LPA11 LPA/DR11-K TEST CRLPFC0 AH-E435C-MC FICHE 1 OF 1 FEB 1981

IDENTIFICATION

PRODUCT CODE:	AC-E434C-MC
DIAGNOSTIC CODE:	MAINDEC-11-CRLPF-C
PRODUCT NAME:	CRLPFC0 LPA/DR11-K TEST
DATE:	DEC 1980
MAINTAINER:	DIAGNOSTIC GROUP

TABLE OF CONTENTS

1. ABSTRACT

THIS PROGRAM IS A LOGIC TEST OF THE DR11-K DIGITAL INPUT
OUTPUT CONTROL OPTION. MOST OPTION FUNCTIONS CAN BE TESTED.

DUE TO THE FLEXIBILITY OF THE OPTION, THE OPERATOR
WILL BE REQUIRED TO SUPPLY OPTION CHARACTERSTICS.
THE PROGRAM WILL HANDLE ALL CONFIGURATIONS OF INPUT INTERRUPT
SWITCHES AND INPUT DATA LATCHING JUMPERS. FOR SYSTEMS WITH
CONSECUTIVE MULTIPLE DR11-K'S, THESE CONGIGURATIONS MUST
BE THE SAME. THE FOLLOWING JUMPERS MUST
BE INSERTED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A.

THIS PROGRAM WILL TEST SEQUENTIAL DR11-K'S STARTING AT THE
BUS ADDRESS AND VECTOR IN LOCATIONS "$BASE" AND "$VECT1".
FOR NORMAL FACTORY CONFIG., ALL QUESTIONS SHOULD BE ANSWERED
WITH A VALUE OF 0.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZDRG-E"
IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT
THE DR11-K OPTION WHEN IT IS ON THE LPA11-KX 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-DZDRG-E" YOU SHOULD RUN "MD-11-CRLPA"
BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 10.

** AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC TEST MUST BE RUN. **

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERMINAL AND 16K OF MEMORY
DR11-K OPTION INSTALLED IN THE LPA-11
BC08R-1 ONE FOOT OUTPUT TO INPUT WRAPAROUND CABLE
LPA11-KX OPTION

3. LOADING PROCEDURE

THE PROCEDURE FOR LOADING A BINARY FILE SHOULD BE FOLLOWED.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS (LOGIC TEST)

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

1) TYPE CONTROL G <^G>; THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.

2) THE MACHINE WILL THEN TYPE: SWR=XXXXXX NEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)

3) AFTER THE ''NEW='' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:

   A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)
   IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.

   B) IF A CONTROL U <^U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

WILL BE USED AS THE SOFTWARE DISPLAY REGISTER.

SW 15 = 1	100000	HALT ON ERROR
SW 14 = 1	040000	LOOP ON CURRENT SUB-TEST
SW 13 = 1	020000	INHIBIT ERROR TYPOUTS
SW 12 = 1	010000	LOOP ON CURRENTLY SELECTED DR11-K
SW 10 = 1	002000	NO OUTPUT TO INPUT WRAPAROUND CABLE
SW 09 = 1	001000	LOOP ON ERROR
SW 08 = 1	000400	LOOP ON TEST IN SWR <7:0>

4.2 STARTING ADDRESSES

200	IS THE STARTING ADDRESS OF THE LOGIC TEST.
204	IS THE RESTART ADDRESS OF THE LOGIC TEST.
210	IS THE STARTING ADDRESS OF THE CONTROL LINE LOOP.
214	IS THE STARTING ADDRESS OF THE USER LINK LOOP.

5. OPERATING PROCEDURE

THE FOLLOWING JUMPERS MUST BE INSTALLED TO EXECUTE THE
LOGIC TEST: W21A, W22A AND W23A
IF THE CUSTOMER HAS SELECTED THE "B" SECTION OF THESE JUMPERS
IT MUST BE RETURNED TO THE "FACTORY" POSITION BEFORE RUNNING
THE LOGIC TEST. ** WORST CASE WILL ONLY BE CHANGING THREE JUMPERS. **
THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH
REGISTER WHEN REQUESTED BY THE PROGRAM OR AN ERROR WILL OCCUR.
WITH THE INPUT INTERRUPT SWITCHES AND DATA LATCHING JUMPERS
IN THE FACTORY POSITION, ALL SWITCH REGISTER BITS SHOULD BE RESET.
ONCE STARTED THE TEST WILL RUN IN IT'S NORMAL MANNER
WITHOUT OPERATOR INTERVENTION OR SWITCH SELECTION.
*** NOTE: OPERATOR MUST INSERT INFORMATION WHEN REQUESTED BY
PROGRAM. THE MACHINE WILL TYPE: NEW = AFTER NEW =
THE INFORMATION IS INSERTED. REFER TO SECTION 4.1 2) ***

6. ERRORS

THIS PROGRAM USES THE DIAGNOSTIC 'SYSMAC' PACKAGE FOR
ERROR REPORTING AND TYPEOUT. REFER TO THE "ERROR POINTER TABLE"
FOR TYPE AND DESCRIPTION OF ERRORS.

```
        BYTE    $TSTNM: (LOC. 1102)     CURRENT TEST NUMBER
        BYTE    $ITEMB: (LOC. 1114)     ITEM #N ERROR TABLE INDEX
        WORD    $ERRPC: (LOC. 1116)     ERRORING P.C.
        WORD    $PASS:  (LOC. 1176)     CURRENT PASS COUNT
```

7. RESTRICTIONS

1. IF SEQUENTIAL DR11-K'S, ALL DR11-K'S MUST BE IN THE
   SAME INTERRUPT SWITCHES AND DATA PATH JUMPER CONFIGERATION.
2. THE FOLLOWING JUMPERS MUST BE IN THE "FACTORY" POSITION:
   W21A, W22A AND W23A
3. THE OPERATOR MUST SUPPLY THE CORRECT INTERRUPT SWITCHES
   AND DATA PATH JUMPER AND SWITCH CONFIGURATION
   INFORMATION TO THE INITALIZATION QUESTIONS OR AN ERROR WILL OCCUR.
4. FOR MULTIPLE GROUPS OF CONSECUTIVE DR11-K'S:
   THIS DIAGNOSTIC MUST BE RUN FOR EACH GROUP.
5. AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC MUST BE RUN FIRST

8. MISCELANEOUS

8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 60 SECONDS FOR COMPLETION
ON A PDP11/05 TYPE AND WILL TYPE 'END PASS NNNN.'.
THE CONTROL LINE LOOP WILL NEVER EXIT.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS

''$BASE'' (LOC. 1244) CONTAINS THE DR11-K BASE DEVICE ADDRESS <767770>
''$VECT1'' (LOC. 1240) THE LOW 9 BITS CONTAIN THE DR11-K BASE INTERRUPT VECTOR <300>
''$VECT1'' (LOC. 1240) THE HIGH 3 BITS CONTAIN THE DR11-K BR LEVEL #4 <200>

*NOTE: IF THESE LOCATIONS ARE CHANGED, THE OPERATOR MUST START
THE TEST AGAIN AT LOC. 200. THE PROGRAM WILL USE THE BASE
ADDRESS AND VECTOR AND UPDATE THE ACTUAL PROGRAM VALUES.

8.3 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 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 ENCLOSED IN QUOTES.

8.4 XXDP/ACT/APT

THE PROGRAM IS CHAINABLE UNDER XXDP. THE APT HOOKS ARE ALSO
INSTALLED BY NOT TESTED.

9. PROGRAM DESCRIPTION

9.1 LOGIC TEST <SA 200 AND 204>

THE LOGIC TEST IS A TEST OF THE CONTROL AND INPUT/OUTPUT REGISTERS. ALL JUMPERS AND SWITCHES COMBINATIONS EXCEPT:

W21A, W22A AND W23A CAN BE DIAGNOSED.

THE PROGRAM CHECKS THAT THE DR11-K ''RESET' WILL WORK CORRECTLY.

9.2 CONTROL LINE LOOP <SA 210>

THIS TEST LOOP PROVIDES THE OPERATOR WITH A SCOPE LOOP FOR CHECKING W21, W22 AND W23 IN THE ''B'' POSITION.

9.3 USER LINK LOOP <SA 214>

THIS LOOP ENABLES THE OPERATOR TO EXAMINE OR MODIFY LOCATIONS ON THE LPA-11K I/O BUS (REFER TO 8.3).

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 BMC-11 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) FIELD 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 CRLPF-C

PRODUCT CODE: MAINDEC-11-DZDRG-B

PRODUCT NAME: DR11-K DIGITAL I/O TEST

DATE: APRIL 1976

MAINTAINER: DIANOSTIC GROUP

****

PRODUCT CODF: MAINDEC-11-DRLPF-A

PRODUCT NAME: LPA/DR11-K DIGITAL I/O TEST

DATE: JANUARY 1978

MAINTAINER: DIAGNOSTIC GROUP

REASON FOR DEVELOPMENT:

1) TO ENABLE THE OPERATOR TO CHECK OUT THE DR11-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
      DRLPF (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: DRLPF.CTL
      THIS FILE EXPLAINS SEQUENCE OF ASSEMBLES AND LINKS.
      IT IS IN TOPS-20 FORMAT.

****

SEQ 0010

PRODUCT CODE: AC-E434B-MC

DIAGNOSTIC CODE: MD-11-CRLPF-B

PRODUCT NAME: CRLPFB LPA/DR11-K DIGITAL I/O TEST

DATE: JAN. 1978

DATE REVIS'D: JULY 1979

MAINTAINER: DIAGNOSTIC GROUP

//////////

PROBLEMS

1. IF THE ''WRAP-AROUND'' CABLE IS CONNNECTED, THE PROGRAM GETS A ''BUS TRAP'' IN ''TST34''. THE PROGRAM TRYS TO ADDRESS THE DR-11K ON THE 11 BUS AND NOT THE LPA BUS

