2419    2422*   2428*
SWREG   000176          1185#   1272    2428
SW0   = 000001          1177#
SW00  = 000001          1177#
SW01  = 000002          1177#
SW02  = 000004          1177#
SW03  = 000010          1177#   1897    1915    1961    2038    2282
SW04  = 000020          1177#
SW05  = 000040          1177#
SW06  = 000100          1177#
SW07  = 000200          1177#
SW08  = 000400          1177#
SW09  = 001000          1177#   1765    1775    1785    1795    1817    1818    1819    1820    1935    1967    2044
SW1   = 000002          1177#
SW10  = 002000          1177#   1822    1844    1927    1978    2056    2108
SW11  = 004000          1177#
SW12  = 010000          1177#   2267
SW13  = 020000          1177#
SW14  = 040000          1177#
SW15  = 100000          1177#
SW2   = 000004          1177#
SW3   = 000010          1177#
SW4   = 000020          1177#
SW5   = 000040          1177#
SW6   = 000100          1177#
SW7   = 000200          1177#
SW8   = 000400          1177#
SW9   = 001000          1177#
TBITVE= 000014          1177#
TEMP    014632          1265*   1269*   1315    1731*   1739*   2406#
TEMP1   014634          2280*   2290    2408#
```

```
TICKS   014630          1901*  1919*  1957*  2034*  2314*  2338*  2405#
TIME    022010          2439#*
TIMER   012246          1904   1922   2026   2088   2308#
TIMERA  012262          2312#
TIMER1  012304          2315   2317#
TIMER2  012306          2313   2322#
TIMER4  012336          2327#
TIMSV   014626          2310*  2312   2322*  2323*  2324*  2325*  2327   2404#
TITLE   012402          1320   2347#
TKVEC = 000060          1177#
TPVEC = 000064          1177#
TRAPVE= 000034          1177#  1272*
TRTVEC= 000014          1177#
TST1    002544          1367   1407#
TST10   003216          1475#
TST11   003256          1480   1483#
TST12   003320          1488   1491#
TST13   003370          1502#
TST14   003430          1507   1510#
TST15   003472          1515   1518#
TST16   003542          1527#
TST17   004012          1559   1561#
TST2    002672          1424#
TST20   004050          1566   1569#
TST21   004166          1582   1585#
TST22   004304          1598   1602#
TST23   004422          1616   1619#
TST24   004540          1632   1634#
TST25   004610          1640   1643#
TST26   004660          1649   1652#
TST27   004730          1658   1661#
TST3    002732          1429   1432#
TST30   005000          1667   1670#
TST31   005060          1679   1682#
TST32   005206          1693   1701   1706#
TST33   005334          1717   1723   1727#
TST34   005444          1729   1738   1746#
TST35   005520          1753   1755#
TST36   005574          1762   1764#
TST37   005656          1765   1772   1774#
TST4    002774          1437   1440#
TST40   005740          1775   1782   1784#
TST41   006020          1785   1792   1794#
TST42   006100          1795   1802   1817#
TST43   006164          1817   1818#
TST44   006250          1818   1819#
TST45   006334          1819   1820#
TST46   006420          1820   1821#
TST47   006554          1823   1835   1839   1843#
TST5    003044          1449#
TST50   006672          184.   1862#
TST6    003104          1454   1457#
TST7    003146          1462   1465#
TYPDS = 104405          1877   2432#
TYPE  = 104401          1319   1337   1347   1372   1378   1877   1889   1891   2138   2199   2269   2417   2419
                        2420   2422   2426   2427   2428   2432#  2439
```