2. USER LINK SECTION HAS A NON STANDARD STARTING ADDRESS

3. SECTIONS OF THE ORIGNAL PROGRAM STILL REMAIN EVEN WHEN THE LPA-11 CANNOT SUPPORT THESE MODES

SOLUTIONS

1. A 8. LOCATION PATCH SOLVES THE PROBLEM

2. ASSIGN 214 AS THE STARTING ADDRESS FOR THE ''USER LINK'' LOOP.

3. EDIT OUT THOSE SECTIONS THAT CAN NEVER BE USED.

\\\\\\\\\\

VERSION ''C'' WAS GENERATED DUE TO CHANGE IN THE DMC-11 ROM.
THE PROGRAM IS NOW TOLERANT OF BOTH DMC-11 MICRO-CODE VERSIONS.

```
LNKX11   V023   24-OCT-80   9:36

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


LOAD MAP

IDENT:  4.01

TRANSFER ADDRESS: 000001
LOW LIMIT: 042000
HIGH LIMIT: 046000
**********
MODULE  LPA
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 TEST COULD
                                NOT BE DONE THROUGH THE LPA-11

                                  TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN
                                  TEST FOR UNEXPECTED INTERRUPT
                                  TEST THAT THE INPUT CAN INTR. USING MAINT. BIT
                                  TEST THAT THE INPUT INTR. CLEAR INT. ENABLE VIA MAINT. BIT
                                  TEST THAT THE OUTPUT CAN INTR. USING MAINT. BIT
                                  TEST FOR INTR. FROM DRA INPUT  TEST FOR INTR. FROM DRA OUTPUT
                                  PRE INTERRUPT SETUP
                                  TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT
                                  TEST FOR NO INTR. FROM DRA INPUT LEVEL INDICATED VIA MAINT. INT.
                                  TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN
                                  TEST THAT RESET CLEARS DIGITAL STATUS REGISTER


                                !
                        .TITLE  LPA DR11-K  LOGIC TEST  MD-11-CRLPF-C
                        ;*COPYRIGHT (C) 1980
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY EDWARD C. BADGER MOD. BY R. SHOOP
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                        ;*

 167770                         ABASE=167770
 100300                         AVECT1=100300
                        .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
```

```
(1)             000004                  R4=     %4              ;;GENERAL REGISTER
(1)             000005                  R5=     %5              ;;GENERAL REGISTER
(1)             000006                  R6=     %6              ;;GENERAL REGISTER
(1)             000007                  R7=     %7              ;;GENERAL REGISTER
(1)             000006                  SP=     %6              ;;STACK POINTER
(1)             000007                  PC=     %7              ;;PROGRAM COUNTER
(1)
(1)                                     ;*PRIORITY LEVEL DEFINITIONS
(1)             000000                  PR0=    0               ;;PRIORITY LEVEL 0
(1)             000040                  PP1=    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
1186                                .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            LOOP ON CURRENTLY SELECTED DR11-K
(1)                                 ;*        10            OUTPUT TO INPUT WRAPAROUND CABLE NOT CONNECTED
(1)                                 ;*         9            LOOP ON ERROR
(1)                                 ;*         8            LOOP ON TEST IN SWR<7:0>
1187
1188                                .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  001544                 JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
```

```
1190  000204  000137  001554                  JMP     @#BEGIN1        ;JUMP TO THE RESTART ADDRESS
1191  000210  000137  013476                  JMP     @#EXTTST        ;JUMP TO THE CONTROL LINES LOOP
1192  000214  000137  022200                  JMP     @#$UTK          ;JUMP TO THE USER LINK LOOP
1193                                  .SBTTL  ACT11 HOOKS
 (1)
 (2)                                  ;;******************************************************************
 (1)                                  ;HOOKS REQUIRED BY ACT11
 (1)            000220                        $SVPC=.                 ;SAVE PC
 (1)            000046                        .=46
 (1)  000046    013216                        $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                        .=$SVPC                 ;; RESTORE PC
1194            001000                        .=1000
1195                                  .SBTTL  APT PARAMETER BLOCK
 (1)
 (2)                                  ;;******************************************************************
 (1)                                  ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
 (2)                                  ;;******************************************************************
 (1)            001000                        .$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    001166                $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
 (1)  001004    000030                $TSTM:  .WORD   30      ;;RUN TIM OF LONGEST TEST
 (1)  001006    000010                $PASTM: .WORD   10      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
 (1)  001010    000030                $UNITM: .WORD   30      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
 (1)  001012    00003'                        .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
1196
 (1)
 (2)
 (1)                                .SBTTL  COMMON TAGS
 (1)
 (1)                                ;;**********************************************************************
 (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                $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
 (1)  001162     077                $QUES:  .ASCII  /?/             ;;QUESTION MARK
 (1)  001163     015                $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
 (1)  001164  000012                $LF:    .ASCIZ  <12>            ;;LINE FEED
 (2)                                ;;**********************************************************************
 (2)                                .SBTTL  APT MAILBOX-ETABLE
 (2)
 (3)                                ;;**********************************************************************
 (2)                                .EVEN
 (2)  001166                        $MAIL:                          ;;APT MAILBOX
 (2)  001166  000000                $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
 (2)  001170  000000                $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
 (2)  001172  000000                $TESTN: .WORD   ATESTN          ;;TEST NUMBER
 (2)  001174  000000                $PASS:  .WORD   APASS           ;;PASS COUNT
 (2)  001176  000000                $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
 (2)  001200  000000                $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
 (2)  001202  000000                $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
 (2)  001204  000000                $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
 (2)  001206                        $ETABLE:                        ;;APT ENVIRONMENT TABLE
```

```
(2)  001206     000              $ENV:    .BYTE   AENV     ;;ENVIRONMENT BYTE
(2)  001207     000              $ENVM:   .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
(2)  001210  000000              $SWREG:  .WORD   ASWREG   ;;APT SWITCH REGISTER
(2)  001212  000000              $USWR:   .WORD   AUSWR    ;;USER SWITCHES
(2)  001214  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)  001216     000              $MAMS1:  .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
(2)  001217     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)  001220  000000              $MADR1:  .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
(2)                              ;*                          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2)  001222     000              $MAMS2:  .BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
(2)  001223     000              $MTYP2:  .BYTE   AMTYP2   ;;MEM.TYPE,BLK#2
(2)  001224  000000              $MADR2:  .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
(2)  001226     000              $MAMS3:  .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
(2)  001227     000              $MTYP3:  .BYTE   AMTYP3   ;;MEM.TYPE,BLK#3
(2)  001230  000000              $MADR3:  .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
(2)  001232     000              $MAMS4:  .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
(2)  001233     000              $MTYP4:  .BYTE   AMTYP4   ;;MEM.TYPE,BLK#4
(2)  001234  000000              $MADR4:  .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
(2)  001236  100300              $VECT1:  .WORD   AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001240  000000              $VECT2:  .WORD   AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001242  167770              $BASE:   .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001244  000000              $DEVM:   .WORD   ADEVM    ;;DEVICE MAP
(2)  001246  000000              $CDW1:   .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
(2)  001250                      $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)  001250                  $ERRTB:
1197
1198                            ;ITEM   1
1199  001250  014416                   EM1             ;STATUS REGISTER IN ERROR
1200  001252  015036                   DH1             ;ERRPC  DRADD   STATUS EXPECTED
1201  001254  015326                   DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
1202  001256  015370                   DF0
1203
1204                            ;ITEM   2
1205  001260  014447                   EM2             ;INPUT REGISTER IN ERROR
1206  001262  015106                   DH2             ;ERRPC  DRADD   INPUT EXPECTED
1207  001264  015326                   DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
1208  001266  015370                   DF0
1209
1210                            ;ITEM 3
1211  001270  014477                   EM3             ;OUTPUT REGISTER IN ERROR
1212  001272  015154                   DH3             ;ERRPC  DRADD   OUTPUT EXPECTED
1213  001274  015326                   DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
1214  001276  015370                   DF0
1215
1216                            ;ITEM   4
1217  001300  014530                   EM4             ;INPUT FAILED TO INTERRUPT
1218  001302  015222                   DH4             ;ERRPC  DRADD
1219  001304  015342                   DT4             ;$ERRPC DRADD
1220  001306  015370                   DF0
1221
1222                            ;ITEM   5
1223  001310  014562                   EM5             ;OUTPUT FAILED TO INTERRUPT
1224  001312  015222                   DH4             ;ERRPC  DRADD
1225  001314  015342                   DT4             ;$ERRPC DRADD
1226  001316  015370                   DF0
1227
1228                            ;ITEM   6
1229  001320  014615                   EM6             ;UNEXPECTED INTERRUPT
1230  001322  015222                   DH4             ;ERRPC  DRADD
1231  001324  015342                   DT4             ;$ERRPC DRADD
1232  001326  015370                   DF0
1233
1235                            ;ITEM   7
1236  001330  014642                   EM7             ;OPERATOR INTERVENTION ERROR
1237  001332  015222                   DH4             ;ERRPC  DRADD
1238  001334  015342                   DT4             ;$ERRPC DRADD
```

```
  1239  001336  015370                        DF0
  1240
  1241                                ;ITEM   10
  1242  001340  014676                        EM10            ;INTERRUPT INPUT BIT FAILED TO SET INPUT READY
  1243  001342  015247                        DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
  1244  001344  015352                        DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
  1245  001346  015370                        DF0
  1246
  1247                                ;ITEM   11
  1248  001350  014761                        EM11            ;NON-INTERRUPTING INPUT BIT SET INPUT READY
  1249  001352  015247                        DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
  1250  001354  015352                        DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
  1251  001356  015370                        DF0
  1252                                        ;
   (1)                                        ;ADDRESS OF KMC-11 OF LPA-11    THE ADDR FOR KMAD0 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)  001360                        LPCI:
   (1)  001360  170460                KMAD0:  .WORD   170460          ;BASE KMC ADDR. MAY BE PATCHED BY USER.
   (1)
   (1)  001362                        LPMR:
   (1)  001362  170461                KMAD1:  .WORD   170460+1                ;>DO NOT        <;KMC-CSR ADDR
   (1)  001364                        LPCO:
   (1)  001364  170462                KMAD2:  .WORD   170460+2                ;>PATCH         <;
   (1)  001366                        LPSO:
   (1)  001366  170463                KMAD3:  .WORD   170460+3                ;>THIS AREA     <
   (1)  001370                        LPADL:
   (1)  001370  170464                KMAD4:  .WORD   170460+4                ;
   (1)  001372                        LPADH:
   (1)  001372  170465                KMAD5:  .WORD   170460+5                ;>DO NOT        <
   (1)  001374                        LPMS1:
   (1)  001374  170466                KMAD6:  .WORD   170460+6                ;>PATCH         <
   (1)  001376                        LPMS2:
   (1)  001376  170467                KMAD7:  .WORD   170460+7                ;>THIS AREA     <
   (1)
   (1)  001400  000300                VECTOR: .WORD   AVECT1&777      ;BASE VECTOR OF KMC
   (1)  001402  000304                VECTPS: .WORD   4+AVECT1&777    ;VECOTR ADDR.+2
   (1)
   (1)  001404  000005                VERSN:  .WORD   5               ;CURRENT VERSION NUMBER OF MICROCODE.
   (1)
   (1)  001406  000000                .DVLS:  .WORD   0               ;/DEVICE LIST OF I/O ADDR. DEFINED
   (1)  001410  000020                        .BLKW   16.             ;/BY INIT.
   (1)
  1253
  1254  001450  167770                BASEBA: 167770                  ;STARTING BASE BUS ADDRESS
  1255  001452  000300                BASEIV: 300                     ;STARTING BASE INTERRUPT ADDRESS
  1256  001454  000200                BASEBR: 200
  1257  001456  000240                CPU:    240                     ;1 MS. CPU DELAY FACTOR 240 FOR 11/05
  1258                                                                ;                       620 FOR 11/40
  1259  001460  000000                NMBEXT: 0                       ;ADDITIONAL DR-11-K
  1260  001462  000000                NBEXT:  0
  1261
```

```
1262  001464  167770                    DRADD:  167770                  ;CURRENT DR11-K STARTING ADDRESS
1263  001466  000300                    DRIV:   300                     ;CURRENT DR11-K STARTING INTERRUPT VECTOR
1264
1265
1266  001470  167770                    LOC1:   167770                  ;LOC BOX #1 ADDR
1267  001472  167760                    LOC2:   167760                  ;LOC BOX #2 ADDR
1268  001474  167750                    LOC3:   167750                  ;LOC BOX #3 ADDR
1269  001476  167770                    GRSTAT: 167770          ;DR STATUS
1270  001500  167772                    GRDAI:  167772          ;INPUT REG.
1271  001502  167774                    GRDIO:  167774          ;OUTPUT REG.
1272  001504  167775                    GRBHIO: 167775          ;OUTPUT REG HIGH BYTE
1273  001506  000000                    NOTLCH: 0
1274  001510  000000                    INTBIT: 0
1275  001512  000300                    GRIVA:  300
1276  001514  000302                    GRIVSA: 302
1277  001516  000304                    GRIVB:  304
1278  001520  000306                    GRIVSB: 306
1279  001522  000200                    DIOBRL: 200
1280  001524  000000                    BRLEV1: 0
1281  001526  000000                    BRLEV2: 0
1282  001530  000000                    BRLEV3: 0
1283  001532  000000                    ODDJMP: 0
1284  001534  000000                    MINSIN: 0
1285  001536  000000                    TRANST: 0
1286  001540  000000                    JUMPER: 0       ;1 IF RUNNING H322 JUMPER CONFIG ;2 IF COULTER JUMPER CONFIG
1287  001542  000000                    $TMDAT: 0
1288
1295                                    ;;*******************************************************
1296                                    ;       DIGITAL I-O LOGIC TEST
1297                                    ;*******************************************************
1298  001544  005000            000000  BEGIN:  CLR     R0                      ;CLEAR R0
1299  001546  005037  001540    000000          CLR     JUMPER                  ;INDICATE NORMAL FACTORY BUILD
1300  001552  000402                            BR      RBEG
1301  001554  012700  177777    000000  BEGIN1: MOV     #-1,R0                  ;LOAD R0
1302  001560  000240            000000  RBEG:   NOP
1303                                    .SBTTL  INITIALIZE THE COMMON TAGS
 (1)                                    ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
 (1)  001562  012706  001100            000000          MOV     #$CMTAG,R6      ;,FIRST LOCATION TO BE CLEARED
 (1)  001566  005026                    000000          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
 (1)  001570  022706  001140            000000          CMP     #SWR,R6 ;;DONE?
 (1)  001574  001374                    000000          BNE     .-6             ;;LOOP BACK IF NO
 (1)  001576  012706  001100            000000          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
 (1)                                    ;;INITIALIZE A FEW VECTORS
 (1)  001602  012737  015410  000020    000000          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
 (1)  001610  012737  000340  000022    000000          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
 (1)  001616  012737  015602  000030    000000          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
 (1)  001624  012737  000340  000032    000000          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
 (1)  001632  012737  020332  000034    000000          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
 (1)  001640  012737  000340  000036    000000          MOV     #340,@#TRAPVEC+2;LEVEL 7
 (1)  001646  012737  016356  000024    000000          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
 (1)  001654  012737  000340  000026    000000          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
 (1)  001662  013737  013164  013156    000000          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
 (1)  001670  005037  001160            000000          CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
 (1)  001674  112737  000001  001115    000000          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
 (1)  001702  012737  001702  001106    000000          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
 (1)  001710  012737  001710  001110    000000          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
```

```
   (2)                                  ;;S'ZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   (2)                                  ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
   (2)  001716  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
   (2)  001722  012737  001756  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
   (2)  001730  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
   (2)  001736  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
   (2)  001744  022777  177777  177166          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
   (2)  001752  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
   (2)                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
   (2)  001754  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
   (2)  001756  012716  001764          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
   (2)  001762  000002                          RTI
   (2)  001764  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
   (2)  001772  012737  000174  001142          MOV     #DISPREG,DISPLAY
   (2)  002000  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
   (1)
   (2)  002004  005037  001174                  CLR     $PASS           ;;CLEAR PASS COUNT
   (2)  002010  132737  000200  001207          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
   (2)  002016  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
   (2)  002020  012737  001210  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
   (2)  002026                          67$:
  1304                                          ;
   (1)                                          ;THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
   (1)                                          ;
   (1)
   (1)  002026  010046                          MOV     R0,-(SP)
   (1)  002030  010146                          MOV     R1,-(SP)
   (1)  002032  013700  001360                  MOV     KMAD0,R0        ;GET KMC-11 ADDRESS.
   (1)  002036  012701  001362                  MOV     #KMAD1,R1       ;GET ADDR. OF ADDR. LIST.
   (1)
   (1)  002042  005200                  68$:    INC     R0              ;UPDATE ADDR.
   (1)  002044  010021                          MOV     R0,(1)+         ;WRITE ADDR.
   (1)  002046  020127  001400                  CMP     R1,#KMAD7+2     ;DONE ALL ADDRESSES?
   (1)  002052  001373                          BNE     68$             ;NO - DO NEXT ADDR.
   (1)  002054  005037  001406                  CLR     .DVLS           ;CLR ADDR. LIST.
   (1)  002060  012601                          MOV     (SP)+,R1
   (1)  002062  012600                          MOV     (SP)+,R0
  1305  002064  013737  001242  001450          MOV     $BASE,BASEBA            ;GET APT DEFINED ADDRESS
  1306  002072  013737  001236  001452          MOV     $VECT1,BASEIV           ;GET VECTOR
  1307  002100  042737  160000  001452          BIC     #160000,BASEIV
  1308  002106  113737  001237  001454          MOVB    $VECT1+1,BASEBR         ;GET PRIORITY
  1309  002114  042737  177437  001454          BIC     #177437,BASEBR
  1310  002122  012737  002170  000004          MOV     #1$,@#ERRVEC            ;LOAD BUS ERROR
  1311  002130  013702  001450                  MOV     BASEBA,R2               ;LOAD STARTING ADDRESS
  1312  002134  005003                          CLR     R3                      ;CLEAR COUNT
  1313  002136  010237  001542          2$:     MOV     R2,$TMDAT               ;TEST IF EXISTENT
  1314
   (1)                                  ;*      MOV     @$TMDAT,$GDDAT  ;/READ DEVICE REG $TMDAT,PUT DATA IN $GDDAT.
  1315  002152  005737  021104                  TST     $AERR
  1316  002156  001004                          BNE     1$
  1317  002160  162702  000010                  SUB     #10,R2                  ;EXIST, UPDATE TEST ADDRESS
  1318  002164  005203                          INC     R3                      ;UPDATE # OF DR11'S
  1319  002166  000763                          BR      2$
  1320  002170  005703                  1$:     TST     R3                      ;TEST IF FIRST DOES EXIST
  1321  002172  001014                          BNE     3$                      ;BR
  1322  002174  032777  020000  176736          BIT     #SW13,@SWR              ;TEST IF INHIBIT TYPEOUT
```

```
1323  002202  001002                          BNE     4$                      ;BR IF YES
1324  002204  104401  013660                  TYPE,   BUSTRP                  ;TELL OPERATOR BASE DR11K DOESN'T EXIST
1325  002210  032777  100000  176722  4$:     BIT     #SW15,@SWR              ;TEST HALT ON ERROR
1326  002216  001401                          BEQ     5$                      ;BR IF NO HALT
1327  002220  000000                          HALT                            ;FIRST DR11-K DOES NOT EXIST
1328                                                                          ;CHECK THE PROGRAM DEVICE ADDRESS
1329  002222  000745                  5$:     BR      2$                      ;TRY AGAIN
1330
1331  002224  005303                  3$:     DEC     R3                      ;ADJUST R3
1332  002226  010337  001460                  MOV     R3,NMBEXT               ;SAVE THE NUMBER OF ADDITIONAL DR11-K
1333  002232  012737  002756  000004          MOV     #VECTRP,@#ERRVEC        ;RESET BUS ERROR
1334  002240  012737  000340  000006          MOV     #340,@#ERRVEC+2
1335  002246  004737  022056          RBEG1:  JSR     PC,$RESET
1336  002252  122737  000001  001134          CMPB    #1,$AUTOB               ;TEST IF UNDER A MONITOR
1337  002260  001402                          BEQ     4$                      ;BR IF
1338  002262  005700                          TST     R0                      ;TEST R0
1339  002264  001402                          BEQ     2$                      ;BR IF CLEARED
1340  002266  000137  003266          4$:     JMP     IOTEST                  ;JUMP IF SET
1341  002272                          2$:
 (1)                                .SBTTL  TYPE PROGRAM NAME
 (1)                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
 (1)  002272  005227  177777                INC     #-1             ;;FIRST TIME?
 (1)  002276  001062                        BNE     64$             ;;BRANCH IF NO
 (1)  002300  022737  013216  000042        CMP     #$ENDAD,@#42    ;;ACT-11?
 (1)  002306  001456                        BEQ     64$             ;;BRANCH IF YES
 (1)  002310  104401  002356                TYPE    ,65$            ;;TYPE ASCIZ STRING
 (2)                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
 (2)  002314  005737  000042                TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
 (2)  002320  001012                        BNE     66$             ;;BRANCH IF YES
 (2)  002322  123727  001206  000001        CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
 (2)  002330  001406                        BEQ     66$             ;;BRANCH IF YES
 (2)  002332  023727  001140  000176        CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
 (2)  002340  001005                        BNE     67$             ;;BRANCH IF NO
 (2)  002342  104406                        GTSWR                   ;;GET SOFT-SWR SETTINGS
 (2)  002344  000403                        BR      67$
 (2)  002346  112737  000001  001134  66$:  MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
 (2)  002354                        67$:
 (1)  002354  000433                        BR      64$             ;;GET OVER THE ASCIZ
 (1)                                ;;65$:  .ASCIZ  <CRLF><15><12>"CRLPFC   LPA/DR11-K  INPUT-OUTPUT DIAGNOSTIC  "<15><12><C
 (1)  002444                        64$:
1342  002444  013746  001460                MOV     NMBEXT,-(SP)
1343  002450  104403                        TYPOS
1344  002452  000002                        .WORD   2
1345  002454  104401  002462                TYPE    ,69$            ;;TYPE ASCIZ STRING
 (1)  002460  000422                        BR      68$             ;;GET OVER THE ASCIZ
 (1)                                ;;69$:  .ASCIZ  /(8) ADDITIONAL DR11-K'S CONNECTED/<15><12>
 (1)  002526                        68$:
1346  002526  022737  000001  001540          CMP     #1,JUMPER               ;TEST IF LOC-BOX CONFIG.
1347  002534  001013                          BNE     1$                      ;BR IF NOT
1348  002536  005037  001506                  CLR     NOTLCH                  ;LOAD LOC-BOX JUMPER CONFIG
1349  002542  012737  177777  001510          MOV     #-1,INTBIT              ;
1350  002550  005037  001534                  CLR     MINSIN
1351  002554  005037  001536                  CLR     TRANST
1352  002560  000137  003266                  JMP     IOTEST                  ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
1353  002564  022737  000002  001540  1$:     CMP     #2,JUMPER               ;TEST IF COULTER CONFIG.
1354  002572  001015                          BNE     3$              ;BR IF NOT
```

```
1355  002574  012737  177777  001506          MOV     #-1,NOTLCH              ;LOAD COULTER JUMPER CONFIG.
1356  002602  012737  170000  001510          MOV     #170000,INTBIT
1357  002610  005037  001534                  CLR     MINSIN
1358  002614  012737  170000  001536          MOV     #170000,TRANST
1359  002622  000137  003266                  JMP     IOTEST                  ;RUN THE LOGIC TEST WITH COULTER JUMPERS
1360  002626  105737  001134          3$:     TSTB    $AUTOB                  ;TEST IF RUNNING UNDER A MONITOR
1361  002632  001402                          BEQ     20$                     ;BR IF NOT
1362  002634  000137  003266                  JMP     IOTEST                  ;RUN THE LOGIC TESTS
1363  002640  004537  003202          20$:    JSR     R5,TALK                 ;TALK TO THE ANIMALS
1364  002644  013721                          SWNLB
1365  002646  001506                          NOTLCH                          ;ABOUT NON LATCH INPUT BITS
1366  002650  004537  003202                  JSR     R5,TALK                 ;TALK TO THE ANIMAL AGAIN
1367  002654  014017                          SWINTB
1368  002656  001510                          INTBIT                          ;ABOUT THE INTERRUPTING BITS
1369  002660  004537  003202                  JSR     R5,TALK                 ;ISN'T THIS BOARING
1370  002664  014115                          SWPOSB
1371  002666  001534                          MINSIN                          ;AGAIN-ABOUT THE MINUS INPUT BITS
1372  002670  004537  003202                  JSR     R5,TALK                 ;HO-HUM
1373  002674  014211                          SWTRAB
1374  002676  001536                          TRANST                          ;AGAIN-ABOUT THE TRANSITION BITS
1375
1376                                  ;       DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
1377  002700  012737  010000  001124          MOV     #BIT12,$GDDAT           ;LOAD TEST BIT
1378  002706  033737  001124  001534  10$:     BIT     $GDDAT,MINSIN           ;TEST IF NEG INPUT BIT
1379  002714  001407                          BEQ     11$                     ;BR IF NOT
1380  002716  033737  001124  001536          BIT     $GDDAT,TRANST           ;TEST IF TRANS. INPUT
1381  002724  001403                          BEQ     11$                     ;BR IF NOT
1382  002726  104007                          ERROR   7                       ;LOGICAL OPERATOR ERROR
1383                                                                          ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
1384  002730  000137  001544                  JMP     BEGIN                   ;
1385
1386  002734  006137  001124          11$.    ROL     $GDDAT                  ;MOVE LEFT
1387  002740  103362                          BCC     10$                     ;BR UNTIL DONE
1388
1389  002742  004537  003202                  JSR     R5,TALK                 ;ASK THE OPERATOR
1390  002746  014307                          SWDPOB
1391  002750  001530                          BRLEV3                          ;ABOUT PROGRAM OPTIONS
1392  002752  000137  003266                  JMP     IOTEST
1393
1394                                  ;INTERRUPT TO UNEXPECTED VECTOR
1395  002756  021627  001000          VECTRP: CMP     (SP),#1000              ;TEST IF IN PROGRAM CODE
1396  002762  103402                          BLO     1$                      ;BR IF NOT
1397  002764  000000                  70$:    HALT                            ;FATAL BUS TRAP IN PROGRAM AREA
1398  002766  000776                          BR      70$
1399  002770  021627  000200          1$:     CMP     (SP),#200               ;TEST IF IN SACRED VECTOR AREA
1400  002774  101002                          BHI     2$                      ;BR IF NOT
1401  002776  000000                  71$:    HALT                            ;FATAL VECTOR TRAP
1402  003000  000776                          BR      71$
1403  003002  012637  003200          2$:     MOV     (SP)+,72$               ;GET ADDR THE TRAP OCCURRED TO
1404  003006  162737  000004  003200          SUB     #4,72$                  ;MAKE REAL ADDRESS
1405  003014  032777  020000  176116          BIT     #SW13,@SWR              ;TEST IF TYPE ERROR INHIBIT
1406  003022  001050                          BNE     3$                      ;BR IF INHIBIT
1407  003024  104401  003032                  TYPE    ,65$                ;;TYPE ASCIZ STRING
 (1)  003030  000417                          BR      64$                 ;;GET OVER THE ASCIZ
 (1)                              ;;65$:     .ASCIZ  <15><12>/DR11K INTERRUPTED TO LOC. /
 (1)  003070                      64$.
```

```
                                        MOV     72$,-(SP)               ;LOAD VALUE TO BE TYPED
                                        TYPOC
                                        TYPE    ,67$                ;;TYPE ASCIZ STRING
                                        BR      66$                 ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  / AND WILL NOW USE THAT VECTOR/<15><12>
                                66$:
003144  042737  000007  003200  3$:     BIC     #7,72$                  ;MASK OFF LOWER 3 BITS
003152  032777  100000  175760          BIT     #SW15,@SWR              ;TEST IF HALT ON ERROR
003160  001401                          BEQ     4$                      ;BR IF NOT
003162  000000                          HALT            ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
003164  022626                  4$:     CMP     (SP)+,(SP)+             ;CLEAN THE STACK
003166  013737  003200  001466          MOV     72$,DRIV                ;LOAD NEW VECTOR ADDRESS
003174  000137  003314                  JMP     RBEG2                   ;TEST THE DR11K AT THE NEW VECTOR
003200  000000                  72$:    0

                                ;OPERATOR QUESTIONS AND ANSWER ROUTINE

003202  012537  003214          TALK:   MOV     (R5)+,10$               ;GET ASCII POINTER
003206  012537  003260                  MOV     (R5)+,11$               ;GET POINTER TO DATA FROM OPERATOR
003212  104401                          TYPE
003214  013721                  10$:    SWNLB                           ;TELL OPERATOR TO LOAD SWITCHES
003216  023727  001140  000176          CMP     SWR,#SWREG              ;TEST IF SWITCH REG. EXISTS
003224  001006                          BNE     1$                      ;BR IF NO
003226  104401  020041                  TYPE    ,$MSWR                  ;TYPE "SWR ="
003232  104412                          RDOCT                           ;READ OCTAL
003234  012677  175700                  MOV     (SP)+,@SWR              ;SAVE THE SWITCHES
003240  000403                          BR      2$
003242  104401                  1$:     TYPE
003244  014374                          DEPCNT                          ;TELL OPERATOR TO DEPRESS CONT
003246  000000                          HALT                            ;WAIT FOR OPERATOR
003250  017777  175664  000002  2$:     MOV     @SWR,@11$               ;LOAD SWITCH VALUE
003256  000205                          RTS     R5                      ;EXIT
003260  001506                  11$:    NOTLCH                          ;POINTER TO DATA TO BE LOADED

                                ;UNEXPECTED INTERRUPT
003262  104006                  UNEXPT: ERROR   6                       ;UNEXPECTED INTERRUPT DURING A SUB-TEST
003264  000002                          RTI                             ;EXIT

003266  004737  022056          IOTEST: JSR     PC,$RESET
003272  013737  001450  001464          MOV     BASEBA,DRADD            ;LOAD INITIAL STARTING ADDRESS
003300  013737  001452  001466          MOV     BASEIV,DRIV             ;LOAD INITIAL STARTING VECTOR
003306  013737  001460  001462          MOV     NMBEXT,NBEXT            ;RELOAD # AVAILABLE
003314  004737  022056          RBEG2:  JSR     PC,$RESET
003320  012701  000240                  MOV     #240,R1                 ;LOAD R1
003324  012702  000242                  MOV     #242,R2
003330  010221                  5$:     MOV     R2,(R1)+                ;SAVE THE ADDRESS
003332  012721  004700                  MOV     #4700,(R1)+             ;LOAD "JSR      PC,R0"
003336  022222                          CMP     (R2)+,(R2)+             ;BUMP R2
003340  020227  001076                  CMP     R2,#$CMTAG-2            ;TEST FOR LAST
003344  001371                          BNE     5$                      ;BR UNTIL DONE

003346  004737  022056          IOTST1: JSR     PC,$RESET
003352  004737  003360                  JSR     PC,SETADD               ;SET UP BUS ADDRESS AND VECTOR
003356  000453                          BR      IOTTS1
003360  013737  001464  001476  SETADD: MOV     DRADD,GRSTAT            ;LOAD 1ST ADDRESS
003366  013737  001464  001500          MOV     DRADD,GRDAI             ;LOAD 2ND ADDRESS
```

```
 1461  003374  062737  000002  001500          ADD     #2,GRDAI
 1462  003402  013737  001464  001502          MOV     DRADD,GRDIO             ;LOAD 3RD ADDRESS
 1463  003410  062737  000004  001502          ADD     #4,GRDIO
 1464  003416  013737  001464  001504          MOV     DRADD,GRBHIO            ;LOAD 4TH ADDRESS
 1465  003424  062737  000005  001504          ADD     #5,GRBHIO
 1466  003432  013737  001466  001512          MOV     DRIV,GRIVA              ;LOAD FIRST VECTOR
 1467  003440  013737  001466  001514          MOV     DRIV,GRIVSA
 1468  003446  062737  000002  001514          ADD     #2,GRIVSA
 1469  003454  013737  001466  001516          MOV     DRIV,GRIVB              ;LOAD 2ND VECTOR
 1470  003462  062737  000004  001516          ADD     #4,GRIVB
 1471  003470  013737  001466  001520          MOV     DRIV,GRIVSB
 1472  003476  062737  000006  001520          ADD     #6,GRIVSB
 1473  003504  000207                          RTS     PC
 1474  003506  013737  001454  001522  IOTTS1: MOV     BASEBR,DIOBRL           ;LOAD BR LEVEL
 1475  003514  005037  001532                  CLR     ODDJMP
 1476  003520  053737  001534  001532          BIS     MINSIN,ODDJMP           ;SET MINUS INPUT BITS
 1477  003526  053737  001536  001532          BIS     TRANST,ODDJMP           ;SET TRANSITION INPUT BITS
 1478  003534  012777  003262  175750          MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
  (1)  003542  012777  000340  175744          MOV     #340,@GRIVSA
  (1)  003550  012777  003262  175740          MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
  (1)  003556  012777  000340  175734          MOV     #340,@GRIVSB
 1479                                  ;;*****************************************************************
  (3)                                  ;*TEST 1         TEST FOR NO BUS ERRORS
  (3)                                  ;;*****************************************************************
  (2)  003564  000004                  TST1:   SCOPE
 1480  003566  012737  000001  001172          MOV     #1,$TESTN
 1481  003574  012737  000001  001102          MOV     #1,$TSTNM
 1482  003602  012737  000000  001542          MOV     #0,$TMDAT
 1483
  (1)                                  ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
 1484
  (1)                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1485
  (1)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1486
 1487                                  ;;*****************************************************************
  (3)                                  ;*TEST 2         TEST THAT OUTPUT REG. CAN HOLD #-1
  (3)                                  ;;*****************************************************************
  (2)  003640  000004                  TST2:   SCOPE
 1488  003642  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD EXPECTED
 1489  003650  012737  177777  001542          MOV     #-1,$TMDAT              ;ALL ONES TO REGISTER
 1490
  (1)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1491
  (1)                                  ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
 1492  003676  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1493  003704  001401                          BEQ     TST3                    ;;BR IF EQUAL
 1494  003706  104003                          ERROR   3                       ;REG WILL NOT HOLD ONES
 1495                                  ;;*****************************************************************
  (3)                                  ;*TEST 3         TEST THAT RESET CLEARS OUTPUT REG.
  (3)                                  ;;*****************************************************************
  (2)  003710  000004                  TST3:   SCOPE
 1496  003712  005037  001124                  CLR     $GDDAT
 1497  003716  012737  177777  001542          MOV     #-1,$TMDAT
 1498
  (1)                                  ;*      MOV     $TMDAT,@GRDIO   :/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
```

```
 1499  003734  004737  022056                JSR     PC,$RESET                       ;SET DATA TO ALL ONES
 1500
  (1)                                 ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
 1501  003750  005737  001126                TST     $BDDAT
 1502  003754  001401                        BEQ     TST4            ;;BR IF EQUAL
 1503  003756  104003                        ERROR   3                       ;REG FAILED TO CLEAR
 1504
 1505                                      ;;*****************************************************************
  (3)                                      ;*TEST 4        TEST THAT OUTPUT REG. CAN HOLD #52525
  (3)                                      ;;*****************************************************************
  (2)  003760  000004                      TST4:   SCOPE
 1506  003762  012737  052525  001124              MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
 1507  003770  012737  052525  001542              MOV     #52525,$TMDAT
 1508
  (1)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1509
  (1)                                 ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
 1510  004016  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
 1511  004024  001401                        BEQ     TST5            ;;BR IF EQUAL
 1512  004026  104003                        ERROR   3                       ;DATA NOT=52525
 1513
 1514                                      ;;*****************************************************************
  (3)                                      ;*TEST 5        TEST THAT OUTPUT REG. CAN HOLD #125252
  (3)                                      ;;*****************************************************************
  (2)  004030  000004                      TST5:   SCOPE
 1515  004032  012737  125252  001124              MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
 1516  004040  012737  125252  001542              MOV     #125252,$TMDAT
 1517
  (1)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1518
  (1)                                 ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
 1519  004066  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
 1520  004074  001401                        BEQ     TST6            ;;BR IF EQUAL
 1521  004076  104003                        ERROR   3                       ;DATA NOT=125252
 1522
 1523                                      ;;*****************************************************************
  (3)                                      ;*TEST 6        TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
  (3)                                      ;;*****************************************************************
  (2)  004100  000004                      TST6:   SCOPE
 1524  004102  012737  004114  001110              MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
 1525  004110  005037  001124                      CLR     $GDDAT                  ;CLEAR PATTERN
 1526  004114                           2$:
  (1)
  (1)                                 ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
 1527
  (1)                                 ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
 1528  004134  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
 1529  004142  001401                        BEQ     1$              ;;BR IF EQUAL
 1530  004144  104003                        ERROR   3                       ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
 1531  004146  005237  001124           1$:  INC     $GDDAT                  ;UPDATE THE PATTERN
 1532  004152  001360                        BNE     2$                      ;TRY AGAIN
 1533                                      ;;*****************************************************************
  (3)                                      ;*TEST 7        FLOAT A 1 ACROSS THE OUTPUT REGISTER
  (3)                                      ;;*****************************************************************
  (2)  004154  000004                      TST7:   SCOPE
 1534  004156  012737  004172  001110              MOV     #1$,$LPERR              :LOAD SCOPE ERROR RETURN
```

```
 1535  004164  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE
 1536
 1537  004172  012737  000000  001542  1$:     MOV     #0,$TMDAT               ;CLEAR OUTPUT
 1538
  (1)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1539  004210  053737  001124  001542          BIS     $GDDAT,$TMDAT           ;SET THAT BIT
 1540
  (1)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1541
  (1)                                   ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
 1542  004236  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST RESULTS
 1543  004244  001401                          BEQ     2$                      ;BR IF EQUAL ?
 1544  004246  104003                          ERROR   3
 1545
 1546  004250  006337  001124          2$:     ASL     $GDDAT                  ;SHIFT EXPECTED DATA
 1547  004254  001346                          BNE     1$                      ;BR UNTIL DONE
 1548                                  ;;*******************************************************************
  (3)                                  ;*TEST 10       FLOAT A 0 ACROSS THE OUTPUT REGISTER
  (3)                                  ;;*******************************************************************
  (2)  004256  000004                  TST10:  SCOPE
 1549  004260  012737  004274  001110          MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
 1550  004266  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE
 1551
 1552  004274  012737  177777  001542  1$:     MOV     #-1,$TMDAT              ;LOAD OUTPUT TO A ONE
 1553
  (1)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1554  004312  043737  001124  001542          BIC     $GDDAT,$TMDAT           ;CLEAR A BIT
 1555
  (1)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1556
  (1)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1557  004340  005137  001126                  COM     $BDDAT                  ;COMPLEMENT IT
 1558  004344  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;EQUAL ?
 1559  004352  001401                          BEQ     2$                      ;BR IF EQUAL
 1560  004354  104003                          ERROR   3
 1561
 1562  004356  006337  001124          2$:     ASL     $GDDAT                  ;SHIFT LEFT
 1563  004362  001344                          BNE     1$                      ;BRANCH UNTIL DONE
 1564
 1565                                  ;;*******************************************************************
  (3)                                  ;*TEST 11       TEST FOR SLOW OUTPUT GATES WITH #125252
  (3)                                  ;;*******************************************************************
  (2)  004364  000004                  TST11:  SCOPE
 1566  004366  012737  125252  001124          MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
 1567
  (1)                                   ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
 1572
  (2)                                   ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
  (1)  004414  005137  001542                  COM     $TMDAT
  (2)
  (2)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
  (2)
  (2)                                   ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
  (1)  004440  005137  001542                  COM     $TMDAT
  (2)
  (2)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
```

```
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004464  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004510  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004534  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004560  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004604  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004630  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1573
    (1)                                 ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
1574 004654  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST REGISTER
1575 004662  001401                            BEQ     TST12           ;;BR IF CORRECT
1576 004664  104003                            ERROR   3
1577                                ;;*****************************************************************
    (3)                             ;*TEST 12       TEST FOR SLOW OUTPUT GATES WITH #52525
    (3)                             ;;*****************************************************************
    (2)  004666  000004            TST12:   SCOPE
1578 004670  012737  052525  001124             MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
1579
    (1)                                 ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1584
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE RFG GRDIO,PUT DATA IN $TMDAT.
    (1)  004716  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004742  005137  001542                COM     $TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
    (1)  004766  005137  001542                COM     $TMDAT
    (2)
```

```
   (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
   (2)
   (2)                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
   (1)  005012  005137  001542                  COM     $TMDAT
   (2)
   (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
   (2)
   (2)                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
   (1)  005036  005137  001542                  COM     $TMDAT
   (2)
   (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
   (2)
   (2)                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
   (1)  005062  005137  001542                  COM     $TMDAT
   (2)
   (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
   (2)
   (2)                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
   (1)  005106  005137  001542                  COM     $TMDAT
   (2)
   (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
   (2)
   (2)                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
   (1)  005132  005137  001542                  COM     $TMDAT
   (2)
   (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1585
   (1)                                  ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
 1586  005156  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;TEST PATTERN
 1587  005164  001401                           BEQ     TST13           ;;BR IF EQUAL
 1588  005166  104003                           ERROR   3                       ;OUTPUT REGISTER IN ERROR
 1589
 1590                                  ;;*****************************************************************
   (3)                                 ;*TEST 13       TEST OUTPUT DATA ACCEPT FLAG
   (3)                                 ;;*****************************************************************
   (2)  005170  000004                 TST13:  SCOPE
 1591  005172  012737  000000  001542            MOV     #0,$TMDAT
 1592
   (1)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1593  005210  012737  177777  001542            MOV     #-1,$TMDAT
 1594
   (1)                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1595  005226  012737  100000  001124            MOV     #BIT15,$GDDAT           ;LOAD EXPECTED
 1596
   (1)                                  ;*      MOV     $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
 1597
   (1)                                  ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
 1598  005254  033737  001124  001126            BIT     $GDDAT,$BDDAT
 1599  005262  001001                           BNE     TST14           ;;BR IF SET
 1600  005264  104001                           ERROR   1                       ;ERROR, BIT 15 FAILED TO SET
 1601
 1602                                  ;;*****************************************************************
   (3)                                 ;*TEST 14       TEST OUTPUT INTERRUPT ENABLE
   (3)                                 ;;*****************************************************************
   (2)  005266  000004                 TST14:  SCOPE
 1603  005270  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR STATUS
```

```
 1604
  (1)                                   ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
 1605  005306  012737  040000  001124           MOV     #BIT14,$GDDAT           ;LOAD EXPECTED
 1606
  (1)                                   ;*      MOV     $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
 1607
  (1)                                   ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
 1608  005334  033737  001124  001126           BIT     $GDDAT,$BDDAT
 1609  005342  001001                           BNE     TST15           ;;BR IF SET
 1610  005344  104001                           ERROR   1                       ;ERROR BIT 14 FAILED TO SET
 1611
 1612                                   ;;**********************************************************************
  (3)                                   ;*TEST 15       TEST INPUT DATA READY FLAG
  (3)                                   ;;**********************************************************************
  (2)  005346  000004                   TST15:  SCOPE
 1613  005350  012737  000200  001124           MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
 1614
  (1)                                   ;*      MOV     $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
 1615
  (1)                                   ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
 1616  005376  023737  001124  001126           CMP     $GDDAT,$BDDAT           ;COMPARE
 1617  005404  001401                           BEQ     TST16           ;;BR IF EQUAL
 1618  005406  104001                           ERROR   1                       ;ERROR, BIT 7 FAILED TO SET
 1619
 1620                                   ;;**********************************************************************
  (3)                                   ;*TEST 16       TEST INPUT INTERRUPT ENABLE
  (3)                                   ;;**********************************************************************
  (2)  005410  000004                   TST16:  SCOPE
 1621  005412  012737  000000  001542           MOV     #0,$TMDAT               ;CLEAR STATUS
 1622
  (1)                                   ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
 1623  005430  012737  000100  001124           MOV     #BIT6,$GDDAT            ;LOAD EXPECTED
 1624
  (1)                                   ;*      MOV     $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
 1625
  (1)                                   ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
 1626  005456  023737  001124  001126           CMP     $GDDAT,$BDDAT           ;COMPARE
 1627  005464  001401                           BEQ     TST17           ;;BR IF EQUAL
 1628  005466  104001                           ERROR   1                       ;ERROR, BIT 6 FAILED TO SET
 1629
 1634
 1639
 1640                                   ;;**********************************************************************
  (3)                                   ;*TEST 17       TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
  (3)                                   ;;**********************************************************************
  (2)  005470  000004                   TST17:  SCOPE
 1641  005472  032777  002000  173440           BIT     #BIT10,@SWR             ;TEST SWITCH BIT
 1642  005500  001402                           BEQ     1$                      ;BRANCH IF DOWN
 1643  005502  000137  012146                   JMP     DRT21                   ;BYPASS SOME TEST USING THE EXTERNAL CABLE
 1644  005506                         1$:
  (1)  005506  012737  000000  001542           MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (2)
  (2)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1645  005524  012737  177777  001542           MOV     #-1,$TMDAT
  (2)
  (2)                                   ;*      MOV     $TMDAT,@GRDAT   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAT
```

```
 1646  005542  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
 1647  005546  012737  000000  001542          MOV     #0,$TMDAT               ;LOAD THE OUTPUT
 1648
  (1)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1649
  (1)                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
 1650  005574  043737  001532  001126          BIC     ODDJMP,$BDDAT           ;MASK
 1651  005602  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1652  005610  001401                          BEQ     TST20           ;;BR IF EQUAL
 1653  005612  104002                          ERROR   2                       ;EPROR, INPUT DID NOT EQUAL THE OUTPUT REG.
 1654
 1655                                  ;;**********************************************************************
  (3)                                 ;*TEST 20       TEST INPUT WITH #-1
  (3)                                 ;;**********************************************************************
  (2)  005614  000004                 TST20:  SCOPE
 1656  005616  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (2)
  (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1657  005634  012737  177777  001542          MOV     #-1,$TMDAT
  (2)
  (2)                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1658  005652  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD EXPECTED
 1659  005660  043737  001532  001124          BIC     ODDJMP,$GDDAT           ;MASK
 1660
  (1)                                  ·*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
 1661
  (1)                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
 1662  005706  043737  001532  001126          BIC     ODDJMP,$BDDAT           ;MASK
 1663  005714  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1664  005722  001401                          BEQ     TST21           ;;BR IF EQUAL
 1665  005724  104002                          ERROR   2                       ;ERROR, INPUT DID NOT EQUAL THE OUTPUT
 1666
 1667                                  ;;**********************************************************************
  (3)                                 ;*TEST 21       TEST INPUT WITH #52525
  (3)                                 ;;**********************************************************************
  (2)  005726  000004                 TST21:  SCOPE
 1668  005730  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (2)
  (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1669  005746  012737  177777  001542          MOV     #-1,$TMDAT
  (2)
  (2)                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1670  005764  012737  052525  001124          MOV     #52525,$GDDAT           ;LOAD EXPECTED
 1671  005772  043737  001532  001124          BIC     ODDJMP,$GDDAT           ;MASK
 1672
  (1)                                  ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
 1673
  (1)                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
 1674  006020  043737  001532  001126          BIC     ODDJMP,$BDDAT           ;MASK
 1675  006026  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1676  006034  001401                          BEQ     TST22           ;;BR IF EQUAL
 1677  006036  104002                          ERROR   2                       ;ERROR, INPUT DID NOT EQUAL OUTPUT
 1678
 1679                                  ;;**********************************************************************
  (3)                                 ;*TEST 22       TEST INPUT WITH #125252
  (3)                                 ;;**********************************************************************
```

```
   (2)  006040  000004                    TST22:  SCOPE
 1680   006042  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
   (2)
   (2)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1681   006060  012737  177777  001542            MOV     #-1,$TMDAT
   (2)
   (2)                                     ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1682   006076  012737  125252  001124            MOV     #125252,$GDDAT          ;LOAD EXPECTED
 1683   006104  043737  001532  001124            BIC     ODDJMP,$GDDAT           ;MASK
 1684
   (1)                                     ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
 1685
   (1)                                     ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
 1686   006132  043737  001532  001126            BIC     ODDJMP,$BDDAT           ;MASK
 1687   006140  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
 1688   006146  001401                            BEQ     TST23           ;;BR IF EQUAL
 1689   006150  104002                            ERROR   2                       ;ERROR, INPUT DID NOT EQUAL OUTPUT
 1690                                   ;;*********************************************************************
   (3)                                  ;*TEST 23       TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
   (3)                                  ;;*********************************************************************
   (2)  006152  000004                    TST23:  SCOPE
 1691   006154  005737  001532                    TST     ODDJMP                  ;TEST IF ANY ODD JUMPERS
 1692   006160  001514                            BEQ     TST24           ;;BR IF NONE
 1693   006162  012737  010000  001124            MOV     #BIT12,$GDDAT           ;LOAD TEST BIT
 1694   006170  033737  001532  001124  1$:       BIT     ODDJMP,$GDDAT           ;TEST IF ODD JUMPER BIT
 1695   006176  001502                            BEQ     2$                      ;BR IF NOT
 1696   006200  033737  001506  001124            BIT     NOTLCH,$GDDAT           ;TEST IF LATCHING INPUT BIT
 1697   006206  001076                            BNE     2$                      ;BR IF NOT
 1698
   (1)                                     ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
 1699   006220  012737  177777  001542            MOV     #-1,$TMDAT              ;CLEAR INPUT
 1700
   (1)                                     ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1701
   (1)                                     ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
 1702   006246  043737  001124  001542            BIC     $GDDAT,$TMDAT
 1703
   (1)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1704
   (1)                                     ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
 1705   006274  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
 1706   006302  001401                            BEQ     3$                      ;BR IF EQUAL
 1707   006304  104002                            ERROR   2                       ;ERROR, NEG. INPUT OR NEG. TRANSITION
 1708                                                                              ; INPUT BIT FAILED TO SET INPUT REGISTER
 1709
 1710   006306  033737  001124  001534  3$:       BIT     $GDDAT,MINSIN           ;TEST IF NEG. INPUT BIT
 1711   006314  001033                            BNE     2$                      ;BR IF
 1712   006316  012737  177777  001542            MOV     #-1,$TMDAT              ;CLEAR INPUT
 1713
   (1)                                     ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1714
   (1)                                     ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
 1715   006344  053737  001124  001542            BIS     $GDDAT,$TMDAT
 1716
   (1)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1717
```

```
  (1)                                   ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1718  006372  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1719  006400  001401                           BEQ     2$                      ;BR IF EQUAL
1720  006402  104002                           ERROR   2                       ;ERROR, POSITIVE INPUT TRANSITION
1721                                                                            ;LOGIC FAILED TO SET INPUT REGISTER BIT
1722
1723  006404  006137  001124          2$:     ROL     $GDDAT                  ;CHANGE DATA PATTERN
1724  006410  103267                           BCC     1$                      ;BR IF MORE DATA
1725                                        ;;**********************************************************************
  (3)                                   ;*TEST 24       FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
  (3)                                   ;;**********************************************************************
  (2)   006412  000004                  TST24:  SCOPE
1726  006414  012737  006442  001110            MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
1727
1728  006422  012737  000001  001526            MOV     #BIT0,BRLEV2            ;LOAD EXPECTED
1729  006430  013737  001506  001530            MOV     NOTLCH,BRLEV3           ;GET NON-LATCH
1730  006436  005137  001530                    COM     BRLEV3                  ;COMPLEMENT
1731  006442  013737  001526  001124  2$:      MOV     BRLEV2,$GDDAT           ;LOAD GOOD
1732  006450  033737  001124  001532            BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
1733  006456  001055                           BNE     1$                      ;BYPASS IF ODD JUMPER
1734  006460  033737  001124  001506            BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCH
1735  006466  001451                           BEQ     1$                      ;BR IF LATCHING
1736
1737
  (1)                                   ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1738
  (1)                                   ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1739  006510  043737  001530  001126            BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
1740  006516  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1741  006524  001401                           BEQ     3$              ;;BR IF EQUAL
1742  006526  104002                           ERROR   2                       ;INPUT REGISTER IN ERROR
1743                                                                            ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
1744
1745                                        ;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
1746
1747  006530                                3$:
  (1)
  (1)                                   ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1748  006540  043737  001124  001542            BIC     $GDDAT,$TMDAT
1749
  (1)                                   ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1750
  (1)                                   ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1751  006566  043737  001530  001126            BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
1752  006574  005037  001124                    CLR     $GDDAT                  ;CLEAR EXPECTED
1753  006600  033737  001526  001126            BIT     BRLEV2,$BDDAT           ;TEST FOR BIT
1754  006606  001401                           BEQ     1$              ;;BR IF CLEARED
1755  006610  104002                           ERROR   2                       ;INPUT BIT LATCHED IN ERROR
1756                                                                            ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
1757
1758  006612  006337  001526          1$:     ASL     BRLEV2                  ;CHANGE PATTERN
1759  006616  001311                           BNE     2$                      ;BR UNTIL DONE
1760
1761                                        ;;**********************************************************************
  (3)                                   ;*TEST 25       FLOAT A 1 ACROSS LATCHING INPUT BITS
  (3)                                   ;;**********************************************************************
```

```
  (2)  006620  000004                          TST25:  SCOPE
1762   006622  012737  006672  001110                  MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
1763   006630  012737  000001  001124                  MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE
1764   006636  012737  000000  001542                  MOV     #0,$TMDAT               ;CLEAR OUTPUT REG
1765
  (1)                                          ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1766   006654  012737  177777  001542                  MOV     #-1,$TMDAT              ;CLEAR INPUT
1767
  (1)                                          ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1768
1769   006672  033737  001124  001506  2$:     BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCHING
1770   006700  001030                          BNE     1$                      ;BR IF NON-LATCH
1771   006702  033737  001124  001532          BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
1772   006710  001024                          BNE     1$                      ;BYPASS IF ODD JUMPER
1773
```

```
1775
 (1)                                    ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1776  006722  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1777
 (1)                                    ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1778  006750  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1779  006756  001401                            BEQ     1$              ;;BR IF EQUAL
1780  006760  104002                            ERROR   2               ;INPUT REGISTER FAILED TO LATCH DATA
1781
1782  006762                            1$:
 (1)
 (1)                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1783  006772  053737  001124  001542            BIS     $GDDAT,$TMDAT
1784
 (1)                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1785  007010  006337  001124                    ASL     $GDDAT                  ;CHANGE PATTERN
1786  007014  001326                            BNE     2$                      ;BR UNTIL DONE
1787
1788                                    ;;*******************************************************************
 (3)                                    ;*TEST 26       FLOAT A 0 ACROSS LATCHING INPUT BITS
 (3)                                    ;;*******************************************************************
 (2)  007016  000004                    TST26:  SCOPE
1789  007020  012737  007070  001110            MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
1790  007026  012737  000001  001530            MOV     #BIT0,BRLEV3            ;LOAD EXPECTED
1791  007034  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1792  007052  012737  177777  001542            MOV     #-1,$TMDAT
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1793
1794  007070  033737  001530  001506  2$:       BIT     BRLEV3,NOTLCH           ;TEST FOR LATCHING
1795  007076  001041                            BNE     1$                      ;BR IF NOT
1796  007100  033737  001124  001532            BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
1797  007106  001035                            BNE     1$                      ;BYPASS IF ODD JUMPER BIT
1798
1799  007110  012737  177777  001124            MOV     #-1,$GDDAT              ;LOAD
1800  007116  043737  001506  001124            BIC     NOTLCH,$GDDAT
1801  007124  043737  001530  001124            BIC     BRLEV3,$GDDAT           ;MAKE BRLEV3
1802
 (1)                                    ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1803  007142  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1804
1805
 (1)                                    ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1806  007170  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1807  007176  001401                            BEQ     1$              ;;BR IF EQUAL
1808  007200  104002                            ERROR   2               ;INPUT REGISTER FAILED TO LATCH DATA
1809
1810  007202                            1$:
 (1)
 (1)                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
```

```
 1811  007212  053737  001530  001542          BIS     BRLEV3,$TMDAT
 1812
  (1)                                ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1813  007230  006337  001530                   ASL     BRLEV3                  ;CHANGE PATTERN
 1814  007234  001315                           BNE     2$                      ;BR UNTIL DONE
 1815                                     ;;*****************************************************************
  (3)                                     ;*TEST 27       TEST FOR SLOW INPUT GATES WITH #125252
  (3)                                     ;;*****************************************************************
  (2)  007236  000004                     TST27:  SCOPE
 1816
 1817  007240  012737  125252  001124          MOV     #125252,$GDDAT          ;LOAD EXPECTED
 1818  007246  043737  001506  001124          BIC     NOTLCH,$GDDAT           ;CONVERT
 1819  007254  043737  001532  001124          BIC     ODDJMP,$GDDAT           ;MASK
 1820  007262  013700  001124                  MOV     $GDDAT,R0               ;LOAD PATTERN
 1821  007266  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (2)
  (2)                                ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1822  007304  012737  177777  001542          MOV     #-1,$TMDAT
  (2)
  (2)                                ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1823
 1833
  (2)                                ;*         MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
  (1)  007332  010037  001542                   MOV     R0,$TMDAT
  (2)
  (2)                                ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
  (2)  007346  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (3)
  (3)                                ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
  (2)
  (3)                                ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
  (2)
  (2)                                ;*         MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
  (1)  007374  013701  001542                   MOV     $TMDAT,R1
  (2)
  (2)                                ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
  (1)  007410  005100                           COM     R0
  (2)
  (2)                                ;*         MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
  (1)  007422  010037  001542                   MOV     R0,$TMDAT
  (2)
  (2)                                ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
  (2)  007436  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (3)
  (3)                                ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
  (2)
  (2)                                ;*         MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
  (1)  007464  013701  001542                   MOV     $TMDAT,R1
  (2)
  (2)                                ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
  (1)  007500  005100                           COM     R0
  (2)
  (2)                                ;*         MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
  (1)  007512  010037  001542                   MOV     R0,$TMDAT
  (2)
  (2)                                ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
  (2)  007526  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (3)
```

```
(3)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                     ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE RFG GRDAI,PUT DATA IN $TMDAT.
(1)  007554  013701  001542                     MOV     $TMDAT,R1
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1)  007570  005100                             COM     R0
(2)
(2)                                     ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1)  007602  010037  001542                     MOV     R0,$TMDAT
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)  007616  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
(3)
(3)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                     ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1)  007644  013701  001542                     MOV     $TMDAT,R1
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1)  007660  005100                             COM     R0
(2)
(2)                                     ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1)  007672  010037  001542                     MOV     R0,$TMDAT
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)  007706  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
(3)
(3)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(3)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                     ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1)  007734  013701  001542                     MOV     $TMDAT,R1
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1)  007750  005100                             COM     R0
(2)
(2)                                     ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1)  007762  010037  001542                     MOV     R0,$TMDAT
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)  007776  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
(3)
(3)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                     ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1)  010024  013701  001542                     MOV     $TMDAT,R1
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1)  010040  005100                             COM     R0
(2)
(2)                                     ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1)  010052  010037  001542                     MOV     R0,$TMDAT
(2)
(2)                                     ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)  010066  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
(3)
```

```
    (3)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1) 010114  013701  001542                  MOV     $TMDAT,R1
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1) 010130  005100                          COM     R0
    (2)
    (2)                                 ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
    (1) 010142  010037  001542                  MOV     R0,$TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2) 010156  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1) 010204  013701  001542                  MOV     $TMDAT,R1
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1) 010220  005100                          COM     R0
    (2)
    (2)                                 ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
    (1) 010232  010037  001542                  MOV     R0,$TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2) 010246  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1) 010274  013701  001542                  MOV     $TMDAT,R1
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1) 010310  005100                          COM     R0
    (2)
    (2)                                 ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
    (1) 010322  010037  001542                  MOV     R0,$TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2) 010336  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1) 010364  013701  001542                  MOV     $TMDAT,R1
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1) 010400  005100                          COM     R0
    (2)
    (2)                                 ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
    (1) 010412  010037  001542                  MOV     R0,$TMDAT
    (2)
    (2)                                 ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2) 010426  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