```
TYPOC = 104402           2420    2428    2432#   2439
TYPON = 104404           2432#
TYPOS = 104403           1375    2432#   2439
VADDR   001454           1253#   1360    1867    1868    1869    1870    1871
VCXTMP  011010           2095#
VCYTMP  011012           2096#
VECTOR  001404           1252#
VECTPS  001406           1252#
VERSN   001410           1252#
VISUAL  007114           1342    1886#
VVECT   001456           1254#   1872    1873
WAITIN  002222           1321#   1339    1350
W1      021176           2439#*
W2      021200           2439#*
W3      021202           2439#*
XPOS    011016           2098#
YPOS    011014           2097#
YPT     011022           2100#
$AERR   020726           1358    2439#*
$APTHD  001000           1193#
$ASTAT= ****** U         2430
$ATYC   017350           2430#
$ATY1   017324           2430#
$ATY3   017332           2427    2430#
$ATY4   017342           2419    2430#
$AUTOB  001134           1194#   1370    2428
$BASE   001246           1194#   1288    1354
$BDADR  001122           1194#
$BDDAT  001126           1194#   1354*   1357    1360*   1427    1428    1435    1436    1443    1444    1452    1453    1460
                         1461    1468    1469    1478    1479    1486    1487    1494    1495    1505    1506    1513    1514
                         1521    1522    1539    1540    1545    1546    1551    1552    1557    1558    1564    1565    1573
                         1574    1580    1581    1589    1590    1596    1597    1606    1607    1614    1615    1623    1624
                         1630    1631    1638    1639    1647    1648    1656    1657    1665    1666    1677    1678    1688
                         1689    1698    1700    1712    1713    1721    1722    1751    1752    1760    1761    1770    1771
                         1780    1781    1790    1791    1800    1801    1817    1818    1819    1820    1837    1838    1857
                         2163*   2164    2169    2392    2393    2394    2395    2396    2399
$CDW1   001252           1194#
$CHARC  016442           2427#*
$CKSWR  016446           2428#   2432
$CMTAG  001100           1194#   1272
$CM1  = 000002           1194#
$CM2  = 000004           1194#
$CM3  = 000002           1194#
$CNT    021676           2439#*
$CNTLG  017173           2428#
$CNTLU  017166           2428#
$CPUOP  001220           1194#
$CRLF   001167           1194#   2419    2420    2427    2428
$DATR   021350           2439#*
$DBLK   015070           2417#
$DEVCT  001202           1194#
$DEVM   001250           1194#
$DOAGN  007070           1877#
$DTBL   015060           2417#
$ENDAD  007060           1191    1877#
$ENDCT  007026           1877#
```

```
$ENDMG  007077          1877#
$ENULL  007074          1877#
$ENV    001212          1194#   2419    2427    2430
$ENVM   001213          1194#   1272    2427    2430
$EOP    006776          1866    1877#
$EOPCT  007020          1877#
$ERFLG  001103          1194#   2418*   2419*
$ERMAX  001115          1194#   2418
$ERROR  015232          1272    2419#
$ERRPC  001116          1194#   2392    2393    2394    2395    2396    2397    2398    2399    2419*   2420
$ERRTB  001254          1194#   2420
$ERRTY  015362          2419    2420#
$ERTTL  001112          1194#   2419*
$ETABL  001212          1194#
$ETEND  001254          1193    1194#
$FATAL  001174          1194#   2430*
$FFLG   017570          2430#*
$FILLC  001156          1194#   2427
$FILLS  001155          1194#   2427
$GDADR  001120          1194#
$GDDAT  001124          1194#   1425*   1426    1428    1433*   1434    1436    1441*   1442    1444    1447*   1450*   1451
                        1453    1458*   1459    1461    1466*   1467    1469    1472*   1476*   1477    1479    1484*   1485
                        1487    1492*   1493    1495    1498*   1503*   1504    1506    1511*   1512    1514    1519*   1520
                        1522    1525*   1538*   1540    1544*   1546    1550*   1552    1556*   1558    1562*   1565    1572*
                        1574    1579*   1581    1588*   1590    1595*   1597    1605*   1607    1613*   1615    1622*   1624
                        1629*   1631    1637*   1639    1646*   1648    1655*   1657    1664*   1666    1673*   1676*   1678
                        1685*   1695*   1702*   1709*   1718*   1724*   1749*   1752    1758*   1761    1768*   1771    1778*
                        1781    1788*   1798*   1817*   1818*   1819*   1820*   1836*   1838    1856*   1943    1944*   1945
                        2162*   2164    2168*   2169    2392    2393    2394    2395    2396    2399
$GET42  007050          1877#
$GTSWR  016516          2428#   2432
$HD   = 000000          1176
$HIBIS  001000          1193#
$HIOCT  017322          2429#*
$ICNT   001104          1194#
$ILLUP  015662          2422#
$INIP   021544          1357    1410    1412    1414    1416    1418    1427    1435    1443    1452    1460    1468    1478
                        1486    1494    1505    1513    1521    1539    1545    1551    1557    1564    1573    1580    1589
                        1596    1606    1614    1623    1630    1638    1647    1656    1665    1677    1688    1698    1712
                        1721    1736    1751    1760    1770    1780    1790    1800    1817    1818    1819    1820    1826
                        1830    1837    1851    1857    1929    1943    1980    1983    1986    1994    1997    2005    2008
                        2016    2019    2058    2061    2064    2077    2080    2083    2110    2252    2259    2439#
$INTAG  001135          1194#   2428
$ITEMB  001114          1194#   2419*   2420
$LF     001170          1194#   2419    2427    2428
$LFLG   017567          2430#*
$LOAD   020730          2439#
$LPADR  001106          1194#   1272*   1408*   2418*
$LPAI   020240          2439#
$LPERR  001110          1194#
$LPW    021352          2439#
$MADR1  001224          1194#
$MADR2  001230          1194#
$MADR3  001234          1194#
$MADR4  001240          1194#
$MAIL   001172          1193    1194#   1272    2418    2419    2427
```