```

```
 (3)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 (2)
 (2)                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
 (1) 010454  013701  001542                     MOV     $TMDAT,R1
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 (1) 010470  005100                             COM     R0
1834
1835 010472  010137  001126                     MOV     R1,$BDDAT               ;LOAD READ
1836 010476  000240                             NOP
1837 010500  000240                             NOP
1838 010502  000240                             NOP
1839 010504  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
1840 010512  001401                             BEQ     TST30           ;;BR IF EQUAL
1841 010514  104002                             ERROR   2                       ;INPUT GATE SLOW
1842
1843                                    ;;*******************************************************************
 (3)                                    ;*TEST 30       TEST FOR SLOW INPUT GATES WITH #52525
 (3)                                    ;;*******************************************************************
 (2) 010516  000004                     TST30:  SCOPE
1844
1845 010520  012737  052525  001124             MOV     #52525,$GDDAT           ;SETUP EXPECTED
1846 010526  043737  001532  001124             BIC     ODDJMP,$GDDAT           ;MASK ODD JUMPER BITS
1847 010534  043737  001506  001124             BIC     NOTLCH,$GDDAT           ;CONVERT
1848 010542  013700  001124                     MOV     $GDDAT,R0
1849 010546  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1850 010564  012737  177777  001542             MOV     #-1,$TMDAT
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1851
1860 010602  010037  001542                     MOV     R0,$TMDAT
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 (2) 010616  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (3)
 (3)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 (2)
 (2)                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
 (1) 010644  013701  001542                     MOV     $TMDAT,R1
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 (1) 010660  005100                             COM     R0
 (1) 010662  010037  001542                     MOV     R0,$TMDAT
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 (2) 010676  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (3)
 (3)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 (2)
 (2)                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
 (1) 010724  013701  001542                     MOV     $TMDAT,R1
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 (1) 010740  005100                             COM     R0
```

```
    (1)  010742  010037  001542                    MOV     R0,$TMDAT
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  010756  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*         MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011004  013701  001542                    MOV     $TMDAT,R1
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011020  005100                            COM     R0
    (1)  011022  010037  001542                    MOV     R0,$TMDAT
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011036  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*         MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011064  013701  001542                    MOV     $TMDAT,R1
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011100  005100                            COM     R0
    (1)  011102  010037  001542                    MOV     R0,$TMDAT
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011116  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*         MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011144  013701  001542                    MOV     $TMDAT,R1
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011160  005100                            COM     R0
    (1)  011162  010037  001542                    MOV     R0,$TMDAT
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011176  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*         MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011224  013701  001542                    MOV     $TMDAT,R1
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011240  005100                            COM     R0
    (1)  011242  010037  001542                    MOV     R0,$TMDAT
    (2)
    (2)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011256  012737  000000  001542            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*         MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*         MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
```

```
    (1)  011304  013701  001542                    MOV     $TMDAT,R1
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011320  005100                             COM     R0
    (1)  011322  010037  001542                     MOV     R0,$TMDAT
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011336  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*          MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011364  013701  001542                     MOV     $TMDAT,R1
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011400  005100                             COM     R0
    (1)  011402  010037  001542                     MOV     R0,$TMDAT
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011416  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*          MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011444  013701  001542                     MOV     $TMDAT,R1
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011460  005100                             COM     R0
    (1)  011462  010037  001542                     MOV     R0,$TMDAT
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011476  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*          MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011524  013701  001542                     MOV     $TMDAT,R1
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011540  005100                             COM     R0
    (1)  011542  010037  001542                     MOV     R0,$TMDAT
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)  011556  012737  000000  001542             MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
    (3)
    (3)                                 ;*          MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
    (2)
    (2)                                 ;*          MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
    (1)  011604  013701  001542                     MOV     $TMDAT,R1
    (2)
    (2)                                 ;*          MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
    (1)  011620  005100                             COM     R0
1861
1862  011622  010137  001126                        MOV     R1,$BDDAT               ;LOAD VALUE READ
1863  011626  000240                                NOP
1864  011630  000240                                NOP
```

```
 1865  011632  000240                          NOP
 1866  011634  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1867  011642  001401                          BEQ     TST31           ;;BR IF EQUAL
 1868  011644  104002                          ERROR   2
 1869                                  ;;*****************************************************************
  (3)                                  ;*TEST 31       TEST THAT RESET CLEARS INPUT REGISTER BITS
  (3)                                  ;;*****************************************************************
  (2)  011646  000004                  TST31:  SCOPE
 1870  011650  012737  000000  001542          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
  (2)
  (2)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1871  011666  012737  177777  001542          MOV     #-1,$TMDAT
  (2)
  (2)                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
 1872  011704  012737  177777  001542          MOV     #-1,$TMDAT              ;LOAD OUTPUT
 1873
  (1)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1874  011722  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED
 1875  011726  004737  022056                  JSR     PC,$RESET
 1876
  (1)                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
 1877  011742  043737  001532  001126          BIC     ODDJMP,$BDDAT           ;MASK ODD JUMPERS
 1878  011750  005737  001126                  TST     $BDDAT
 1879  011754  001401                          BEQ     TST32           ;;BR IF ALL BITS CLEARED
 1880  011756  104002                          ERROR   2                       ;INPUT REG. FAILED TO CLEAR UPON RESET INST.
 1881
 1882                                  ;;*****************************************************************
  (3)                                  ;*TEST 32       TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
  (3)                                  ;;*****************************************************************
  (2)  011760  000004                  TST32:  SCOPE
 1883  011762  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
 1884  011770  012737  000000  001542          MOV     #0,$TMDAT
 1885
  (1)                                  ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
 1886
  (1)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1887  012016  022727  000000  000000          CMP     #0,#0                   ;DELAY
 1888
  (1)                                  ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
 1889  012034  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1890  012042  001401                          BEQ     TST33           ;;BR IF EQUAL
 1891  012044  104001                          ERROR   1                       ;INPUT DATA READY FLAG FAILED TO SET
 1892
 1893                                  ;TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
 1894
 1895                                  ;;*****************************************************************
  (3)                                  ;*TEST 33       TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
  (3)                                  ;;*****************************************************************
  (2)  012046  000004                  TST33:  SCOPE
 1896  012050  012737  100000  001124          MOV     #BIT15,$GDDAT           ;LOAD EXPECTED
 1897  012056  012737  000000  001542          MOV     #0,$TMDAT
 1898
  (1)                                  ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
 1899
  (1)                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
 1900  012104  022727  000000  000000          CMP     #0,#0
```

```
1901
 (1)                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1902  012122  013700  001542                      MOV     $TMDAT,R0
1903
 (1)                                    ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
1904  012136  005737  001126                      TST     $BDDAT
1905  012142  100401                              BMI     TST34           ;;BR IF SET
1906  012144  104001                              ERROR   1                       ;INPUT DATA READY FLAG FAILED TO SET
1907
1908
1909  012146                                DRT21:
 (4)                                    ;;**********************************************************************
 (3)                                    ;*TEST 34       TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
 (3)                                    ;;**********************************************************************
 (2)  012146  000004                    TST34:  SCOPE
1910  012150  032777  002000  166762              BIT     #BIT10,@SWR             ;TEST CABLE SWITCH
1911  012156  001172                              BNE     TST35           ;;BYPASS IF NO I/O CABLE
1912
1913  012160  012737  012174  001110              MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
1914  012166  012737  000001  001530              MOV     #BIT0,BRLEV3            ;LOAD INTERRUPT BIT
1915  012174  005037  001126          1$:         CLR     $BDDAT                  ;CLEAR BAD DATA
1916  012200  012737  000200  001124              MOV     #BIT7,$GDDAT            ;LOAD GOOD DATA
1917
1918  012206  033737  001530  001510              BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
1919  012214  001550                              BEQ     3$              ;;NO TRY NEXT BIT
1920  012216  012737  000000  001542              MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1921  012234  012737  177777  001542              MOV     #-1,$TMDAT
 (2)
 (2)                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1922  012252  012737  000000  001542              MOV     #0,$TMDAT               ;CLEAR STATUS
1923
 (1)                                    ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1924
 (1)                                    ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1925  012300  053737  001530  001542              BIS     BRLEV3,$TMDAT
1926
 (1)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1927
 (1)                                    ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1928  012326  043737  001530  001542              BIC     BRLEV3,$TMDAT
1929
 (1)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1930
 (1)                                    ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1931  012354  053737  001530  001542              BIS     BRLEV3,$TMDAT
1932
 (1)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1933
1934  012372  012737  000000  001542              MOV     #0,$TMDAT               ;CLEAR STATUS
1935
 (1)                                    ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1936
 (1)                                    ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1937                                                                            ;IF INTERRUPT INPUT SWITCH IS ON
```

```
                                        ;*      MOV     @GRSTAT,$TMDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
012430  105737  001542                        TSTB    $TMDAT
012434  100401                                BMI     2$              ;;BR IF SET
012436  104010                                ERROR   10                      ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
                                                                                ;?? DID OPERATOR GIVE CORRECT
                                                                                ;INPUT INTERRUPT BITS ??
012440                                  2$:
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
012450  043737  001530  001542                  BIC     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
012476  053737  001530  001542                  BIS     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
012514  005037  001124                        CLR     $GDDAT                  ;CLEAR EXPECTED
012520  005077  166752                        CLR     @GRSTAT                 ;CLEAR STATUS
012524  117737  166746  001126                MOVB    @GRSTAT,$BDDAT          ;READ STATUS
012532  100001                                BPL     3$              ;;BR IF CLEARED
012534  104001                                ERROR   1                       ;INPUT READY FAILED TO CLEAR
012536  006337  001530                3$:     ASL     BRLEV3                  ;CHANGE BIT
012542  001214                                BNE     1$                      ;BR IF NOT DONE
                                        ;;**************************************************************
                                        ;*TEST 35       TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
                                        ;;**************************************************************
012544  000004                        TST35:  SCOPE
012546  032777  002000  166364                BIT     #BIT10,@SWR             ;TEST CABLE SWITCH
012554  001127                                BNE     TST36           ;;BYPASS IF NO I/O CABLE
012556  012737  012572  001110                MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
012564  012737  000001  001530                MOV     #BIT0,BRLEV3            ;LOAD NON-INTERRUPT BIT
012572  012737  000200  001126          1$:   MOV     #200,$BDDAT             ;LOAD BAD DATA
012600  005037  001124                        CLR     $GDDAT                  ;CLEAR GOOD DATA
012604  033737  001530  001510                BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
012612  001105                                BNE     3$              ;;YES SKIP AND TRY NEXT BIT
012614  012737  000000  001542                MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
012632  012737  177777  001542                MOV     #-1,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
012650  012737  000000  001542                MOV     #0,$TMDAT               ;CLEAR STATUS
                                        ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
012676  053737  001530  001542                  BIS     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
012714  043737  001530  001542                  BIC     BRLEV3,$TMDAT           ;CLEAR OUTPUT
```

```
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
001530  001542                         ;       BIS     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
000000  001542                         ;       MOV     #0,$TMDAT               ;CLEAR FLAG FROM DATA READY
                                        ;*      MOV     $TMDAT,@        ;/ PUT DATA FROM $TMDAT TO DEVICE REG
                                                                                ;SHOULD REMAIN SET VIA DIRECT SET SIDE
                                        ;*      MOV     @GRSTAT,$TMDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
105737  001542                         ;       TSTB    $TMDAT
100001                                  ;       BPL     3$              ;;BR IF CLEAR
104011                                  ;       ERROR   11                      ;INPUT NON-INTERRUPT BIT SET INPUT READY
                                                                                ;?? DID OPERATOR GIVE CORRECT
                                                                                ;INPUT INTERRUPT BITS ??
006337  001530                         ;3$:    ASL     BRLEV3                  ;CHANGE BIT
001257                                  ;       BNE     1$                      ;BR IF NOT DONE
                                        ;;*****************************************************************
                                        ;*TEST 36       DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
                                        ;;*****************************************************************
000004                                  ;TST36: SCOPE
005737  001462                         ;BYPASS: TST    NBEXT                   ;TEST IF ANY
001415                                  ;       BEQ     1$                      ;BR IF NONE
032777  010000  166066                  ;       BIT     #SW12,@SWR              ;TEST BIT12 OF SWR
001024                                  ;       BNE     BYPAS1                  ;INHIBIT TESTING NEXT DR11-K
162737  000010  001464                  ;       SUB     #10,DRADD               ;UPDATE DEVICE ADDRESS
062737  000010  001466                  ;       ADD     #10,DRIV                ;UPDATE DEVICE VECTOR
005337  001462                         ;       DEC     NBEXT                   ;ANOTHER ONE ?
000413                                  ;       BR      BYPAS1                  ;BR IF ANOTHER
013737  001450  001464                  ;1$:    MOV     BASEBA,DRADD            ;RELOAD ADDRESS
013737  001452  001466                  ;       MOV     BASEIV,DRIV             ;RELOAD VECTOR
013737  001460  001462                  ;       MOV     NMBEXT,NBEXT            ;RELOAD NUMBER
000137  013134                         ;       JMP     $EOP                    ;DONE
012700  177777                         ;BYPAS1: MOV    #-1,R0
000137  003314                         ;       JMP     RBEG2                   ;TEST ANOTHER UNIT

                                        .SBTTL  END OF PASS ROUTINE

                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO BYPAS1

013134                                  $EOP:
000004                                          SCOPE
005037  001102                                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
005237  001174                                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
```

```
    (1)  013146  042737  100000  001174          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
    (1)  013154  005327                           DEC     (PC)+           ;;LOOP?
    (1)  013156  000001                  $EOPCT:  .WORD   1
    (1)  013160  003022                           BGT     $DOAGN          ;;YES
    (1)  013162  012737                           MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
    (1)  013164  000001                  $ENDCT:  .WORD   1
    (1)  013166  013156                           $EOPCT
    (1)  013170  104401  013235                   TYPE    ,$ENDMG         ;;TYPE "END PASS #"
    (2)  013174  013746  001174                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
    (2)  013200  104405                           TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
    (1)  013202  104401  013232                   TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
    (1)  013206  013700  000042          $GET42:  MOV     @#42,R0         ;;GET MONITOR ADDRESS
    (1)  013212  001405                           BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
    (1)  013214  000005                           RESET                   ;;CLEAR THE WORLD
    (1)  013216  004710                  $ENDAD:  JSR     PC,(R0)         ;;GO TO MONITOR
    (1)  013220  000240                           NOP                     ;;SAVE ROOM
    (1)  013222  000240                           NOP                     ;;FOR
    (1)  013224  000240                           NOP                     ;;ACT11
    (1)  013226                          $DOAGN:
    (1)  013226  000137                           JMP     @(PC)+          ;;RETURN
    (1)  013230  013124                  $RTNAD:  .WORD   BYPAS1
    (1)  013232     377     377     000  $ENULL:  .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
    (1)  013235     015  042412  042116  $ENDMG:  .ASCIZ  <15><12>/END PASS #/
    (1)  013242  050040  051501  020123
    (1)  013250  000043
 2013                                     .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)  013252                          $TYPDS:
    (3)  013252  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
    (3)  013254  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
    (3)  013256  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
    (3)  013260  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
    (3)  013262  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
    (1)  013264  012746  020200                   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
    (1)  013270  016605  000020                   MOV     20(SP),R5       ;;GET THE INPUT NUMBER
    (1)  013274  100004                           BPL     1$              ;;BR IF INPUT IS POS.
    (1)  013276  005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
    (1)  013300  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
    (1)  013306  005000                  1$:      CLR     R0              ;;ZERO THE CONSTANTS INDEX
    (1)  013310  012703  013466                   MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
    (1)  013314  112723  000040                   MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
    (1)  013320  005002                  2$:      CLR     R2              ;;CLEAR THE BCD NUMBER
    (1)  013322  016001  013456                   MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
    (1)  013326  160105                  3$:      SUB     R1,R5           ;;FORM THIS BCD DIGIT
    (1)  013330  002402                           BLT     4$              ;;BR IF DONE
    (1)  013332  005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
```

```
(1)  013334  000774                        BR      3$
(1)  013336  060105                 4$:    ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  013340  005702                         TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  013342  001002                         BNE     5$              ;;FALL THROUGH IF 0
(1)  013344  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  013346  100407                         BMI     7$              ;;BR IF YES
(1)  013350  106316                 5$:     ASLB    (SP)            ;;MSD?
(1)  013352  103003                         BCC     6$              ;;BR IF NO
(1)  013354  116663  000001 177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  013362  052702  000060         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  013366  052702  000040         7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  013372  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  013374  005720                         TST     (R0)+           ;;JUST INCREMENTING
(1)  013376  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
(1)  013402  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
(1)  013404  003002                         BGT     8$              ;;GO TO EXIT
(1)  013406  010502                         MOV     R5,R2           ;;GET THE LSD
(1)  013410  000764                         BR      6$              ;;GO CHANGE TO ASCII
(1)  013412  105726                 8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  013414  100003                         BPL     9$              ;;BR IF NO
(1)  013416  116663  177777 177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  013424  105013                 9$:     CLRB    (R3)            ;;SET THE TERMINATOR
(3)  013426  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  013430  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  013432  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  013434  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  013436  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  013440  104401  013466                 TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  013444  016666  000002 000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  013452  012616                         MOV     (SP)+,(SP)
(1)  013454  000002                         RTI                     ;;RETURN TO USER
(1)  013456  023420                 $DTBL:  10000.
(1)  013460  001750                         1000.
(1)  013462  000144                         100.
(1)  013464  000012                         10.
(1)  013466  000004                 $DBLK:  .BLKW   4
2014
2015                                ;       MISC. EXTERNAL LOGIC TEST
2016  013476  012706  001100       EXTTST: MOV     #STACK,SP
2017  013502  004737  003360               JSR     PC,SETADD               ;SET UP ADDRESS AND VECTOR
2018  013506  012737  040000 013656 2$:     MOV     #BIT14,EXTCNT           ;LOAD COUNT
2019  013514                       1$:
 (1)  013514  012737  000000 001542         MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (2)
 (2)                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
2020  013532  012737  125252 001542         MOV     #125252,$TMDAT          ;LOAD OUTPUT
2021
 (1)                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
2022
 (1)                                ;*      MOV     @GRDAI,EXTTMP   ;/READ DEVICE REG GRDAI,PUT DATA IN EXTTMP.
2023
 (1)                                ;*      MOV     EXTTMP,@GRDAI   ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
2024  013570  012737  000000 001542         MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
 (2)
 (2)                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
2025  013606  012737  052525 001542         MOV     #52525,$TMDAT           ;LOAD OUTPUT
```

```
2026
 (1)                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
2027
 (1)                                    ;*      MOV     @GRDAI,EXTTMP   ;/READ DEVICE REG GRDAI,PUT DATA IN EXTTMP.
2028
 (1)                                    ;*      MOV     EXTTMP,@GRDAI   ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
2029  013644  005337  013656                       DEC     EXTCNT                  ,FINISHED COUNT
2030  013650  001321                               BNE     1$
2031  013652  000715                               BR      2$                      ;LOOP
2032
2033  013654  000000                       EXTTMP: 0
2034  013656  000000                       EXTCNT: 0
2035
2036
2037  013660  005015  052502  020123      BUSTRP: .ASCIZ  <15><12>/BUS TIME-OUT ON SELECTED DR11K/
      013666  044524  042515  047455
      013674  052125  047440  020116
      013702  042523  042514  052103
      013710  042105  042040  030522
      013716  045461     000
2038  013721     123  052105  051440      SWNLB:  .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING INPUT BITS/
      013726  044527  041524  020110
      013734  042522  044507  052123
      013742  051105  041040  052111
      013750  020123  050505  040525
      013756  020114  047524  052040
      013764  042510  047040  047117
      013772  046055  052101  044103
      014000  047111  020107  047111
      014006  052520  020124  044502
      014014  051524     000
2039  014017     123  052105  051440      SWINTB: .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING INPUT BITS/
      014024  044527  041524  020110
      014032  042522  044507  052123
      014040  051105  041040  052111
      014046  020123  050505  040525
      014054  020114  047524  052040
      014062  042510  044440  052116
      014070  051105  052522  052120
      014076  047111  020107  047111
      014104  052520  020124  044502
      014112  051524     000
2040  014115     123  052105  051440      SWPOSB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO POSITIVE INPUT BITS/
      014122  044527  041524  020110
      014130  042522  044507  052123
      014136  051105  041040  052111
      014144  020123  032461  030455
      014152  020062  050505  040525
      014160  020114  047524  050040
      014166  051517  052111  053111
      014174  020105  047111  052520
      014202  020124  044502  051524
      014210     000
2041  014211     123  052105  051440      SWTRAB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO TRANSITION INPUT BITS/
      014216  044527  041524  020110
      014224  042522  044507  052123
```

```
                014232  051105  041040  052111
                014240  020123  032461  030455
                014246  020062  050505  040525
                014254  020114  047524  052040
                014262  040522  051516  052111
                014270  047511  020116  047111
                014276  052520  020124  044502
                014304  051524     000
  2042          014307     123  052105  051440  SWDPOB: .ASCIZ  /SET SWITCH REGISTER WITH THE DESIRED PROGRAM OPTIONS/
                014314  044527  041524  020110
                014322  042522  044507  052123
                014330  051105  053440  052111
                014336  020110  044124  020105
                014344  042504  044523  042522
                014352  020104  051120  043517
                014360  040522  020115  050117
                014366  044524  047117  000123
  2043          014374  005015  042504  051120  DEPCNT: .ASCIZ  <15><12>/DEPRESS CONT./<15><12>
                014402  051505  020123  047503
                014410  052116  006456  000012
  2044          014416  052123  052101  051525  EM1:    .ASCIZ  /STATUS REGISTER IN ERROR/
                014424  051040  043505  051511
                014432  042524  020122  047111
                014440  042440  051122  051117
                014446     000
  2045          014447     111  050116  052125  EM2:    .ASCIZ  /INPUT REGISTER IN ERROR/
                014454  051040  043505  051511
                014462  042524  020122  047111
                014470  042440  051122  051117
                014476     000
  2046          014477     117  052125  052520  EM3:    .ASCIZ  /OUTPUT REGISTER IN ERROR/
                014504  020124  042522  044507
                014512  052123  051105  044440
                014520  020116  051105  047522
                014526  000122
  2047          014530  047111  052520  020124  EM4:    .ASCIZ  /INPUT FAILED TO INTERRUPT/
                014536  040506  046111  042105
                014544  052040  020117  047111
                014552  042524  051122  050125
                014560  000124
  2048          014562  052517  050124  052125  EM5:    .ASCIZ  /OUTPUT FAILED TO INTERRUPT/
                014570  043040  044501  042514
                014576  020104  047524  044440
                014604  052116  051105  052522
                014612  052120     000
  2049          014615     125  042516  050130  EM6:    .ASCIZ  /UNEXPECTED INTERRUPT/
                014622  041505  042524  020104
                014630  047111  042524  051122
                014636  050125  000124
  2050          014642  050117  051105  052101  EM7:    .ASCIZ  /OPERATOR INTERVENTION ERROR/
                014650  051117  044440  052116
                014656  051105  042526  052116
                014664  047511  020116  051105
                014672  047522  000122
  2051          014676  047111  042524  051122  EM10:   .ASCIZ  /INTERRUPT INPUT BIT FAILED TO SET INPUT READY FLAG/
                014704  050125  020124  047111
```

```
                014712  052520  020124  044502
                014720  020124  040506  046111
                014726  042105  052040  020117
                014734  042523  020124  047111
                014742  052520  020124  042522
                014750  042101  020131  046106
                014756  043501     000
2052            014761     116  047117  044455  EM11:   .ASCIZ  /NON-INTERRUPT INPUT BIT SET INPUT READY FLAG/
                014766  052116  051105  052522
                014774  052120  044440  050116
                015002  052125  041040  052111
                015010  051440  052105  044440
                015016  050116  052125  051040
                015024  040505  054504  043040
                015032  040514  000107
2053            015036  051105  050122  020103  DH1:    .ASCIZ  /ERRPC   DRADD  TSTNUM   STATUS  EXPECTED/
                015044  020040  051104  042101
                015052  004504  051524  047124
                015060  046525  020040  051440
                015066  040524  052524  020123
                015074  042440  050130  041505
                015102  042524  000104
2054            015106  051105  050122  020103  DH2:    .ASCIZ  /ERRPC   DRADD  TSTNUM  INPUT   EXPECTED/
                015114  020040  051104  042101
                015122  004504  051524  047124
                015130  046525  044411  050116
                015136  052125  020040  042440
                015144  050130  041505  042524
                015152  000104
2055            015154  051105  050122  020103  DH3:    .ASCIZ  /ERRPC   DRADD  TSTNUM  OUTPUT  EXPECTED/
                015162  020040  051104  042101
                015170  004504  051524  047124
                015176  046525  047411  052125
                015204  052520  020124  042440
                015212  050130  041505  042524
                015220  000104
2056            015222  051105  050122  020103  DH4:    .ASCIZ  /ERRPC   DRADD  TSTNUM/
                015230  020040  051104  042101
                015236  004504  051524  047124
                015244  046525     000
2057            015247     105  051122  041520  DH10:   .ASCIZ  /ERRPC   DRADD  TSTNUM  STATUS  EXPECT  INPUT BIT/
                015254  020040  042040  040522
                015262  042104  052011  052123
                015270  052516  004515  052123
                015276  052101  051525  020040
                015304  054105  042520  052103
                015312  020040  047111  052520
                015320  020124  044502  000124
2058                                            .EVEN
2059            015326  001116  001464  015406  DT1:    $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,0
                015334  001126  001124  000000
2060            015342  001116  001464  015406  DT4:    $ERRPC,DRADD,TSTNUM,0
                015350  000000
2061            015352  001116  001464  015406  DT10:   $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,BRLEV3,0
                015360  001126  001124  001530
                015366  000000
```

```
2062  015370  000000  000000  000000  DF0:    0,0,0,0,0,0,0
      015376  000000  000000  000000
      015404  000000
2063  015406  000000                  TSTNUM: 0
2064
2065                                  .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)                                  ;*SW09=1        LOOP ON ERROR
 (1)                                  ;*SW08=1        LOOP ON TEST IN SWR<7:0>
 (1)                                  ;*CALL
 (1)                                  ;*      SCOPE           ;;SCOPE=IOT
 (1)
 (1)  015410                          $SCOPE:
 (1)  015410  104407                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
 (1)  015412  032777  040000  163520  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 (1)  015420  001062                          BNE     $OVER           ;;YES IF SW14=1
 (1)                                  ;#####START OF CODE FOR THE XOR TESTER#####
 (1)  015422  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
!(1)                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 (1)  015424  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 (1)  015430  012737  015450  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 (1)  015436  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
 (1)  015442  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 (1)  015446  000431                          BR      $SVLAD          ;;GO TO THE NEXT TEST
 (1)  015450  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 (1)  015452  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 (1)  015456  000417                          BR      7$              ;;LOOP ON THE PRESENT TEST
 (1)  015460                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 (1)  015460  032777  000400  163452          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 (1)  015466  001404                          BEQ     2$              ;;BR IF NO
 (1)  015470  127737  163444  001102          CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
 (1)  015476  001433                          BEQ     $OVER           ;;BR IF YES
 (1)  015500  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 (1)  015504  001412                          BEQ     $SVLAD          ;;BR IF NO
 (1)  015506  032777  001000  163424          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 (1)  015514  001404                          BEQ     4$              ;;BR IF NO
 (1)  015516  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 (1)  015524  000420                          BR      $OVER
 (1)  015526  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 (1)  015532  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 (1)  015536  113737  001102  001172          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 (1)  015544  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 (1)  015550  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 (1)  015554  005037  001160                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 (1)  015560  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 (1)  015566  013777  001102  163346  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 (1)  015574  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 (1)  015600  000002                          RTI                     ;;FIXES PS
2066                                  .SBTTL  ERROR HANDLER ROUTINE
 (1)
```