```
$MAMS1  001222          1194#
$MAMS2  001226          1194#
$MAMS3  001232          1194#
$MAMS4  001236          1194#
$MBADR  001002          1193#
$MFLG   017566          2430#*
$MNEW   017211          2428#
$MSGAD  001206          1194#   2430*
$MSGLG  001210          1194#   2430*
$MSGTY  001172          1194#   2430*
$MSWR   017200          2428#
$MTYP1  001223          1194#
$MTYP2  001227          1194#
$MTYP3  001233          1194#
$MTYP4  001237          1194#
$NULL   001154          1194#   2427
$NWTST= 000001          1407#   1424#   1432#   1440#   1449#   1457#   1465#   1475#   1483#   1491#   1502#   1510#   1518#
                        1527#   1561#   1569#   1585#   1602#   1619#   1634#   1643#   1652#   1661#   1670#   1682#   1706#
                        1727#   1746#   1755#   1764#   1774#   1784#   1794#   1817#   1818#   1819#   1820#   1821#   1843#
                        1862#
$OCNT   016160          2426#*
$OFS  = 021606          2439#
$OMODE  016162          2426#*
$OUTLP  021446          1426    1434    1442    1451    1459    1467    1477    1485    1493    1504    1512    1520    1529
                        1531    1533    1535    1537    1571    1578    1587    1594    1604    1612    1621    1628    1636
                        1645    1654    1663    1675    1687    1697    1711    1720    1733    1735    1748    1757    1767
                        1777    1787    1797    1817    1818    1819    1820    1825    1828    1847    1850    1926    1931
                        1938    1940    1942    1945    1950    1951    1956    1970    1971    1974    1975    1982    1985
                        1988    1996    1999    2007    2010    2018    2021    2033    2047    2048    2051    2052    2060
                        2063    2066    2067    2073    2075    2079    2082    2085    2112    2124    2126    2164    2169
                        2213    2214    2215    2216    2231    2232    2233    2234    2251    2254    2279    2439#
$OVER   015216          2418#
$PASS   001200          1194#   1272*   1349*   1877*
$PASTM  001006          1193#
$PUTS   022216          2439#
$PWRAD  015656          2422#
$PWRDN  015516          1272    2422#
$PWRMG  015652          2422#
$PWRUP  015570          2422#
$QUES   001166          1194#   2419    2427    2428
$RDCHR  016730          2428#   2432
$RDDEC= ****** U        2432
$RDLIN  017050          2428#   2432
$RDOCT  017222          2429#   2432
$RDSZ = 000010          2428#
$REGAD  001160          1194#
$REG0   001162          1194#
$REG1   001164          1194#
$RESET  021700          1563    1750    1759    1769    1779    1789    1799    1817    1818    1819    1820    1860    1863
                        2439#
$RTNAD  007072          1877#
$R2A  = ****** U        2432
$SAD    021674          2439#*
$SAVRE= ****** U        2432
$SAVR6  015666          2422#*
$SCOPE  015100          1272    2418#
```

```
$SETUP= 000117   1196#  1271#  1272  1877  2418  2419  2428
$STUP = 177777   1196#  1271#
$SVLAD  015174   2418#
$SVPC = 000220   1191#
$SWR  = 160400   1155#  1176  1183  1194  1272  1407  1424  1432  1440  1449  1457  1465  1475
                 1483  1491  1502  1510  1518  1527  1561  1569  1585  1602  1619  1634  1643
                 1652  1661  1670  1682  1706  1727  1746  1755  1764  1774  1784  1794  1817
                 1818  1819  1820  1821  1843  1862  1877  2418  2419  2422
$SWREG  001214   1194#  1272
$SWRMK= 000000   1183  2418
$TESTN  001176   1194#  2418*
$TKB    001146   1194#  2179  2191  2428  2439
$TKS    001144   1194#  2177  2189  2428*  2439
$TLKR   021204   2439#
$TLKW   021070   2439#
$TMDAT  001462   1256#  1357  1410  1411  1412  1413  1414  1415  1416  1417  1418  1419  1528*
                 1529  1530*  1531  1532*  1533  1534*  1535  1536*  1537  1570*  1571  1577*  1578
                 1586*  1587  1593*  1594  1603*  1604  1611*  1612  1620*  1621  1627*  1628  1635*
                 1636  1644*  1645  1653*  1654  1662*  1663  1674*  1675  1686*  1687  1696*  1697
                 1710*  1711  1719*  1720  1732*  1733  1734*  1735  1736  1737  1747*  1748  1756*
                 1757  1766*  1767  1776*  1777  1786*  1787  1796*  1797  1817*  1818*  1819*  1820*
                 1824*  1825  1826  1827*  1828  1830  1831  1846*  1847  1849*  1850  1851  1925*
                 1926  1929  1930*  1931  1937*  1938  1939*  1940  1941*  1942  1946  1949*  1950
                 1951  1955*  1956  1969*  1970  1971  1973*  1974  1975  1980  1981*  1982  1983
                 1984*  1985  1986  1987*  1988  1994  1995*  1996  1997  1998*  1999  2005  2006*
                 2007  2008  2009*  2010  2016  2017*  2018  2019  2020*  2021  2032*  2033  2046*
                 2047  2048  2050*  2051  2052  2058  2059*  2060  2061  2062*  2063  2064  2065*
                 2066  2067  2072*  2073  2074*  2075  2077  2078*  2079  2080  2081*  2082  2083
                 2084*  2085  2110  2111*  2112  2123*  2124  2125*  2126  2127  2212*  2213  2214
                 2215  2216  2231  2232  2233  2234  2250*  2251  2252  2253*  2254  2259  2260
                 2278*  2279
$TN   = 000051   1156#  1176  1367  1407#  1424#  1429  1432#  1437  1440#  1449#  1454  1457#  1462
                 1465#  1475#  1480  1483#  1488  1491#  1502#  1507  1510#  1515  1518#  1527#  1559
                 1561#  1566  1569#  1582  1585#  1598  1602#  1616  1619#  1632  1634#  1640  1643#
                 1649  1652#  1658  1661#  1667  1670#  1679  1682#  1693  1701  1706#  1717  1723
                 1727#  1729  1738  1746#  1753  1755#  1762  1764#  1765  1772  1774#  1775  1782
                 1784#  1785  1792  1794#  1795  1802  1817#  1818#  1819#  1820#  1821#  1823  1835
                 1839  1843#  1845  1862#
$TOUT   021640   2439#
$TPB    001152   1194#  2427*
$TPFLG  001157   1194#  2427
$TPS    001150   1194#  2427
$TRAP   017572   1272  2432#
$TRAP2  017614   2432#
$TRP  = 000013   2432#
$TRPAD  017626   2432#
$TSTM   001004   1193#
$TSTNM  001102   1194#  1874*  1877*  2418*  2419
$TTYIN  017156   2428#
$TYPBN= ****** U  2432
$TYPDS  014654   2417*  2432
$TYPE   016164   2427#  2430  2432
$TYPEC  016376   2427#  2428
$TYPEX  016444   2427#
$TYPOC  015762   2426#  2432
$TYPON  015776   2426#  2432
```

```
$TYPOS  015736          2426#   2432
$UNIT   001204          1194#   1356*   1361*   1364    1374    1380    1383    1386*   1864*   1865    2398
$UNITM  001010          1193#
$USWR   001216          1194#
$UTK    022022          1189    2439#
$VECT1  001242          1194#   1297    1301
$VECT2  001244          1194#
$XTSTR  015116          2418#
$$GET4= 000000          1877#
$OFILL  016161          2426#*
$40CAT= ****** U        2418    2419
.       = 022326        1185#   1191#   1192#   1193#   1194#   1252#   1272    1877    2296    2417#   2418    2419    2420#
                        2422    2427    2428#   2430#   2434#   2435#   2439#
.DVLS   001412          1252#   1273*   1355*   2439*
.$ASTA= ****** U        2430
.$X     = 001000        1193#
```