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

015602                                  $ERROR:
015602  104407                                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
015604  113737  001102  015406                  MOVB    $TSTNM,TSTNUM
015612  105237  001103                  7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
015616  001775                                  BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
015620  013777  001102  163314                  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
015626  005237  001112                          INC     $ERTTL                  ;;INC THE ERROR COUNT
015632  011637  001116                          MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
015636  162737  000002  001116                  SUB     #2,$ERRPC
015644  117737  163246  001114                  MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
015652  032777  020000  163260                  BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
015660  001004                                  BNE     20$                     ;;SKIP TYPEOUTS
015662  004737  015774                          JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
015666  104401  001163                          TYPE    ,$CRLF
015672                                  20$:
015672  122737  000001  001206                  CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
015700  001007                                  BNE     2$                      ;;NO,SKIP APT ERROR REPORT
015702  113737  001114  015714                  MOVB    $ITEMB,21$              ;;SET ITEM NUMBER AS ERROR NUMBER
015710  004737  020102                          JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
015714     000                          21$:    .BYTE   0
015715     000                                  .BYTE   0
015716  000777                          22$:    BR      22$                     ;;APT ERROR LOOP
015720  005777  163214                  2$:     TST     @SWR                    ;;HALT ON ERROR
015724  100002                                  BPL     3$                      ;;SKIP IF CONTINUE
015726  000000                                  HALT                            ;;HALT ON ERROR!
015730  104407                                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
015732  032777  001000  163200          3$:     BIT     #BIT09,@SWR             ;;LOOP ON ERROR SWITCH SET?
015740  001402                                  BEQ     4$                      ;;BR IF NO
015742  013716  001110                          MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
015746  005737  001160                  4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
015752  001402                                  BEQ     5$                      ;;BR IF NONE
015754  013716  001160                          MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
015760                                  5$:
015760  022737  013216  000042                  CMP     #$ENDAD,@#42            ;;ACT-11 AUTO-ACCEPT?
015766  001001                                  BNE     6$                      ;;BRANCH IF NO
015770  000000                                  HALT                            ;;YES
015772                                  6$:
015772  000002                                  RTI                             ;;RETURN

                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                        ;;**********************************************************************
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
```

```
(1)
(1)  015774                          $ERRTYP:
(1)  015774  104401  001163                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016000  010046                           MOV     R0,-(SP)        ;;SAVE R0
(1)  016002  005000                           CLR     R0              ;;PICKUP THE ITEM INDEX
(1)  016004  153700  001114                   BISB    @#$ITEMB,R0
(1)  016010  001004                           BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                                   ;;TYPE THE PC OF THE ERROR
(2)  016012  013746  001116                   MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                                   ;;ERROR ADDRESS
(2)  016016  104402                           TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016020  000426                           BR      6$              ;;GET OUT
(1)  016022  005300                   1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1)  016024  006300                           ASL     R0              ;;      WORK FOR THE ERROR TABLE
(1)  016026  006300                           ASL     R0
(1)  016030  006300                           ASL     R0
(1)  016032  062700  001250                   ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(1)  016036  012037  016046                   MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
(1)  016042  001404                           BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1)  016044  104401                           TYPE                    ;;TYPE THE "ERROR MESSAGE"
(1)  016046  000000                   2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
(1)  016050  104401  001163                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016054  012037  016064           3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
(1)  016060  001404                           BEQ     5$              ;;SKIP TYPEOUT IF 0
(1)  016062  104401                           TYPE                    ;;TYPE THE "DATA HEADER"
(1)  016064  000000                   4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
(1)  016066  104401  001163                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016072  011000                   5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
(1)  016074  001004                           BNE     7$              ;;GO TYPE THE DATA
(1)  016076  012600                   6$:     MOV     (SP)+,R0        ;;RESTORE R0
(1)  016100  104401  001163                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED'
(1)  016104  000207                           RTS     PC              ;;RETURN
(1)  016106                          7$:
(2)  016106  013046                           MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
(2)  016110  104402                           TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016112  005710                           TST     (R0)            ;;IS THERE ANOTHER NUMBER?
(1)  016114  001770                           BEQ     6$              ;;BR IF NO
(1)  016116  104401  016124                   TYPE    ,8$             ;;TYPE TWO(2) SPACES
(1)  016122  000771                           BR      7$              ;;LOOP
(1)  016124  020040      000         8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
(1)          016130                           .EVEN
2069
2070                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
 (1)
 (2)                                  ;;**********************************************************************
 (1)                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
 (1)                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
 (1)                                  ;*$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)  016130  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  016134  116637  000001  016353         MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  016142  112637  016355                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  016146  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  016152  000406                          BR      $TYPON
(1)  016154  112737  000001  016353  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  016162  112737  000006  016355         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  016170  112737  000005  016352  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  016176  010346                          MOV     R3,-(SP)        ;;SAVE R3
(1)  016200  010446                          MOV     R4,-(SP)        ;;SAVE R4
(1)  016202  010546                          MOV     R5,-(SP)        ;;SAVE R5
(1)  016204  113704  016355                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  016210  005404                          NEG     R4
(1)  016212  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  016216  110437  016354                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  016222  113704  016353                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  016226  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  016232  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  016234  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)  016236  000404                          BR      3$              ;;GO DO MSB
(1)  016240  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  016242  006105                          ROL     R5
(1)  016244  006105                          ROL     R5
(1)  016246  010503                          MOV     R5,R3
(1)  016250  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  016252  105337  016354                  DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  016256  100016                          BPL     7$              ;;BR IF NO
(1)  016260  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
(1)  016264  001002                          BNE     4$              ;;TEST FOR 0
(1)  016266  005704                          TST     R4              ;;SUPPRESS THIS 0?
(1)  016270  001403                          BEQ     5$              ;;BR IF YES
(1)  016272  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  016274  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  016300  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  016304  110337  016350                  MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  016310  104401  016350                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  016314  105337  016352          7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  016320  003347                          BGT     2$              ;;BR IF MORE TO DO
(1)  016322  002402                          BLT     6$              ;;BR IF DONE
(1)  016324  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  016326  000744                          BR      2$              ;;GO DO THE LAST DIGIT
(1)  016330  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  016332  012604                          MOV     (SP)+,R4        ;;RESTORE R4
(1)  016334  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  016336  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
```

```
(1)  016344  012616                        MOV     (SP)+,(SP)
(1)  016346  000002                        RTI                         ;;RETURN
(1)  016350     000                8$:     .BYTE   0                   ;;STORAGE FOR ASCII DIGIT
(1)  016351     000                        .BYTE   0                   ;;TERMINATOR FOR TYPE ROUTINE
(1)  016352     000                $OCNT:  .BYTE   0                   ;;OCTAL DIGIT COUNTER
(1)  016353     000                $OFILL: .BYTE   0                   ;;ZERO FILL SWITCH
(1)  016354  000000                $OMODE: .WORD   0                   ;;NUMBER OF DIGITS TO TYPE
2071
2072                                       .SBTTL  POWER DOWN AND UP ROUTINES
(1)
(2)                                        ;;**********************************************************************
(1)                                        ;POWER DOWN ROUTINE
(1)  016356  012737  016522  000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1)  016364  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
(3)  016372  010046                        MOV     R0,-(SP)            ;;PUSH R0 ON STACK
(3)  016374  010146                        MOV     R1,-(SP)            ;;PUSH R1 ON STACK
(3)  016376  010246                        MOV     R2,-(SP)            ;;PUSH R2 ON STACK
(3)  016400  010346                        MOV     R3,-(SP)            ;;PUSH R3 ON STACK
(3)  016402  010446                        MOV     R4,-(SP)            ;;PUSH R4 ON STACK
(3)  016404  010546                        MOV     R5,-(SP)            ;;PUSH R5 ON STACK
(3)  016406  017746  162526                MOV     @SWR,-(SP)          ;;PUSH @SWR ON STACK
(1)  016412  010637  016526                MOV     SP,$SAVR6           ;;SAVE SP
(1)  016416  012737  016430  000024        MOV     #$PWRUP,@#PWRVEC    ;;SET UP VECTOR
(1)  016424  000000                        HALT
(1)  016426  000776                        BR      .-2                 ;;HANG UP
(1)
(2)                                        ;;**********************************************************************
(1)                                        ;POWER UP ROUTINE
(1)  016430  012737  016522  000024 $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1)  016436  013706  016526                MOV     $SAVR6,SP           ;;GET SP
(1)  016442  005037  016526                CLR     $SAVR6              ;;WAIT LOOP FOR THE TTY
(1)  016446  005237  016526          1$:   INC     $SAVR6              ;;WAIT FOR THE INC
(1)  016452  001375                        BNE     1$                  ;;OF  WORD
(3)  016454  012677  162460                MOV     (SP)+,@SWR          ;;POP STACK INTO @SWR
(3)  016460  012605                        MOV     (SP)+,R5            ;;POP STACK INTO R5
(3)  016462  012604                        MOV     (SP)+,R4            ;;POP STACK INTO R4
(3)  016464  012603                        MOV     (SP)+,R3            ;;POP STACK INTO R3
(3)  016466  012602                        MOV     (SP)+,R2            ;;POP STACK INTO R2
(3)  016470  012601                        MOV     (SP)+,R1            ;;POP STACK INTO R1
(3)  016472  012600                        MOV     (SP)+,R0            ;;POP STACK INTO R0
(1)  016474  012737  016356  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1)  016502  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
(1)  016510  104401                        TYPE                        ;;REPORT THE POWER FAILURE
(1)  016512  016530                $PWRMG: .WORD   PWRMSG              ;;POWER FAIL MESSAGE POINTER
(1)  016514  012716                        MOV     (PC)+,(SP)          ;;RESTART AT BEGIN1
(1)  016516  001554                $PWRAD: .WORD   BEGIN1              ;;RESTART ADDRESS
(1)  016520  000002                        RTI
(1)  016522  000000                $ILLUP: HALT                        ;;THE POWER UP SEQUENCE WAS STARTED
(1)  016524  000776                        BR      .-2                 ;; BEFORE THE POWER DOWN WAS COMPLETE
(1)  016526  000000                $SAVR6: 0                           ;;PUT THE SP HERE
2073 016530  005015  042522  052123 PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
     016536  051101  044524  043516
     016544  040440  052106  051105
     016552  040440  050040  053517
     016560  051105  043040  044501
     016566  052514  042522  005015
```