```
COMMEN  1177#
CROSS   1393#
CSPACE  1390#
ENDCOM  1177#
ERROR   1177#  1366  1385  1422  1430  1438  1446  1455  1463  1471  1481  1489  1497  1508  1516
        1524   1542  1548  1554  1560  1567  1576  1583  1592  1599  1609  1617  1626  1633  1641
        1650   1659  1668  1680  1691  1703  1715  1725  1743  1754  1763  1773  1783  1793  1803
        1817   1818  1819  1820  1840  1858  2439
ESCAPE  1177#
GETPRI  1177#
GETSWR  1177#
MOVEI    170#  1357  1410  1412  1414  1416  1418  1427  1435  1443  1452  1460  1468  1478  1486
        1494   1505  1513  1521  1539  1545  1551  1557  1564  1573  1580  1589  1596  1606  1614
        1623   1630  1638  1647  1656  1665  1677  1688  1698  1712  1721  1736  1751  1760  1770
        1780   1790  1800  1817  1818  1819  1820  1826  1830  1837  1851  1857  1929  1943  1980
        1983   1986  1994  1997  2005  2008  2016  2019  2058  2061  2064  2077  2080  2083  2110
        2252   2259  2439
MOVEM    157#  1426  1434  1442  1451  1459  1467  1477  1485  1493  1504  1512  1520  1529  1531
        1533   1535  1537  1571  1578  1587  1594  1604  1612  1621  1628  1636  1645  1654  1663
        1675   1687  1697  1711  1720  1733  1735  1748  1757  1767  1777  1787  1797  1817  1818
        1819   1820  1825  1828  1847  1850  1926  1931  1938  1940  1942  1945  1950  1951  1956
        1970   1971  1974  1975  1982  1985  1988  1996  1999  2007  2010  2018  2021  2033  2047
        2048   2051  2052  2060  2063  2066  2067  2073  2075  2079  2082  2085  2112  2124  2126
        2164   2169  2213  2214  2215  2216  2231  2232  2233  2234  2251  2254  2279
MULT    1177#
NEWONE  1804#  1817  1818  1819  1820
NEWTST  1177#  1407  1424  1432  1440  1449  1457  1465  1475  1483  1491  1502  1510  1518  1527
        1561   1569  1585  1602  1619  1634  1643  1652  1661  1670  1682  1706  1727  1746  1755
        1764   1774  1784  1794  1817  1818  1819  1820  1821  1843  1862
POP     1177#  2417  2422  2429  2430
PUSH    1177#  2417  2422  2429  2430
REPORT  1177#
SCOPE   1177#  1407  1424  1432  1440  1449  1457  1465  1475  1483  1491  1502  1510  1518  1527
        1561   1569  1585  1602  1619  1634  1643  1652  1661  1670  1682  1706  1727  1746  1755
        1764   1774  1784  1794  1817  1818  1819  1820  1821  1843  1862  1877
SETPRI  1177#
SETTRA  2432#
SETUP   1177#  1272
SKIP    1177#  1367  1382  1384  1420  1429  1437  1445  1454  1462  1470  1480  1488  1496  1507
        1515   1523  1541  1547  1553  1559  1566  1575  1582  1591  1598  1608  1616  1625  1632
        1640   1649  1658  1667  1679  1690  1693  1701  1714  1717  1723  1729  1738  1740  1742
        1753   1762  1765  1772  1775  1782  1785  1792  1795  1802  1817  1818  1819  1820  1823
        1835   1839  1845  1866
SLASH   1177#
SPACE   1177#
STARS   1177#  1191  1193  1194  1407  1424  1432  1440  1449  1457  1465  1475  1483  1491  1502
        1510   1518  1527  1561  1569  1585  1602  1619  1634  1643  1652  1661  1670  1682  1706
        1727   1746  1755  1764  1774  1784  1794  1817  1818  1819  1820  1821  1843  1862  1877
        2417   2418  2419  2420  2422  2426  2427  2428  2429  2430  2432
SWRSU   1177#  1272#
TOUT    1252#  2439
TRMTRP  2432#
TSTSW9  1397#  1765  1775  1785  1795
TYPBIN  1177#
TYPDEC  1177#  1877
TYPNAM  1177#
```

```
TYPNUM   1177#
TYPOCS   1177#
TYPOCT   1177#  2420  2428  2439
TYPTXT   1177#  2439
$CAL.     748#  2439
$DMAST    914#
$DMDT    1048#
$MMAST    772#
$SCMRE   1194#
$SCMTM   1194#
$SESCA   1177#
$SNEWT   1177#  1407  1424  1432  1440  1449  1457  1465  1475  1483  1491  1502  1510  1518  1527
         1561   1569  1585  1602  1619  1634  1643  1652  1661  1670  1682  1706  1727  1746  1755
         1764   1774  1784  1794  1817  1818  1819  1820  1821  1843  1862
$SSET    2432#
$SSETM   1272#
$SSKIP   1177#  1367  1429  1437  1454  1462  1480  1488  1507  1515  1559  1566  1582  1598  1616
         1632   1640  1649  1658  1667  1679  1693  1701  1717  1723  1729  1738  1753  1762  1765
         1772   1775  1782  1785  1792  1795  1802  1817  1818  1819  1820  1823  1835  1839  1845
.EQUAT   1157#  1177
.HEADE   1157#  1176
.KMADR     55#  1252
.KSIS     184#  1273
.LOADL    459#  2439
.LPAIN    209#  2439
.PUTCS    418#  2439
.RESET    329#  2439
.SETUP   1158#  1196  1271
.SWRHI   1159#  1183
.SWRLO   1183#
.UTK      699#  2439
.$ACT1   1161#  1191
.$APIB   1161#  1194#
.$APTH   1161#  1193
.$APTY   1161#  2430
.$CATC   1157#  1185
.$CMTA   1157#  1194
.$EOP    1157#  1877
.$ERRO   1157#  2419
.$ERRT   1159#  2420
.$INLP    652#  2439
.$MMAC    141#
.$OUTL    610#  2439
.$PARM   1158#
.$POWE   1158#  2422
.$RDOC   1160#  2429
.$READ   1158#  2428
.$SAVE   1158#
.$SCOP   1158#  2418
.$SPAC   1158#
.$SWDO   1158#
.$TLKW    511#  2439
.$TOUT   1252#  2439
.$TRAP   1158#  2432
.$TYPD   1159#  2417
.$TYPE   1157#  1158#  2427
```