```
      016574  000
              016576                        .EVEN
                                    .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) 016576 105737 001157           $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
 (1) 016602 100002                          BPL     1$              ;;BR IF YES
 (1) 016604 000000                          HALT                    ;;HALT HERE IF NO TERMINAL
 (1) 016606 000430                          BR      3$              ;;LEAVE
 (1) 016610 010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
 (1) 016612 017600 000002                   MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
 (1) 016616 122737 000001 001206            CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 (1) 016624 001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
 (1) 016626 132737 000100 001207            BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
 (1) 016634 001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
 (1) 016636 010037 016646                   MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
 (1) 016642 004737 020072                   JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
 (1) 016646 000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
 (1) 016650 132737 000040 001207    62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
 (1) 016656 001003                          BNE     60$             ;;YES,SKIP TYPE OUT
 (1) 016660 112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 (1) 016662 001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
 (1) 016664 005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
 (1) 016666 012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
 (1) 016670 062716 000002           3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
 (1) 016674 000002                          RTI                     ;;RETURN
 (1) 016676 122716 000011           4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
 (1) 016702 001430                          BEQ     8$
 (1) 016704 122716 000200                   CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
 (1) 016710 001006                          BNE     5$
 (1) 016712 005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
 (1) 016714 104401                          TYPE                    ;;TYPE A CR AND LF
 (1) 016716 001163                          $CRLF
 (1) 016720 105037 017054                   CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
 (1) 016724 000755                          BR      2$              ;;GET NEXT CHARACTER
 (1) 016726 004737 017010           5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
 (1) 016732 123726 001156           6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
 (1) 016736 001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
 (1) 016740 013746 001154                   MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
 (1)                                                                ;;AND THE NULL CHAR.
 (1) 016744 105366 000001           7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
 (1) 016750 002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

```
(1)  016752  004737  017010            JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  016756  105337  017054            DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  016762  000770                    BR      7$              ;;LOOP
(1)
(1)                                    ;HORIZONTAL TAB PROCESSOR
(1)
(1)  016764  112716  000040            8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  016770  004737  017010            9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  016774  132737  000007  017054            BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  017002  001372                            BNE     9$              ;;TAB STOP
(1)  017004  005726                            TST     (SP)+           ;;POP SPACE OFF STACK
(1)  017006  000724                            BR      2$              ;;GET NEXT CHARACTER
(1)  017010  105777  162134            $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  017014  100375                            BPL     $TYPEC
(1)  017016  116677  000002  162126            MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  017024  122766  000015  000002            CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  017032  001003                            BNE     1$              ;;BRANCH IF NO
(1)  017034  105037  017054                    CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  017040  000406                            BR      $TYPEX          ;;EXIT
(1)  017042  122766  000012  000002  1$:      CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  017050  001402                            BEQ     $TYPEX          ;;BRANCH IF YES
(1)  017052  105227                            INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  017054  000000                    $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  017056  000207                    $TYPEX: RTS     PC
(1)
2076                                   .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)  017060  011646                    $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1)  017062  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
(3)  017070  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  017072  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  017074  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(1)  017076  104411                    1$:     RDLIN                   ;;READ AN ASCIZ LINE
(1)  017100  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
(1)  017102  005001                            CLR     R1              ;;CLEAR DATA WORD
(1)  017104  005002                            CLR     R2
(1)  017106  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
(1)  017110  001412                            BEQ     3$              ;;IF ZERO GET OUT
(1)  017112  006301                            ASL     R1              ;;*2
(1)  017114  006102                            ROL     R2
(1)  017116  006301                            ASL     R1              ;;*4
(1)  017120  006102                            ROL     R2
(1)  017122  006301                            ASL     R1              ;;*8
(1)  017124  006102                            ROL     R2
(1)  017126  042716  177770                    BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
(1)  017132  062601                            ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
(1)  017134  000764                            BR      2$              ;;LOOP
(1)  017136  005726                    3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
```

```
(1)  017140  010166  000012                MOV     R1,12(SP)       ;;SAVE THE RESULT
(1)  017144  010237  017160                MOV     R2,$HIOCT
(3)  017150  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  017152  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  017154  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  017156  000002                        RTI                     ;;RETURN
(1)  017160  000000                $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
2077                               .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)  017162  022737  000176  001140 $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
(1)  017170  001074                        BNE     15$             ;;BRANCH IF NO
(1)  017172  105777  161746                TSTB    @$TKS           ;;CHAR THERE?
(1)  017176  100071                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
(1)  017200  117746  161742                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
(1)  017204  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
(1)  017210  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
(1)  017214  001062                        BNE     15$             ;;NO, RETURN TO USER
(1)  017216  123727  001134  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  017224  001456                        BEQ     15$             ;;BRANCH IF YES
(1)
(1)  017226  104401  020034                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  017232  104401  020041        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  017236  013746  000176                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  017242  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  017244  104401  020052                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  017250  005046                19$:    CLR     -(SP)           ;;CLEAR COUNTER
(1)  017252  005046                        CLR     -(SP)           ;;THE NEW SWR
(1)  017254  105777  161664        7$:     TSTB    @$TKS           ;;CHAR THERE?
(1)  017260  100375                        BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  017262  117746  161660                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  017266  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1)  017272  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  017276  001005                        BNE     10$             ;;BRANCH IF NOT
(1)  017300  104401  020027                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  017304  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  017310  000757                        BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  017312  021627  000015        10$:    CMP     (SP),#15        ;;IS IT A <CR>?
(1)  017316  001022                        BNE     16$             ;;BRANCH IF NO
(1)  017320  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  017324  001403                        BEQ     11$             ;;BRANCH IF YES
(1)  017326  016677  000002  161604        MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  017334  062706  000006        11$:    ADD     #6,SP           ;;CLEAR UP STACK
```

```
(1)  017340  104401  001163          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  017344  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  017352  001003                          BNE     15$             ;;BRANCH IF NOT
(1)  017354  012777  000100  161562          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  017362  000002                  15$:    RTI                     ;;RETURN
(1)  017364  004737  017010          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  017370  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
(1)  017374  002420                          BLT     18$             ;;BRANCH IF YES
(1)  017376  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
(1)  017402  003015                          BGT     18$             ;;BRANCH IF YES
(1)  017404  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  017410  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  017414  001403                          BEQ     17$             ;;BRANCH IF YES
(1)  017416  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  017420  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
(1)  017422  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
(1)  017424  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  017430  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  017434  000707                          BR      7$              ;;GET THE NEXT ONE
(1)  017436  104401  001162          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  017442  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)  017444  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  017446  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  017454  105777  161464          1$:     TSTB    @$TKS           ;;WAIT FOR
(1)  017460  100375                          BPL     1$              ;;A CHARACTER
(1)  017462  117766  161460  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  017470  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  017476  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  017504  001013                          BNE     3$              ;;BRANCH IF NO
(1)  017506  105777  161432          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  017512  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  017514  117746  161426                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  017520  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  017524  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  017530  001366                          BNE     2$              ;;IF NOT DISCARD IT
(1)  017532  000750                          BR      1$              ;;YES, RESUME
(1)  017534  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  017542  002407                          BLT     4$              ;;BRANCH IF YES
(1)  017544  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  017552  003003                          BGT     4$              ;;BRANCH IF YES
(1)  017554  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
(1)  017562  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)  017564  010346                     $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  017566  005046                             CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  017570  012703  020020             1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  017574  022703  020027             2$:     CMP     #$TTYIN+7,R3    ;;BUFFER FULL?
(1)  017600  101456                             BLOS    4$              ;;BR IF YES
(1)  017602  104410                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  017604  112613                             MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  017606  122713  000177             10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  017612  001022                             BNE     5$              ;;BR IF NO
(1)  017614  005716                             TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  017616  001007                             BNE     6$              ;;BR IF NO
(1)  017620  112737  000134  020016             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  017626  104401  020016                     TYPE    ,9$
(1)  017632  012716  177777                     MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  017636  005303                     6$:     DEC     R3              ;;BACKUP BY ONE
(1)  017640  020327  020020                     CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  017644  103434                             BLO     4$              ;;BR IF YES
(1)  017646  111337  020016                     MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  017652  104401  020016                     TYPE    ,9$             ;;GO TYPE
(1)  017656  000746                             BR      2$              ;;GO READ ANOTHER CHAR.
(1)  017660  005716                     5$:     TST     (SP)            ;;RUBOUT KEY SET?
(1)  017662  001406                             BEQ     7$              ;;BR IF NO
(1)  017664  112737  000134  020016             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  017672  104401  020016                     TYPE    ,9$
(1)  017676  005016                             CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  017700  122713  000025             7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  017704  001003                             BNE     8$              ;;BR IF NO
(1)  017706  104401  020027                     TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
(1)  017712  000726                             BR      1$              ;;GO START OVER
(1)  017714  122713  000022             8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "^R"?
(1)  017720  001011                             BNE     3$              ;;BRANCH IF NO
(1)  017722  105013                             CLRB    (R3)            ;;CLEAR THE CHARACTER
(1)  017724  104401  001163                     TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
(1)  017730  104401  020020                     TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  017734  000717                             BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  017736  104401  001162             4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  017742  000712                             BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  017744  111337  020016             3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  017750  104401  020016                     TYPE    ,9$
(1)  017754  122723  000015                     CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  017760  001305                             BNE     2$              ;;LOOP IF NOT RETURN
(1)  017762  105063  177777                     CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  017766  104401  001164                     TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  017772  005726                             TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  017774  012603                             MOV     (SP)+,R3        ;;RESTORE R3
(1)  017776  011646                             MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  020000  016666  000004  000002             MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  020006  012766  020020  000004             MOV     #$TTYIN,4(SP)
(1)  020014  000002                             RTI                     ;;RETURN
(1)  020016     000             9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  020017     000                     .BYTE   0                       ;;TERMINATOR
(1)  020020  000007             $TTYIN: .BLKB   7                       ;;RESERVE 7 BYTES FOR TTY INPUT
```

```
(1)  020027  136    006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>     ;;CONTROL 'U'
(1)  020034  043536 005015  000     $CNTLG: .ASCIZ  /^G/<15><12>     ;;CONTROL 'G'
(1)  020041  015    051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  020046  036440 000040
(1)  020052  020040 042516  020127  $MNEW:  .ASCIZ  /  NEW = /
(1)  020060  020075 000
(1)         020064                  .EVEN
2078                                .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                 ;;*****************************************************************
(1)  020064  112737 000001  020330  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  020072  112737 000001  020326  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  020100  000403                 BR      $ATYC
(1)  020102  112737 000001  020330  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  020110                         $ATYC:
(3)  020110  010046                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  020112  010146                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  020114  105737 020326          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  020120  001450                 BEQ     5$              ;;IF NOT:  BR
(1)  020122  122737 000001  001206  CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  020130  001031                 BNE     3$              ;;IF NOT:  BR
(1)  020132  132737 000100  001207  BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  020140  001425                 BEQ     3$              ;;IF NOT:  BR
(1)  020142  017600 000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  020146  062766 000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  020154  005737 001166  1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  020160  001375                 BNE     1$              ;;IF NOT:  WAIT
(1)  020162  010037 001202          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  020166  105720                 TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  020170  001376                 BNE     2$
(1)  020172  163700 001202          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  020176  006200                 ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  020200  010037 001204          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  020204  012737 000004  001166  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  020212  000413                 BR      5$
(1)  020214  017637 000004  020240  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  020222  062766 000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  020230  013746 177776          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  020234  004737 016576          JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  020240  000000                 4$:     .WORD   0
(1)  020242                         5$:
(1)  020242  105737 020330  10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  020246  001416                 BEQ     12$             ;;IF NOT:  BR
(1)  020250  005737 001206          TST     $ENV            ;;RUNNING UNDER APT?
(1)  020254  001413                 BEQ     12$             ;;IF NOT:  BR
(1)  020256  005737 001166  11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  020262  001375                 BNE     11$             ;;IF NOT:  WAIT
(1)  020264  017637 000004  001170  MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  020272  062766 000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  020300  005237 001166          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  020304  105037 020330  12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  020310  105037 020327          CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  020314  105037 020326          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  020320  012601                 MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  020322  012600                 MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  020324  000207                 RTS     PC              ;;RETURN
```

```
(1)  020326  000                  $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  020327  000                  $LFLG:  .BYTE   0               ;;LOG FLAG
(1)  020330  000                  $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          020332                       .EVEN
(1)          000200               APTSIZE=200
(1)          000001               APTENV=001
(1)          000100               APTSPOOL=100
(1)          000040               APTCSUP=040
2079                              .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)  020332  010046               $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
(1)  020334  016600  000002              MOV     2(SP),R0        ;;GET TRAP ADDRESS
(1)  020340  005740                      TST     -(R0)           ;;BACKUP BY 2
(1)  020342  111000                      MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
(1)  020344  006300                      ASL     R0              ;;POSITION FOR INDEXING
(1)  020346  016000  020366              MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
(1)  020352  000200                      RTS     R0              ;;GO TO ROUTINE
(1)
(1)
(1)                               ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1)  020354  011646               $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
(1)  020356  016666  000004  000002      MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
(1)  020364  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)  020366  020354               $TRPAD: .WORD   $TRAP2
(3)  020370  016576                       $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3)  020372  016154                       $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)  020374  016130                       $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)  020376  016170                       $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)  020400  013252                       $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
(1)
(3)  020402  017232                       $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
(1)
(3)  020404  017162                       $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
(3)  020406  017444                       $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3)  020410  017564                       $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3)  020412  017060                       $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
2080 020414  000000               BUFFER: 0                       ;10 WORD BUFFER FOR THE COULTER INTERFACE TEST
2081
(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)  020416                             $LPAI:
(2)  020416  013746  000004                     MOV     4,-(SP)
(2)
(2)  020422  000413                             BR      31$             ;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)  020424  012737  020450  000004             MOV     #30$,4
(2)  020432  005237  170000                     INC     170000
(3)  020436  104401  020444                     TYPE    ,65$            ;;TYPE ASCIZ STRING
(3)  020442  000401                             BR      64$             ;;GET OVER THE ASCIZ
(3)                                     ;;65$:  .ASCIZ  <7>##
(3)  020446                             64$:
(2)  020446  000401                             BR      31$
(2)  020450  022626                     30$:    CMP     (SP)+,(SP)+
(2)  020452  012637  000004             31$:    MOV     (SP)+,4         ;ALL THIS JUNK MUST BE REMOVED!!
(2)  020456  005037  021104                     CLR     $AERR
(2)  020462  004537  021106                     JSR     R5,$LOAD        ;LOAD MICRO-CODE.
(2)  020466  000000G                            .WORD   DRLPX2          ;FILE ''DRLPX2.OBJ''
(2)
(2)  020470  052777  040000  160662             BIS     #BIT14,@KMADO   ;ISSUE KMC+DMC INIT.
(2)
(2)  020476                             1$:
(2)                                                                     ;''HANGS'' HERE THEN KMC-11 ERROR.
(2)  020476  010146                             MOV     R1,-(SP)
(2)  020500  005001                             CLR     R1
(2)  020502  005201                     2$:     INC     R1              ;STALL FOR DMC-UP
(2)  020504  001376                             BNE     2$
(2)  020506  012777  104000  160644             MOV     #BIT15!BIT11,@KMADO     ;SET RUN, AND ENABLE ARBITRATION.
(2)  020514  105201                     25$:    INCB    R1
(2)  020516  001376                             BNE     25$
(2)
(2)  020520  032777  000040  160632             BIT     #BIT5,@KMADO    ;SLAVE READY? (READING IPBM SR)
(2)  020526  001401                             BEQ     3$
```

```
(2)                                                                     ;FATAL LPA-11 ERROR SLAVE NOT READY.
(2)  020530  104000                          ERROR
(2)
(2)  020532  012777  000004  160624  3$:     MOV     #4,@KMAD2          ;READ FAST PATH
(2)  020540                                 4$:
(3)  020540  004537  022016                  JSR     R5, $TOUT          ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  020544  104000                          ERROR                      ;/TIME-OUT ERROR