```
.$TYPO   1157#   2426


. ABS.  022326     000     OVR   RO    ABS   GBL   D
        000000     001     CON   RW    REL   LCL   I


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 CRLPBB,CRLPBB/CRF=CRLPAB.MAC,CRLPBB.P11
 RUN-TIME: 27 14 1 SECONDS
 RUN-TIME RATIO: 97/43=2.2
 CORE USED:  35K  (69 PAGES)
```

```
                                ;THIS FILE IS THE SAME AS "CRLPX0.P11" EXCEPT IT IS LOADED INTO 65000
                                ;IT IS ALSO THE SAME AS "DRLPX2.P11" EXCEPT NAME CHANGE "CRLPX2.P11"
                                ;
                                
                                        .LIST   MC,BIN,BEX,MEB
                                        .NLIST  MD,CND,ME
                                
          177777                        ADDRESS=-1
                                ;       MACRO DEFFINITIONS FOR M8200 AND M8204 MICRO-PROCESSOR
                                ;       INSTRUCTION SET.
                                ;       TO BE USED WITH RSX MACRO-11 ASSEMBLER
                                ;
                                ;       26-MAY-1976
000000'                                 $BEGIN
000000                                  $LOC    42000
                                        .GLOBL  DRLPX2
                                        .ENABL  GBL
                                
                                        ;*
                                        ;*MICRO CODE FOR KMC-11
```

```
                                        ;*THIS CODE WILL BE DOWN LOADED INTO BOTH
                                        ;*KMC-11'S. THE CODE RUNS ASYNCRONOUS TO THE PDP-11 CODE
                                        ;*WE SYNC THROUGH COMMANDS PASSED VIA THE OUT*/IBUS* REGS.
                                        ;*


042000                          DRLPX2: ;JUMP TABLE USED FOR COMMANDS
042000                                  BR      STARTU          ;GOTO START
(2) 042000  100407                      .WORD   .$$$.
042002                                  BR      CMNOP           ;NOP=1
(2) 042002  100420                      .WORD   .$$$.
042004                                  BR      RDSILO          ;=2 READ SILO PUT IN BSEL4
(2) 042004  100430                      .WORD   .$$$.
042006                                  BR      WRSILO          ;=3 READ BSEL4 PUT IN SILO.
(2) 042006  100432                      .WORD   .$$$.
042010                                  BR      RDCMND          ;=4 READ FAST PATH PUT IN BSEL4
(2) 042010  100434                      .WORD   .$$$.
042012                                  BR      WRCMND          ;=5 READ BSEL4, PUT IN FAST PATH.
(2) 042012  100436                      .WORD   .$$$.
042014                                  BR      SAMP            ;=6 TAKE AN A/D SAMPLE
(2) 042014  100440                      .WORD   .$$$.

                                        ;START OF U CODED


042016                          STARTU:
042016                                  MOVE    # 0,BREG
(3) 042016  000400                      .WORD   .$$$.
042020                                  MOVE    BREG,OUT1 <0>   ;CLEAR UNIBUS CSRS
(3) 042020  061220                      .WORD   .$$$.
042022                                  MOVE    BREG,OUT1 <2>
(3) 042022  061222                      .WORD   .$$$.
042024                                  MOVE    BREG,OUT1 <3>
(3) 042024  061223                      .WORD   .$$$.
042026                                  MOVE    BREG,OUT1 <4>
(3) 042026  061224                      .WORD   .$$$.
042030                                  MOVE    BREG,OUT1 <5>
(3) 042030  061225                      .WORD   .$$$.
042032                                  MOVE    BREG,OUT1 <6>
(3) 042032  061226                      .WORD   .$$$.
042034                                  MOVE    BREG,OUT1 <7>
(3) 042034  061227                      .WORD   .$$$.
042036                                  MOVE    BREG,SPAD <6>
(3) 042036  063226                      .WORD   .$$$.

042040                          CMNOP:  MOVE    INP0 <12>,OUT1 <0> ;READ STATUS
(3) 042040  021240                      .WORD   .$$$.
042042                                  MOVE    # 377,BREG
(3) 042042  000777                      .WORD   .$$$.
042044                                  MOVE    BREG,OUT1 <2>   ;INDICATE READY FOR COMMAND.
(3) 042044  061222                      .WORD   .$$$.

042046                          LOOP:   MOVE    INP0 <12>,OUT1 <0> ;READ STATUS
(3) 042046  021240                      .WORD   .$$$.

042050                                  MOVE    INP1 <2>,SPAD <0> ;READ COMMAND REG.
```

```
(3)  042050  123040                        .WORD   .$$$.
496  042052                               BZ      LOOP            ;NO COMMAND THEN LOOP
(2)  042052  101423                       .WORD   .$$$.
497
498  042054                               MOVE    INP1 <2>,SPAD <0> ;RE-READ COMMAND.
(3)  042054  123040                       .WORD   .$$$.
499
500  042056                               BR      SPAD <0>        ;BR BASED ON CMND.
(2)  042056  160600                       .WORD   .$$$.
501                                                              ;NO-USER PROTECTION OFFERED.
502                                                              ;IF YOU ENTER WRONG CODE -
503                                                              ;YOU LOSE.
504
505                                      ;
506                                      ;ROUTINE TO READ THE SILO, PUT IN
507                                      ;*BUS REG 4
508                                      ;CMD=2
509                                      ;
510  042060                        RDSILO: MOVE    INP0 <10>,OUT1 <4> ;READ SILO.
(3)  042060  021204                       .WORD   .$$$.
511                                                              ;WRITE *BUS
512  042062                               BR      CMNOP           ;RETURN.
(2)  042062  100420                       .WORD   .$$$.
513
514                                      ;
515                                      ;ROUTINE TO WRITE SILO, READ DATA FROM
516                                      ;*BUS REG 4
517                                      ;CMD=3
518                                      ;
519
520  042064                        WRSILO: MOVE    INP1 <4>,OUT0 <10> ;READ DATA IN *BUS
(3)  042064  122110                       .WORD   .$$$.
521                                                              ;WRITE SILO.
522  042066                               BR      CMNOP
(2)  042066  100420                       .WORD   .$$$.
523
524                                      ;
525                                      ;ROUTINE TO READ FAST PATH (CMND) REG.
526                                      ;PUT IN *BUS REG 4
527                                      ;CMD=4
528                                      ;
529
530  042070                        RDCMND: MOVE    INP0 <11>,OUT1 <4> ;READ FAST PATH
(3)  042070  021224                       .WORD   .$$$.
531                                                              ;WRITE *BUS.
532  042072                               BR      CMNOP           ;RETURN
(2)  042072  100420                       .WORD   .$$$.
533
534                                      ;
535                                      ;ROUTINE TO WRITE FAST PATH (CMND) REG.
536                                      ;TAKE DATA FROM *BUS REG 4.
537                                      ;CMD=5
538                                      ;
539
540  042074                        WRCMND: MOVE    INP1 <4>,OUT0 <11> ;READ DATA IN *BUS
(3)  042074  122111                       .WORD   .$$$.
```

```
                                                        ;WRITE INTO FAST PATH.
042076                                  BR      CMNOP   ;RETURN.
(2) 042076  100420                      .WORD   .$$$.

                                        ;
                                        ;THIS ROUTINE TAKES AN A/D SAMPLE.
                                        ;CALL= CMND 6 IN BSEL2
                                        ;THESE REGS. MUST BE SET UP IN ADVANCE.
                                        ;BSEL 3 MUST CONTAIN READ CODE FOR A/D BUFFER.
                                        ;BSEL 4,5 MUST CONTAIN  A/D CSR SETTING.
                                        ;BSEL 6 MUST CONTAIN WRITE CODE FOR A/D CSR
                                        ;BSEL 7 MUST CONTAIN READ CODE FOR A/D CSR
                                        ; BSEL 3,6,7 WILL REMAIN  UNEFFECTED.
                                        ; BSEL 4,5 WILL CONTAIN A/D SAMPLE.
                                        ;BSEL2 WILL CONTAIN CODE 377 WHEN DONE.

042100                                  WTMC    SAMP
(4) 042100  020640                      .WORD   .$$$.
(3) 042102  103040                      .WORD   .$$$.
042104                                  MOVE    INP1 <6>,OUTO <11>      ;SEND A/D WRITE CODE.
(3) 042104  122151                      .WORD   .$$$.
042106                                  WTMC    SAMP1
(4) 042106  020640                      .WORD   .$$$.
(3) 042110  103043                      .WORD   .$$$.
042112                                  MOVE    INP1 <4>,OUTO <11>      ;SEND LOW BYTE CSR INFO.
(3) 042112  122111                      .WORD   .$$$.
042114                                  WTMC    SAMP2
(4) 042114  020640                      .WORD   .$$$.
(3) 042116  103046                      .WORD   .$$$.
042120                                  MOVE    INP1 <5>,OUTO <11>      ;SEND HIGH BYTE CSR INFO.
(3) 042120  122131                      .WORD   .$$$.
042122                                  WTMC    SLOOP
(4) 042122  020640                      .WORD   .$$$.
(3) 042124  103051                      .WORD   .$$$.
042126                                  MOVE    INP1 <7>,OUTO <11>      ;SEND READ CODE TO GET A/D CSR.
(3) 042126  122171                      .WORD   .$$$.
042130                                  WTMC    SAMP3
(4) 042130  020640                      .WORD   .$$$.
(3) 042132  103054                      .WORD   .$$$.
042134                                  WTMM    SLOOP1
(4) 042134  020640                      .WORD   .$$$.
(4) 042136  061620                      .WORD   .$$$.
(3) 042140  103056                      .WORD   .$$$.
042142                                  MOVE    INPO <11>,BREG
(3) 042142  020620                      .WORD   .$$$.
042144                                  MOVE    BREG,SPAD <0>
(3) 042144  063220                      .WORD   .$$$.
042146                                  WTMM    SLOOP2
(4) 042146  020640                      .WORD   .$$$.
(4) 042150  061620                      .WORD   .$$$.
(3) 042152  103063                      .WORD   .$$$.
042154                                  MOVE    INPO <11>,BREG
(3) 042154  020620                      .WORD   .$$$.
042156                                  BB7     CMNOP           ;ABORT IF A/D BIT 15 1
```