(3)                                                                     ;/WE FAILED TO COMPLETE
(3)                                                                     ;/CURRENT OPERATION.
(3)                                                                     ;/CONTINUES IN THIS LOOP
(3)                                                                     ;/WOULD MAKE US ''HANG'' HERE
(3)
(3)  020546  000774                          BR              4$
(3)
(3)                                                                     ;/RETURNS HERE-FROM-TIMED OUT.
(2)  020550  122777  000377  160606          CMPB    #377,@KMAD2        ;WAIT TILL KMC DONE COMMAND.
(2)  020556  001370                          BNE     4$
(2)  020560  122777  000377  160602          CMPB    #377,@KMAD4        ;IF FAST PATH=377 THEN ERROR.
(2)  020566  001001                          BNE     35$
(2)  020570  104000                          ERROR                      ;IPBM ERROR (SLAVE SIDE)
(2)                                                                     ;YOU MUST RUN IPBM DIAGNOSTIC.
(2)
(2)  020572  117737  160572  021052  35$:    MOVB    @KMAD4,11$         ;GET THE VERSION NUMBER FROM DMC-11
(2)  020600  005227  177777                  INC     #-1
(2)  020604  001045                          BNE     5$
(2)  020606  005227  177777                  INC     #-1
(2)  020612  001042                          BNE     5$
(3)  020614  104401  020622                  TYPE    ,67$               ;;TYPE ASCIZ STRING
(3)  020620  000426                          BR      66$                ;;GET OVER THE ASCIZ
(3)                                     ;;67$:  .ASCIZ  <200>''M8200-YC (DMC) MICROCODE VERSION NUMBER = ''
(3)  020676                             66$:
(2)  020676  013746  021052                  MOV     11$,-(SP)
(2)  020702  104403                          TYPOS
(2)  020704     002     000                  .BYTE   2,0
(3)  020706  104401  020714                  TYPE    ,69$               ;;TYPE ASCIZ STRING
(3)  020712  000402                          BR      68$                ;;GET OVER THE ASCIZ
(3)                                     ;;69$:  .ASCIZ  <200>''  ''
(3)  020720                             68$:
(2)
(2)  020720  112737  177777  021052  5$:     MOVB    #0-1,11$           ;DAC CODE FOR SLAVE.
(2)  020726  012501                          MOV     (5)+,R1            ;GET NEXT DEVICE ADDR.
(2)  020730  021127  000000          6$:     CMP     (R1),#0            ;TERM REACHED?
(2)  020734  001444                          BEQ     10$
(2)  020736  105237  021052                  INCB    11$
(2)  020742  113777  021052  160420          MOVB    11$,@KMAD4         ;FIFO DATA
(2)  020750  004737  021054                  JSR     PC,20$             ;ISSUE SEND
(2)  020754  112177  160410                  MOVB    (R1)+,@KMAD4       ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
(2)  020760  004737  021054                  JSR     PC,20$             ;ISSUE SEND
(2)  020764  112177  160400                  MOVB    (R1)+,@KMAD4       ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
(2)  020770  004737  021054                  JSR     PC,20$
(2)
(2)  020774  032777  000002  160356  7$:     BIT     #BIT1,@KMAD0       ;WAIT FOR FIFO DATA
(2)  021002  001374                          BNE     7$                 ;=1 NO DATA. =0 DATA.
(2)  021004  112777  000002  160352          MOVB    #2,@KMAD2          ;READ FIFO.
(2)
```

```
(2)  021012                          8$:
(3)  021012  004537  022016                 JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  021016  104000                         ERROR                   ;/TIME-OUT ERROR
(3)                                                                 ;/WE FAILED TO COMPLETE
(3)                                                                 ;/CURRENT OPERATION.
(3)                                                                 ;/CONTINUES IN THIS LOOP
(3)                                                                 ;/WOULD MAKE US "HANG" HERE
(3)
(3)  021020  000774                         BR              8$
(3)
(3)                                                                 ;/RETURNS HERE-FROM-TIMED OUT.
(2)  021022  122777  000377  160334         CMPB    #377,@KMAD2     ;WAIT FOR READ.
(2)  021030  001370                         BNE     8$
(2)  021032  105777  160332                 TSTB    @KMAD4          ;WAS A ZERO RETURNED?
(2)  021036  001734                         BEQ     6$              ;YES GET NEXT ADDR.
(2)                                                                 ;SLAVE WILL RETURN CODE 0 IF
(2)  021040  005237  021104                 INC     $AERR           ;DEV PRESENT.   ELSE
(2)                                                                 ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
(2)  021044  005041                         CLR     -(1)            ;GET RID OF REFERENCE TO BAD ADDR.
(2)  021046  012601                  10$:   MOV     (SP)+,R1
(2)  021050  000205                         RTS     R5              ;RETURN ALL ADDR. CHECKED.
(2)
(2)  021052  000000                  11$:   .WORD   0               ;HOLDS DAC CODE PLUS OFFSET
(2)                                                                 ;TO SLAVES ADDR. TABLE.
(2)
(2)  021054  112777  000003  160302  20$:   MOVB    #3,@KMAD2       ;ISSUE FIFO WRITE
(2)  021062                          21$:
(3)  021062  004537  022016                 JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  021066  104000                         ERROR                   ;/TIME-OUT ERROR
(3)                                                                 ;/WE FAILED TO COMPLETE
(3)                                                                 ;/CURRENT OPERATION.
(3)                                                                 ;/CONTINUES IN THIS LOOP
(3)                                                                 ;/WOULD MAKE US "HANG" HERE
(3)
(3)  021070  000774                         BR              21$
(3)
(3)                                                                 ;/RETURNS HERE-FROM-TIMED OUT.
(2)  021072  122777  000377  160264         CMPB    #377,@KMAD2     ;KMC CODE WILL RETURN A "377"
(2)  021100  001370                         BNE     21$             ;WHEN DONE COMMAND.
(2)  021102  000207                         RTS     PC
(2)
(2)  021104  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)  021106  010446                  $LOAD: MOV     R4,-(SP)        ;SAVE R4.
(2)  021110  010046                         MOV     R0,-(SP)        ;SAVE R0.
(2)  021112  012500                  1$:    MOV     (5)+,R0         ;GET PROG. ADDR.
```

```
(2)  021114  005077  160240                    CLR     @KMAD0          ;CLEAR CSR
(2)  021120  005077  160244                    CLR     @KMAD4          ;CLEAR CRAM ADDR.
(2)  021124  052777  002000  160226  2$:       BIS     #2000,@KMAD0    ;SELECT CRAM.
(2)  021132  012077  160236                    MOV     (0)+,@KMAD6     ;WRITE DATA.
(2)  021136  052777  020000  160214            BIS     #20000,@KMAD0   ;SET CRAM WRITE
(2)  021144  005077  160210                    CLR     @KMAD0          ;DISABLE CRAM.
(2)  021150  005277  160214                    INC     @KMAD4          ;UPDATE CRAM ADDR.
(2)  021154  021027  177777                    CMP     (0),#-1         ;ALL DONE?
(2)  021160  001361                            BNE     2$              ;NO LOOP.
(2)  021162  005077  160202                    CLR     @KMAD4          ;CLEAR CRAM ADDR.
(2)  021166  016500  177776                    MOV     -2(5),R0        ;GET MICRO CODE ADDR.
(2)
(2)  021172  052777  002000  160160  3$:       BIS     #2000,@KMAD0    ;SELECT CRAM
(2)  021200  022077  160170                    CMP     (R0)+,@KMAD6    ;DATA OK?
(2)  021204  001013                            BNE     5$              ;NO - REPORT AN ERROR.
(2)  021206  021027  177777                    CMP     (0),#-1         ;ALL DONE?
(2)  021212  001405                            BEQ     4$              ;YES - EXIT
(2)  021214  005077  160140                    CLR     @KMAD0          ;NO - DESELECT CRAM.
(2)  021220  005277  160144                    INC     @KMAD4          ;UPDATE CRAM ADDR.
(2)  021224  000762                            BR      3$
(2)
(2)  021226  012600                    4$:     MOV     (SP)+,R0        ;RESTORE R0
(2)  021230  012604                            MOV     (SP)+,R4        ;RESTORE R4
(2)  021232  000205                            RTS     R5              ;EXIT
(2)
(2)  021234                            5$:                             ;COME HERE ON LOAD ERROR
(2)  021234  005745                            TST     -(5)
(2)  021236  105204                            INCB    R4              ;UPDATE ERROR COUNTER.
(2)  021240  100324                            BPL     1$              ;IF NOT TOO MANY, TRY AGAIN.
(2)  021242  000000                            HALT                    ;MICRO CODE LOAD ERROR.
(2)                                                                    ;KMC-11 FAULT. YOU COULD TRY
(2)  021244  000722                            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)  021246  010046                    $TLKW:  MOV     R0,-(SP)        ;SAVE R0
(2)  021250  012500                            MOV     (5)+,R0         ;GET DEVICE OFFSET
(2)  021252  052700  000340                    BIS     #340,R0         ;ADD WRITE CODE.
(2)  021256  004737  021530                    JSR     PC,$LPW         ;WAIT FOR FAST PATH READY
(2)  021262  010037  021354                    MOV     R0,W1
(2)  021266  010077  160076                    MOV     R0,@KMAD4
(2)  021272  112777  000005  160064            MOVB    #5,@KMAD2       ;ISSUE FAST PATH WRITE
(2)  021300  004737  021530                    JSR     PC,$LPW         ;WAIT FOR RDY
(2)  021304  011537  021356                    MOV     (5),W2
(2)  021310  112577  160054                    MOVB    (5)+,@KMAD4     ;WRITE LOW BYTE DATA.
(2)
```

```
(2)  021314  112777  000005  160042          MOVB    #5,@KMAD2       ;FP WRITE
(2)  021322  004737  021530                  JSR     PC,$LPW
(2)  021326  111537  021360                  MOVB    (5),W3
(2)  021332  112577  160032                  MOVB    (5)+,@KMAD4     ;WRITE HIGH BYTE
(2)  021336  112777  000005  160020          MOVB    #5,@KMAD2
(2)  021344  004737  021530                  JSR     PC,$LPW
(2)  021350  012600                          MOV     (SP)+,R0
(2)  021352  000205                          RTS     R5              ;EXIT DONE.
(2)  021354  000000                  W1:     0
(2)  021356  000000                  W2:     0
(2)  021360  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)  021362  010046                  $TLKR:  MOV     R0,-(SP)        ;SAVE R0
(2)  021364  012500                          MOV     (5)+,R0         ;GET OFFSET
(2)  021366  052700  000300                  BIS     #300,R0         ;ADD READ CODE
(2)  021372  004737  021530                  JSR     PC,$LPW         ;WAIT TILL READY
(2)  021376  110077  157766                  MOVB    R0,@KMAD4
(2)  021402  112777  000005  157754          MOVB    #5,@KMAD2       ;ISSUE WRITE FP
(2)  021410  004737  021530                  JSR     PC,$LPW
(2)  021414  010037  021524                  MOV     R0,RD1
(2)  021420                          1$:
(3)  021420  004537  022016                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  021424  104000                          ERROR                   ;/TIME-OUT ERROR
(3)                                          ;/WE FAILED TO COMPLETE
(3)                                          ;/CURRENT OPERATION.
(3)                                          ;/CONTINUES IN THIS LOOP
(3)                                          ;/WOULD MAKE US "HANG" HERE
(3)
(3)  021426  000774                          BR              1$
(3)
(3)                                          ;/RETURNS HERE-FROM-TIMED OUT.
(2)  021430  032777  000040  157722          BIT     #BIT5,@KMAD0    ;FAST PATH GOT DATA?
(2)  021436  001370                          BNE     1$
(2)  021440  112777  000004  157716          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
(2)  021446  004737  021530                  JSR     PC,$LPW
(2)  021452  117737  157712  021526          MOVB    @KMAD4,$DATR    ;GET LOW BYTE
(2)  021460                          2$:
(3)  021460  004537  022016                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
(3)
(3)  021464  104000                          ERROR                   ;/TIME-OUT ERROR
(3)                                          ;/WE FAILED TO COMPLETE
(3)                                          ;/CURRENT OPERATION.
(3)                                          ;/CONTINUES IN THIS LOOP
(3)                                          ;/WOULD MAKE US "HANG" HERE
(3)
(3)  021466  000774                          BR              2$
```

```
(3)
(3)                                                             ;/RETURNS HERE-FROM-TIMED OUT.
(2)  021470  032777  000040  157662          BIT     #BIT5,@KMAD0    ;FAST PATH READY?
(2)  021476  001370                          BNE     2$
(2)  021500  112777  000004  157656          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
(2)  021506  004737  021530                  JSR     PC,$LPW
(2)  021512  117737  157652  021527          MOVB    @KMAD4,$DATR+1  ;SAVE HIGH BYTE
(2)  021520  012600                          MOV     (SP)+,R0
(2)  021522  000205                          RTS     R5
(2)  021524  000000                  RD1:    0
(2)  021526  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)  021530  010146                  $LPW:   MOV     R1,-(SP)        ;SAVE R1
(2)  021532  005001                          CLR     R1
(2)  021534  122777  000377  157622  1$:     CMPB    #377,@KMAD2     ;FINISHED INSTRUCTION?
(2)  021542  001403                          BEQ     2$
(2)  021544  005201                          INC     R1              ;TIME OUT?
(2)  021546  001372                          BNE     1$
(2)  021550  000411                          BR      10$
(2)
(2)  021552  032777  000020  157600  2$:     BIT     #BIT4,@KMAD0    ;FAST PATH READ?
(2)  021560  001403                          BEQ     3$
(2)  021562  005201                          INC     R1              ;NO - TIME OUT?
(2)  021564  001372                          BNE     2$
(2)  021566  000402                          BR      10$             ;YES - REPORT AN ERROR
(2)
(2)  021570  012601                  3$:     MOV     (SP)+,R1        ;RESTORE R1
(2)  021572  000207                          RTS     PC              ;EXIT
(2)
(2)  021574                          10$:
(3)  021574  104401  021602                  TYPE    ,65$            ;;TYPE ASCIZ STRING
(3)  021600  000407                          BR      64$             ;;GET OVER THE ASCIZ
(3)                                  ;;65$:  .ASCIZ  <200>#LPA-11 FAULT#
(3)  021620                          64$:
(2)
(2)  021620  000000                  11$:    HALT                    ;LPA-11 FAULT RUN LPA-11
(2)  021622  000776                          BR      11$             ;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)  021624  010046          $OUTLP: MOV     R0,-(SP)        ;SAVE R0
(2)  021626  010146                  MOV     R1,-(SP)        ;SAVE R1
(2)
(2)  021630  012700  001406          MOV     #.DVLS,R0       ;PROGRAM DEFINED LIST.
(2)  021634  005001                  CLR     R1
(2)  021636  005710          1$:     TST     (0)             ;TERMINATOR REACHED?
(2)  021640  001421                  BEQ     10$             ;YES NEXT STEP.
(2)  021642  027520  000000          CMP     @(5),(0)+       ;MATCH WITH ADDR IN LIST?
(2)  021646  001402                  BEQ     2$
(2)  021650  005201                  INC     R1
(2)  021652  000771                  BR      1$
(2)
(2)  021654  010137  021672  2$:     MOV     R1,3$           ;SAVE OFFSET, DEVICE KNOWN.
(2)  021660  005725                  TST     (5)+
(2)  021662  013537  021674          MOV     @(5)+,4$        ;GET DATA TO BE WRITTEN
(2)  021666  004537  021246          JSR     R5,$TLKW        ;DO WRITE
(2)  021672  000000          3$:     .WORD   0               ;DEVICE OFFSET
(2)  021674  000000          4$:     .WORD   0               ;DATA TO BE WRITTEN.
(2)  021676  012601                  MOV     (SP)+,R1
(2)  021700  012600                  MOV     (SP)+,R0
(2)  021702  000205                  RTS     R5
(2)  021704  017520  000000  10$:     MOV     @(5),(0)+       ;SAVE ADDR.
(2)  021710  005010                  CLR     (0)
(2)  021712  004537  020416          JSR     R5,$LPAI
(2)  021716  001406                  .WORD   .DVLS
(2)  021720  000755                  BR      2$
(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)  021722  010046          $INLP:  MOV     R0,-(SP)        ;SAVE R0
(2)  021724  010146                  MOV     R1,-(SP)        ;SAVE R1
(2)
(2)  021726  012700  001406          MOV     #.DVLS,R0       ;PROG DEFINED ADDR. LIST.
(2)  021732  005001                  CLR     R1
(2)  021734  005710          1$:     TST     (0)             ;EOL REACHED?
(2)  021736  001420                  BEQ     10$             ;YES - DEFINE NEW ADDR.
(2)
(2)  021740  027520  000000          CMP     @(5),(0)+       ;ADDR. MATCH?
(2)  021744  001402                  BEQ     2$
```

```
(2)  021746  005201                    INC     R1
(2)  021750  000771                    BR      1$
(2)
(2)  021752  010137  021764    2$:     MOV     R1,3$           ;SAVE LIST OFFSET
(2)  021756  005725                    TST     (5)+
(2)  021760  004537  021362            JSR     R5,$TLKR        ;GO READ DEVICE
(2)          021764            $OFS=.
(2)  021764  000000            3$:     .WORD   0               ;OFFSET OF DEVICE
(2)
(2)  021766  013735  021526            MOV     $DATR,@(5)+     ;STORE DATA.
(2)  021772  012601                    MOV     (SP)+,R1        ;RESTORE R1
(2)  021774  012600                    MOV     (SP)+,R0        ;RESTORE R2
(2)  021776  000205                    RTS     R5              ;EXIT
(2)
(2)  022000  017520  000000    10$:    MOV     @(5),(0)+
(2)  022004  005010                    CLR     (0)
(2)  022006  004537  020416            JSR     R5,$LPAI
(2)  022012  001406                    .WORD   .DVLS
(2)  022014  000756                    BR      2$
(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)  022016  020537  022052    $TOUT:  CMP     R5,$SAD         ;SAME ADDR?
(2)  022022  001405                    BEQ     1$
(2)  022024  010537  022052            MOV     R5,$SAD         ;NO-SAVE THIS ADDR.
(2)  022030  005037  022054            CLR     $CNT            ;CLR CNT AT ADDR.
(2)  022034  000403                    BR      2$
(2)  022036  005237  022054    1$:     INC     $CNT            ;OVERFLOW?
(2)  022042  100402                    BMI     3$              ;YES-ERROR RETURN
(2)  022044  062705  000004    2$:     ADD     #4,R5           ;NO-NON ERROR RETURN
(2)  022050  000205            3$:     RTS     R5              ;RETURN.
(2)
(2)  022052  000000            $SAD:   .WORD   0               ;CONTAINS LOOP ADDR.
(2)  022054  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)  022056  000005            $RESET: RESET                   ;RESET THE WORLD.
(3)
(3)                                    ;*      MOV     @2$,1$  ;/READ DEVICE REG 2$,PUT DATA IN 1$.
(2)  022070  005737  021104            TST     $AERR           ;IF NO ERROR,LOOP
(2)  022074  001004                    BNE     10$             ;THERE WAS AN ERROR.
(2)  022076  062737  000002  022112    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)  022104  000764                     BR      $RESET
(2)  022106                     10$:
(2)  022106  000207                     RTS     PC
(2)  022110  000000             1$:     .WORD   0               ;JUNK LOC.
(2)  022112  160000             2$:     .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)  022114                     SDELAY:
(2)  022114  005737  022176             TST     RTCCSR          ;CLOCK PRESENT?
(2)  022120  100016                     BPL     10$
(2)  022122  012737  000002  022166     MOV     #2,TIME
(2)  022130  052777  000115  000040     BIS     #115,@RTCCSR    ;START CLOCK
(2)  022136  005037  177776             CLR     PS
(2)  022142  005737  022166     1$:     TST     TIME
(2)  022146  001375                     BNE     1$
(2)  022150  005077  000022             CLR     @RTCCSR         ;STOP CLOCK
(2)
(2)  022154  000207                     RTS PC
(2)  022156  105237  022166     10$:    INCB    TIME
(2)  022162  001375                     BNE     10$
(2)  022164  000207                     RTS     PC
(2)
(2)  022166  000000             TIME:   .WORD   0
(2)
(2)  022170  005337  022166     CLKINT: DEC     TIME
(2)  022174  000002                     RTI
(2)  022176  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)  022200                     $UTK:
(2)  022200  005037  001406             CLR     .DVLS
```

```
(2)  022204                                  21$:
(3)  022204  104401  022212                         TYPE    ,65$            ;;TYPE ASCIZ STRING
(3)  022210  000405                                 BR      64$             ;;GET OVER THE ASCIZ
(3)                                          ;;65$: .ASCIZ  <200>#E OR D?#
(3)  022224                                  64$:
(2)  022224  105777  156714                 1$:     TSTB    @$TKS
(2)  022230  100375                                 BPL     1$
(2)  022232  117737  156710  022354                MOVB    @$TKB,20$       ;GET INPUT
(2)  022240  104401  022354                         TYPE,   20$             ;ECHO, NEXT MESSAGE.
(2)  022244  142737  000240  022354                BICB    #240,20$        ;STRIP PARITY, LC
(2)  022252  104412                                 RDOCT                   ;GET ADDR.
(2)  022254  012637  022352                         MOV     (SP)+,14$
(2)  022260  123727  022354  000104                CMPB    20$,#'D         ;DEPOSIT?
(2)  022266  001411                                 BEQ     10$
(2)
(2)  022270  004537  021722                         JSR     R5,$INLP        ;GET DATA
(2)  022274  022352                          2$:    .WORD   14$
(2)  022276  022310                                 .WORD   5$
(2)
(3)  022300  013746  022310                         MOV     5$,-(SP)        ;;SAVE 5$ FOR TYPEOUT
(3)  022304  104402                                 TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(2)  022306  000736                                 BR      21$             ;LOOP.
(2)  022310  000000                          5$:    .WORD   0
(2)
(2)  022312                                  10$:
(3)  022312  104401  022320                         TYPE    ,67$            ;;TYPE ASCIZ STRING
(3)  022316  000404                                 BR      66$             ;;GET OVER THE ASCIZ
(3)                                          ;;67$: .ASCIZ  <200>#DATA= #
(3)  022330                                  66$:
(2)  022330  104412                                 RDOCT
(2)  022332  012637  022350                         MOV     (SP)+,13$
(2)
(2)  022336  004537  021624                 11$:    JSR     R5,$OUTLP       ;OUTPUT ROUTINE.
(2)  022342  022352                          12$:   .WORD   14$             ;DEVICE ADDR.
(2)  022344  022350                                 .WORD   13$             ;DATA
(2)  022346  000716                                 BR      21$
(2)
(2)  022350  000000                          13$:   .WORD   0
(2)  022352  000000                          14$:   .WORD   0
(2)  022354  100001  042504  044526  20$:    .ASCIZ  <1><200>#DEVICE ADDR= #
(2)  022362  042503  040440  042104
(2)  022370  036522  000040
(2)                                                 .EVEN
(2)
(1)
(1)
(2)                                                 ;
(2)                                                 ;THIS ROUTINE LOOKS THROUGH CURENT .DVLS FOR A/D ADDR.
(2)                                                 ;IF UNFOUND,GENERATES IT. THIS ROUTINE'S WHOLE PURPOSE IS
(2)                                                 ;TO SET UP THE USER PROGRAM TO LINK TO FILE ''DRLPX2'' FOR
(2)                                                 ;SAMPLE TAKEING PURPOSES.
(2)                                                 ;       TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
(2)                                                 ;       A/D CSR IN BSEL 4,AND 5.
(2)                                                 ;       (2) HE MUST CALL THIS ROUTINE:
(2)                                                 ;               JSR     R5,$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)                                             ;
(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)  022374  012537  022404          $PUTS:  MOV     (5)+,1$                 ;GET ADDR OF ADDR. OF A/D
(2)  022400  004537  021722                  JSR     R5,$INLP
(2)  022404  000000                  1$:     .WORD   0
(2)  022406  022502                          .WORD   10$
(2)  022410  113777  021764  156756          MOVB    $OFS,@KMAD6
(2)  022416  113777  021764  156752          MOVB    $OFS,@KMAD7
(2)  022424  013737  022404  022444          MOV     1$,2$
(2)  022432  062737  000002  022444          ADD     #2,2$
(2)  022440  004537  021722                  JSR     R5,$INLP
(2)  022444  000000                  2$:     .WORD   0
(2)  022446  022502                          .WORD   10$
(2)  022450  113777  021764  156710          MOVB    $OFS,@KMAD3
(2)  022456  152777  000340  156710          BISB    #340,@KMAD6
(2)  022464  152777  000300  156704          BISB    #300,@KMAD7
(2)  022472  152777  000300  156666          BISB    #300,@KMAD3
(2)  022500  000205                          RTS     R5
(2)  022502  000000                  10$:    .WORD   0
(2)
2082            042000                                .=42000
2083                                  ;HERE RESIDES THE MICRO CODE TO BE LOADED INTO THE KMC-11
2084            042300                                .=42300         ;FREE LOCATION AFTER MICRO-CODE
2085
2086
2087            000001                .END
```

```
ABASE = 167770           1183#   1196
ACDW1 = 000000           1196
ACDW2 = 000000           1196
ACPUOP= 000000           1196
ADDW0 = 000000           1196
ADDW1 = 000000           1196
ADDW10= 000000           1196
ADDW11= 000000           1196
ADDW12= 000000           1196
ADDW13= 000000           1196
ADDW14= 000000           1196
ADDW15= 000000           1196
ADDW2 = 000000           1196
ADDW3 = 000000           1196
ADDW4 = 000000           1196
ADDW5 = 000000           1196
ADDW6 = 000000           1196
ADDW7 = 000000           1196
ADDW8 = 000000           1196
ADDW9 = 000000           1196
ADEVCT= 000000           1196
ADEVM = 000000           1196
AENV  = 000000           1196
AENVM = 000000           1196
AFATAL= 000000           1196
AMADR1= 000000           1196
AMADR2= 000000           1196