```
    (2)  042156  103420                        .WORD   .$$$.
   582   042160                                MOVE    SPAD <0>,BREG
    (3)  042160  060600                        .WORD   .$$$.
   583   042162                                BB7     LOPE
    (2)  042162  103473                        .WORD   .$$$.
   584   042164                                BR      SLOOP                   ;IF A/D NOT DONE,EXIT.
    (2)  042164  100451                        .WORD   .$$$.
   585   042166                        LOPE:   MOVE    INP1 <3>,OUT0 <11>      ;ISSUE READ A/B BUFFER.
    (3)  042166  122071                        .WORD   .$$$.
   586   042170                                WTMM    SLOOP3
    (4)  042170  020640                        .WORD   .$$$.
    (4)  042172  061620                        .WORD   .$$$.
    (3)  042174  103074                        .WORD   .$$$.
   587   042176                                MOVE    INP0 <11>,OUT1 <4>
    (3)  042176  021224                        .WORD   .$$$.
   588   042200                                WTMM    SLOOP4
    (4)  042200  020640                        .WORD   .$$$.
    (4)  042202  061620                        .WORD   .$$$.
    (3)  042204  103100                        .WORD   .$$$.
   589   042206                                MOVE    INP0 <11>,OUT1 <5>
    (3)  042206  021225                        .WORD   .$$$.
   590   042210                                BR      CMNOP
    (2)  042210  100420                        .WORD   .$$$.
   591   042212  177777                        .WORD   -1
   592           000001                        .END
```

```
ADDRES= 177777          SAMP    042100          .ADDWC= 000020          .DMEM = 002400          .SELB = 000220
CLK   = 000020          SAMP1   042106          .AND  = 000260          .DNOP = 000000          .SIMM = 000000
CMNOP   042040          SAMP2   042114          .BB0  = 002000          .DOUT0= 002000          .SIN0 = 020000
DRLPX2  042000 G        SAMP3   042130          .BB1  = 002400          .DOUT1= 001000          .SIN1 = 120000
LOOP    042046          SLOOP   042122          .BB4  = 003000          .DSPAD= 003000          .SMEM = 040000
LOPE    042166          SLOOP1  042134          .BB7  = 003400          .DSPBR= 003400          .SUB  = 000340
MARHLD= 000000          SLOOP2  042146          .BC   = 001000          .DO   = 000400          .SUBWC= 000040
MARINC= 014000          SLOOP3  042170          .BR   = 000400          .FO   = 000020          .SUB2C= 000360
MARLD = 010000          SLOOP4  042200          .BSBRG= 160000          .INC  = 000060          .SO   = 020000
MARLDX= 004000          STARTU  042016          .BSIMM= 100000          .LORN = 000240          .XOR  = 000320
PAGE0 = 000000          WRCMND  042074          .BSMEM= 140000          .MINUS= 000360          .$$$. = 100420
PAGE1 = 001000          WRSILO  042064          .BZ   = 001400          .MO   = 004000          ..LOC = 042040
PAGE2 = 002000          $$$SER= 000001          .CO   = 000400          .OR   = 000300          .2A   = 000120
PAGE3 = 003000          .       = 042214          .DBR  = 000400          .PLUS = 000000          .2AWC = 000140
RDCMND  042070          .ADC    = 000100          .DBRSH= 001400          .SBREG= 060000
RDSILO  042060          .ADD    = 000000          .DEC  = 000160          .SELA = 00020C

. ABS.  042214  000     OVR   RW    ABS   LCL   D
        000000  001     CON   RW    ABS   LCL   I
ABCODE  004000  002     CON   RW    REL   LCL   I

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 CRLPX2,CRLPX2=CRLPX2
 RUN-TIME: 3 3 0 SECONDS
 RUN-TIME RATIO: 14/7=2.0
 CORE USED:  35K  (69 PAGES)
```