AMADR3= 000000           1196
AMADR4= 000000           1196
AMAMS1= 000000           1196
AMAMS2= 000000           1196
AMAMS3= 000000           1196
AMAMS4= 000000           1196
AMSGAD= 000000           1196
AMSGLG= 000000           1196
AMSGTY= 000000           1196
AMTYP1= 000000           1196
AMTYP2= 000000           1196
AMTYP3= 000000           1196
AMTYP4= 000000           1196
APASS = 000000           1196
APRIOR= 000000           1196
APTCSU= 000040           2075    2078#
APTENV= 000001           2066    2075    2078#
APTSIZ= 000200           1303    2078#
APTSPO= 000100           2075    2078#
ASWREG= 000000           1196
ATESTN= 000000           1196
AUNIT = 000000           1196
AUSWR = 000000           1196
AVECT1= 100300           1184#   1196    1252
AVECT2= 000000           1196
BASEBA  001450           1254#   1305*   1311    1444    2005
BASEBR  001454           1256#   1308*   1309*   1474
BASEIV  001452           1255#   1306*   1307*   1445    2006
BEGIN   001544           1188    1298#   1384
```

```
BEGIN1  001554          1190    1301#   2072
BIT0  = 000001          1185#   1535    1550    1728    1763    1790    1914    1964
BIT00 = 000001          1185#
BIT01 = 000002          1185#
BIT02 = 000004          1185#
BIT03 = 000010          1185#
BIT04 = 000020          1185#
BIT05 = 000040          1185#
BIT06 = 000100          1185#
BIT07 = 000200          1185#
BIT08 = 000400          1185#   2065
BIT09 = 001000          1185#   2065    2066
BIT1  = 000002          1185#   2081
BIT10 = 002000          1185#   1641    1910    1960
BIT11 = 004000          1185#   2081
BIT12 = 010000          1185#   1377    1693
BIT13 = 020000          1185#   2066
BIT14 = 040000          1185#   1605    2018    2065    2081
BIT15 = 100000          1185#   1595    1896    2081
BIT2  = 000004          1185#
BIT3  = 000010          1185#
BIT4  = 000020          1185#   2081
BIT5  = 000040          1185#   2081
BIT6  = 000100          1185#   1623
BIT7  = 000200          1185#   1613    1883    1916
BIT8  = 000400          1185#
BIT9  = 001000          1185#
BPTVEC= 000014          1185#
BRLEV1  001524          1280#
BRLEV2  001526          1281#   1728*   1731    1753    1758*
BRLEV3  001530          1282#   1391    1729*   1730*   1739    1751    1790*   1794    1801    1811    1813*   1914*   1918
                        1925    1928    1931    1946    1949    1957*   1964*   1968    1975    1977    1980    1994*   2061
BUFFER  020414          2080#
BUSTRP  013660          1324    2037#
BYPASS  013036          1997#
BYPAS1  013124          2000    2004    2009#   2012
CKSWR = 104407          2065    2066    2079#
CLKINT  022170          2081#
CPU     001456          1257#
CR    = 000015          1185#   2075
CRLF  = 000200          1185#   1341    2075
DDISP = 177570          1185#   1196    1303
DEPCNT  014374          1433    2043#
DF0     015370          1202    1208    1214    1220    1226    1232    1239    1245    1251    2062#
DH1     015036          1200    2053#
DH10    015247          1243    1249    2057#
DH2     015106          1206    2054#
DH3     015154          1212    2055#
DH4     015222          1218    1224    1230    1237    2056#
DIOBRL  001522          1279#   1474*
DISPLA  001142          1196#   1303*   2065*   2066*
DISPRE  000174          1188#   1303
DRADD   001464          1262#   1444*   1459    1460    1462    1464    2001*   2005*   2059    2060    2061
DRIV    001466          1263#   1416*   1445*   1466    1467    1469    1471    2002*   2006*
DRLPX2= ****** G          52#   2081
DRT21   012146          1643    1909#
```

```
DSWR   = 177570          1185#  1196   1303
DT1      015326          1201   1207   1213   2059#
DT10     015352          1244   1250   2061#
DT4      015342          1219   1225   1231   1238   2060#
EMTVEC=  000030          1185#  1303*
EM1      014416          1199   2044#
EM10     014676          1242   2051#
EM11     014761          1248   2052#
EM2      014447          1205   2045#
EM3      014477          1211   2046#
EM4      014530          1217   2047#
EM5      014562          1223   2048#
EM6      014615          1229   2049#
EM7      014642          1236   2050#
ERRVEC=  000004          1185#  1303*  1310*  1333*  1334*  2065*
EXTCNT   013656          2018*  2029*  2034#
EXTTMP   013654          2022   2023   2027   2028   2033#
EXTTST   013476          1191   2016#
GNS    = ****** U        1188   1341   1345   1407   1410   2079   2081
GRBHIO   001504          1272#  1464*  1465*
GRDAI    001500          1270#  1460*  1461*  1484   1594   1645   1649   1657   1661   1669   1673   1681   1685
                         1700   1704   1713   1717   1738   1750   1767   1777   1782   1784   1792   1805   1810
                         1812   1822   1833   1850   1860   1871   1876   1901   1921   1948   1950   1971   2022
                         2023   2027   2028
GRDIO    001502          1271#  1462*  1463*  1485   1490   1491   1498   1500   1508   1509   1517   1518   1526
                         1527   1538   1540   1541   1553   1555   1556   1567   1572   1573   1579   1584   1585
                         1592   1644   1648   1656   1660   1668   1672   1680   1684   1698   1701   1703   1714
                         1716   1737   1747   1749   1765   1775   1776   1791   1802   1803   1821   1833   1849
                         1860   1870   1873   1886   1899   1920   1924   1926   1927   1929   1930   1932   1945
                         1947   1970   1974   1976   1978   1979   1981   1982   2019   2021   2024   2026
GRIVA    001512          1275#  1466*  1478*
GRIVB    001516          1277#  1469*  1470*  1478*
GRIVSA   001514          1276#  1467*  1468*  1478*
GRIVSB   001520          1278#  1471*  1472*  1478*
GRSTAT   001476          1269#  1459*  1483   1596   1597   1604   1606   1607   1614   1615   1622   1624   1625
                         1885   1888   1898   1903   1923   1935   1936   1938   1952*  1953   1973   1986
GTSWR  = 104406          1341   2079#
HT     = 000011          1185#  2075
INTBIT   001510          1274#  1349*  1356*  1368   1918   1968
IOTEST   003266          1340   1352   1359   1362   1392   1443#
IOTST1   003346          1456#
IOTTS1   003506          1458   1474#
IOTVEC=  000020          1185#  1303*
JUMPER   001540          1286#  1299*  1346,  1353
KMAD0    001360          1252#  1304   2081*
KMAD1    001362          1252#  1304
KMAD2    001364          1252#  2081*
KMAD3    001366          1252#  2081*
KMAD4    001370          1252#  2081*
KMAD5    001372          1252#
KMAD6    001374          1252#  2081*
KMAD7    001376          1252#  1304   2081*
LF     = 000012          1185#  2075
LOC1     001470          1266#
LOC2     001472          1267#
LOC3     001474          1268#
```

```
LPADH   001372          1252#
LPADL   001370          1252#
LPCI    001360          1252#
LPCO    001364          1252#
LPMR    001362          1252#
LPMS1   001374          1252#
LPMS2   001376          1252#
LPSO    001366          1252#
MINSIN  001534          1284#   1350*   1357*   1371    1378    1476    1710
NBEXT   001462          1260#   1446*   1997    2003*   2007*
NMBEXT  001460          1259#   1332*   1342    1446    2007
NOTLCH  001506          1273#   1348*   1355*   1365    1437    1696    1729    1734    1769    1794    1800    1818    1847
ODDJMP  001532          1283#   1475*   1476*   1477*   1650    1659    1662    1671    1674    1683    1686    1691    1694
                        1732    1771    1796    1819    1846    1877
PIRQ  = 177772          1185#
PIRQVE= 000240          1185#
PR0   = 000000          1185#
PR1   = 000040          1185#
PR2   = 000100          1185#
PR3   = 000140          1185#
PR4   = 000200          1185#
PR5   = 000240          1185#
PR6   = 000300          1185#
PR7   = 000340          1185#
PS    = 177776          1185#   2081*
PSW   = 177776          1185#
PWRMSG  016530          2072    2073#
PWRVEC= 000024          1185#   1303*   2072*
RBEG    001560          1300    1302#
RBEG1   002246          1335#
RBEG2   003314          1417    1447#   2010
RDCHR = 104410          2077    2079#
RDLIN = 104411          2076    2079#
RDOCT = 104412          1429    2079#   2081
RD1     021524          2081#*
RESVEC= 000010          1185#
RTCCSR  022176          2081#*
SDELAY  022114          2081#
SETADD  003360          1457    1459#   2017
STACK = 001100          1185#   1303    2016
STKLMT= 177774          1185#
SWDPOB  014307          1390    2042#
SWINTB  014017          1367    2039#
SWNLB   013721          1364    1425    2038#
SWPOSB  014115          1370    2040#
SWR     001140          1196#   1303*   1322    1325    1341    1405    1412    1426    1430*   1435    1641    1910    1960
                        1999    2065    2066    2072*   2077*
SWREG   000176          1188#   1303    1341    1426    2077
SWTRAB  014211          1373    2041#
SW0   = 000001          1185#
SW00  = 000001          1185#
SW01  = 000002          1185#
SW02  = 000004          1185#
SW03  = 000010          1185#
SW04  = 000020          1185#
SW05  = 000040          1185#
```

```
SW06  = 000100           1185#
SW07  = 000200           1185#
SW08  = 000400           1185#
SW09  = 001000           1185#
SW1   = 000002           1185#
SW10  = 002000           1185#
SW11  = 004000           1185#
SW12  = 010000           1185#   1999
SW13  = 020000           1185#   1322    1405
SW14  = 040000           1185#
SW15  = 100000           1185#   1325    1412
SW2   = 000004           1185#
SW3   = 000010           1185#
SW4   = 000020           1185#
SW5   = 000040           1185#
SW6   = 000100           1185#
SW7   = 000200           1185#
SW8   = 000400           1185#
SW9   = 001000           1185#
TALK    003202           1363    1366    1369    1372    1389    1422#
TBITVE= 000014           1185#
TIME    022166           2081#*
TKVEC = 000060           1185#
TPVEC = 000064           1185#
TRANST  001536           1285#   1351*   1358*   1374    1380    1477
TRAPVE= 000034           1185#   1303*
TRTVEC= 000014           1185#
TSTNUM  015406           2059    2060    2061    2063#   2066*
TST1    003564           1479#
TST10   004256           1548#
TST11   004364           1565#
TST12   004666           1575    1577#
TST13   005170           1587    1590#
TST14   005266           1599    1602#
TST15   005346           1609    1612#
TST16   005410           1617    1620#
TST17   005470           1627    1640#
TST2    003640           1487#
TST20   005614           1652    1655#
TST21   005726           1664    1667#
TST22   006040           1676    1679#
TST23   006152           1688    1690#
TST24   006412           1692    1725#
TST25   006620           1761#
TST26   007016           1788#
TST27   007236           1815#
TST3    003710           1493    1495#
TST30   010516           1840    1843#
TST31   011646           1867    1869#
TST32   011760           1879    1882#
TST33   012046           1890    1895#
TST34   012146           1905    1909#
TST35   012544           1911    1959#
TST36   013034           1961    1996#
TST4    003760           1502    1505#
TST5    004030           1511    1514#
```

```
TST6    004100          1520    1523#
TST7    004154          1533#
TYPDS = 104405          2012    2079#
TYPE  = 104401          1324    1341    1345    1407    1410    1424    1428    1432    2012    2013    2066    2068    2070
                        2072    2075    2077    2079#   2081
TYPOC = 104402          1409    2068    2077    2079#   2081
TYPON = 104404          2079#
TYPOS = 104403          1343    2079#   2081
UNEXPT  003262          1440#   1478
VECTOR  001400          1252#
VECTPS  001402          1252#
VECTRP  002756          1333    1395#
VERSN   001404          1252#
W1      021354          2081#*
W2      021356          2081#*
W3      021360          2081#*
$AERR   021104          1315    2081#*
$APTHD  001000          1195#
$ASTAT= ****** U        2078
$ATYC   020110          2078#
$ATY1   020064          2078#
$ATY3   020072          2075    2078#
$ATY4   020102          2066    2078#
$AUTOB  001134          1196#   1336    1341*   1360    2077
$BASE   001242          1196#   1305
$BDADR  001122          1196#
$BDDAT  001126          1196#   1491    1492    1500    1501    1509    1510    1518    1519    1527    1528    1541    1542
                        1556    1557*   1558    1573    1574    1585    1586    1597    1598    1607    1608    1615    1616
                        1625    1626    1649    1650*   1651    1661    1662*   1663    1673    1674*   1675    1685    1686*
                        1687    1704    1705    1717    1718    1738    1739*   1740    1750    1751*   1753    1777    1778
                        1805    1806    1835*   1839    1862*   1866    1876    1877*   1878    1888    1889    1903    1904
                        1915*   1953*   1965*   2059    2061
$CDW1   001246          1196#
$CHARC  017054          2075#*
$CKSWR  017162          2077#   2079
$CMTAG  001100          1196#   1303    1453
$CM3  = 000000          1196#
$CNT    022054          2081#*
$CNTLG  020034          2077#
$CNTLU  020027          2077#
$CPUOP  001214          1196#
$CRLF   001163          1196#   2066    2068    2075    2077
$DATR   021526          2081#*
$DBLK   013466          2013#
$DEVCT  001176          1196#
$DEVM   001244          1196#
$DOAGN  013226          2012#
$DTBL   013456          2013#
$ENDAD  013216          1193    1341    2012#   2066
$ENDCT  013164          1303    2012#
$ENDMG  013235          2012#
$ENULL  013232          2012#
$ENV    001206          1196#   1341    2066    2075    2078
$ENVM   001207          1196#   1303    2075    2078
$EOP    013134          2008    2012#
$EOPCT  013156          1303*   2012#
```

```
$ERFLG  001103          1196#   2065*   2066*
$ERMAX  001115          1196#   1303*   2065*
$ERROR  015602          1303    2066#
$ERRPC  001116          1196#   2059    2060    2061    2066*   2068
$ERRTB  001250          1196#   2068
$ERRTY  015774          2066    2068#
$ERTTL  001112          1196#   2066*
$ESCAP  001160          1196#   1303*   2065*   2066
$ETABL  001206          1196#
$ETEND  001250          1195    1196#
$FATAL  001170          1196#   2078*
$FFLG   020330          2078#*
$FILLC  001156          1196#   2075
$FILLS  001155          1196#   2075
$GDADR  001120          1196#
$GDDAT  001124          1196#   1314    1377*   1378    1380    1386*   1488*   1492    1496*   1506*   1510    1515*   1519
                        1525*   1526    1528    1531*   1535*   1539    1542    1546*   1550*   1554    1558    1562*   1566*
                        1567    1574    1578*   1579    1586    1595*   1596    1598    1605*   1606    1608    1513*   1614
                        1616    1623*   1624    1626    1646*   1651    1658*   1659*   1660    1663    1670*   1671*   1672
                        1675    1682*   1683*   1684    1687    1693*   1694    1696    1698    1702    1705    1710    1715
                        1718    1723*   1731*   1732    1734    1737    1740    1748    1752*   1763*   1769    1771    1775
                        1778    1783    1785*   1796    1799*   1800*   1801*   1802    1806    1817*   1818*   1819*   1820
                        1839    1845*   1846*   1847*   1848    1866    1874*   1883*   1889    1896*   1916*   1951*   1966*
                        2059    2061
$GET42  013206          2012#
$GTSWR  017232          2077#   2079
$HD   = 000000          1181
$HIBTS  001000          1195#
$HIOCT  017160          2076#*
$ICNT   001104          1196#
$ILLUP  016522          2072#
$INLF   021722          1314    1491    1500    1509    1518    1527    1541    1556    1572    1573    1584    1585    1597
                        1607    1615    1625    1649    1661    1673    1685    1701    1704    1714    1717    1738    1747
                        1750    1777    1782    1805    1810    1833    1860    1876    1888    1901    1903    1924    1927
                        1930    1938    1945    1948    1974    1979    1986    2022    2027    2081#
$INTAG  001135          1196#   2077
$ITEMB  001114          1196#   2066*   2068
$LF     001164          1196#   2066    2075    2077
$LFLG   020327          2078#*
$LOAD   021106          2081#
$LPADR  001106          1196#   1303*   2065*
$LPAI   000416          2081#
$LPERR  001110          1196#   1303*   1524*   1534*   1549*   1726*   1762*   1789*   1913*   1963*   2065*   2066
$LPW    021530          2081#
$MADR1  001220          1196#
$MADR2  001224          1196#
$MADR3  001230          1196#
$MADR4  001234          1196#
$MAIL   001166          1195    1196#   1303    1341    2065    2066    2075
$MAMS1  001216          1196#
$MAMS2  001222          1196#
$MAMS3  001226          1196#
$MAMS4  001232          1196#
$MBADR  001002          1195#
$MFLG   020326          2078#*
$MNEW   020052          2077#
```

```
$MSGAD  001202          1196#   2078*
$MSGLG  001204          1196#   2078*
$MSGTY  001166          1196#   2078*
$MSWR   020041          1428    2077#
$MTYP1  001217          1196#
$MTYP2  001223          1196#
$MTYP3  001227          1196#
$MTYP4  001233          1196#
$NULL   001154          1196#   2075
$NWTST= 000001          1479#   1487#   1495#   1505#   1514#   1523#   1533#   1548#   1565#   1577#   1590#   1602#   1612#
                        1620#   1640#   1655#   1667#   1679#   1690#   1725#   1761#   1788#   1815#   1843#   1869#   1882#
                        1895#   1909#   1959#   1996#
$OCNT   016352          2070#*
$OFS  - 021764          2081#
$OMODE  016354          2070#*
$OUTLP  021624          1483    1484    1485    1490    1498    1508    1517    1526    1538    1540    1553    1555    1567
                        1572    1579    1584    1592    1594    1596    1604    1606    1614    1622    1624    1644    1645
                        1648    1656    1657    1660    1668    1669    1672    1680    1681    1684    1698    1700    1703
                        1713    1716    1737    1749    1765    1767    1775    1776    1784    1791    1792    1802    1803
                        1812    1821    1822    1833    1849    1850    1860    1870    1871    1873    1885    1886    1898
                        1899    1920    1921    1923    1926    1929    1932    1935    1936    1947    1950    1970    1971
                        1973    1976    1978    1981    1982    1984    2019    2021    2023    2024    2026    2028    2081#
$OVER   015566          2065#
$PASS   001174          1196#   1303*   2012*
$PASTM  001006          1195#
$PUTS   022374          2081#
$PWRAD  016516          2072#
$PWRDN  016356          1303    2072#
$PWRMG  016512          2072#
$PWRUP  016430          2072#
$QUES   001162          1196#   2066    2075    2077
$RDCHR  017444          2077#   2079
$RDDEC= ****** U        2079
$RDLIN  017564          2077#   2079
$RDOCT  017060          2076#   2079
$RDSZ = 000007          2077#
$RESET  022056          1335    1443    1447    1456    1499    1875    2081#
$RTNAD  013230          2012#
$R2A  = ****** U        2079
$SAD    022052          2081#*
$SAVRE= ****** U        2079
$SAVR6  016526          2072#*
$SCOPE  015410          1303    2065#
$SETUP= 000137          1234#   1303    1341    2012    2065    2066    2077
$STUP = 177777          1234#
$SVLAD  015532          2065#
$SVPC = 000220          1193#
$SWR  = 161400          1154#   1181    1186    1196    1303    1479    1487    1495    1505    1514    1523    1533    1548
                        1565    1577    1590    1602    1612    1620    1640    1655    1667    1679    1690    1725    1761
                        1788    1815    1843    1869    1882    1895    1909    1959    1996    2012    2065    2066    2072
$SWREG  001210          1196#   1303
$SWRMK= 000000          1186    2065
$TESTN  001172          1196#   1480*   2065*
$TKB    001146          1196#   2077    2081
$TKS    001144          1196#   2077*   2081
$TLKR   021362          2081#
```

```
$TLKW   021246          2081#
$TMDAT  001542          1287#   1313*   1314    1482*   1483    1484    1485    1489*   1490    1497*   1498    1507*   1508
                        1516*   1517    1537*   1538    1539*   1540    1552*   1553    1554*   1555    1572*   1584*   1591*
                        1592    1593*   1594    1603*   1604    1621*   1622    1644*   1645*   1647*   1648    1656*   1657*
                        1668*   1669*   1680*   1681*   1699*   1700    1701    1702*   1703    1712*   1713    1714    1715*
                        1716    1747    1748*   1749    1764*   1765    1766*   1767    1776*   1782    1783*   1784    1791*
                        1792*   1803*   1810    1811*   1812    1821*   1822*   1833*   1849*   1850*   1860*   1870*   1871*
                        1872*   1873    1884*   1885    1886    1897*   1898    1899    1901    1902    1920*   1921*   1922*
                        1923    1924    1925*   1926    1927    1928*   1929    1930    1931*   1932    1934*   1935    1936
                        1938    1939    1945    1946*   1947    1948    1949*   1950    1970*   1971*   1972*   1973    1974
                        1975*   1976    1977*   1978    1979    1980*   1981    1982    1983*   1984    1986    1987    2019*
                        2020*   2021    2024*   2025*   2026
$TN     = 000037        1155#   1181    1479#   1487#   1493    1495#   1502    1505#   1511    1514#   1520    1523#   1533#
                        1548#   1565#   1575    1577#   1587    1590#   1599    1602#   1609    1612#   1617    1620#   1627
                        1640#   1652    1655#   1664    1667#   1676    1679#   1688    1690#   1692    1725#   1761#   1788#
                        1815#   1840    1843#   1867    1869#   1879    1882#   1890    1895#   1905    1909#   1911    1959#
                        1961    1996#
$TOUT   022016          2081#
$TPB    001152          1196#   2075*
$TPFLG  001157          1196#   2075
$TPS    001150          1196#   2075
$TRAP   020332          1303    2079#
$TRAP2  020354          2079#
$TRP    = 000013        2079#
$TRPAD  020366          2079#
$TSTM   001004          1195#
$TSTNM  001102          1196#   1481*   2012*   2065*   2066
$TTYIN  020020          2077#
$TYPBN= ****** U        2079
$TYPDS  013252          2013#   2079
$TYPE   016576          2075#   2078    2079
$TYPEC  017010          2075#   2077
$TYPEX  017056          2075#
$TYPOC  016154          2070#   2079
$TYPON  016170          2070#   2079
$TYPOS  016130          2070#   2079
$UNIT   001200          1196#
$UNITM  001010          1195#
$USWR   001212          1196#
$UTK    022200          1192    2081#
$VECT1  001236          1196#   1306    1308
$VECT2  001240          1196#
$XTSTR  015422          2065#
$$GET4= 000000          2012#
$OFILL  016353          2070#*
$40CAT= ****** U        2065    2066
.       = 042300        1188#   1193#   1194#   1195#   1196#   1252#   1303    1341#   1407#   2012    2013#   2065    2066
                        2068#   2072    2074#   2075    2077#   2078#   2081#   2082#   2084#
.DVLS   001406          1252#   1304*   2081*
.$ASTA= ****** U        2078
.$X     = 001000        1195#
```

```
CLERIN  1635#  1645  1657  1669  1681  1792  1822  1850  1871  1921  1971
CLEROT  1630#  1644  1656  1668  1680  1776  1791  1803  1821  1833  1849  1860  1870  1920  1970
        2019   2024
CLRVCT  1289#  1478
COMMEN  1185#
ENDCOM  1185#
ERROR   1185#  1382  1440  1494  1503  1512  1521  1530  1544  1560  1576  1588  1600  1610  1618
        1628   1653  1665  1677  1689  1707  1720  1742  1755  1780  1808  1841  1868  1880  1891
        1906   1941  1955  1989  2081
ESCAPE  1185#
GETPRI  1185#
GETSWR  1158#  1185#  1341#
MOVEI    170#  1314  1491  1500  1509  1518  1527  1541  1556  1572  1573  1584  1585  1597  1607
        1615   1625  1649  1661  1673  1685  1701  1704  1714  1717  1738  1747  1750  1777  1782
        1805   1810  1833  1860  1876  1888  1901  1903  1924  1927  1930  1938  1945  1948  1974
        1979   1986  2022  2027  2081
MOVEM    157#  1483  1484  1485  1490  1498  1508  1517  1526  1538  1540  1553  1555  1567  1572
        1579   1584  1592  1594  1596  1604  1606  1614  1622  1624  1644  1645  1648  1656  1657
        1660   1668  1669  1672  1680  1681  1684  1698  1700  1703  1713  1716  1737  1749  1765
        1767   1775  1776  1784  1791  1792  1802  1803  1812  1821  1822  1833  1849  1850  1860
        1870   1871  1873  1885  1886  1898  1899  1920  1921  1923  1926  1929  1932  1935  1936
        1947   1950  1970  1971  1973  1976  1978  1981  1982  1984  2019  2021  2023  2024  2026
        2028
MULT    1185#
NEWTST  1185#  1479  1487  1495  1505  1514  1523  1533  1548  1565  1577  1590  1602  1612  1620
        1640   1655  1667  1679  1690  1725  1761  1788  1815  1843  1869  1882  1895  1909  1959
        1996
POP     1185#  2013  2072  2076  2078
PUSH    1185#  2013  2072  2076  2078
REPORT  1185#
SCOPE   1185#  1479  1487  1495  1505  1514  1523  1533  1548  1565  1577  1590  1602  1612  1620
        1640   1655  1667  1679  1690  1725  1761  1788  1815  1843  1869  1882  1895  1909  1959
        1996   2012
SETPRI  1185#
SETTRA  2079#
SETUP   1185#  1303
SKIP    1185#  1493  1502  1511  1520  1529  1575  1587  1599  1609  1617  1627  1652  1664  1676
        1688   1692  1741  1754  1779  1807  1840  1867  1879  1890  1905  1911  1919  1940  1954
        1961   1969  1988
SLASH   1185#
SPACE   1185#
STARS   1185#  1193  1195  1196  1479  1487  1495  1505  1514  1523  1533  1548  1565  1577  1590
        1602   1612  1620  1640  1655  1667  1679  1690  1725  1761  1788  1815  1843  1869  1882
        1895   1909  1959  1996  2012  2013  2065  2066  2068  2070  2072  2075  2076  2077  2078
        2079
SWRSU   1185#  1303#
TOUT    1252#  2081
TRMTRP  2079#
TYPBIN  1185#
TYPDEC  1185#  2012
TYPNAM  1185#  1341
TYPNUM  1185#
TYPOCS  1185#
TYPOCT  1185#  2068  2077  2081
TYPTXT  1185#  1345  1407  1410  2081
$CAL.    748#  2081
```

```
$DMAST    914#
$DMDT    1048#
$MMAST    772#
$$CMRE   1196#
$$CMTM   1196#
$$ESCA   1185#
$$NEWT   1185#  1479    1487    1495    1505    1514    1523    1533    1548    1565    1577    1590    1602    1612    1620
         1640   1655    1667    1679    1690    1725    1761    1788    1815    1843    1869    1882    1895    1909    1959
         1996
$$SET    2079#
$$SETM   1303#
$$SKIP   1185#  1493    1502    1511    1520    1575    1587    1599    1609    1617    1627    1652    1664    1676    1688
         1692   1840    1867    1879    1890    1905    1911    1961
.EQUAT   1156#  1185
.HEADE   1156#  1181
.KMADR     55#  1252
.KSIS     184#  1304
.LOADL    459#  2081
.LPAIN    209#  2081
.PUTCS    418#  2081
.RESET    329#  2081
.SETUP   1157#  1234
.SWRHI   1158#  1186
.SWRLO   1186#
.UTK      699#  2081
.$ACT1   1159#  1193
.$APTB   1159#  1196#
.$APTH   1159#  1195
.$APTY   1159#  2078
.$CATC   1156#  1188
.$CMTA   1156#  1196
.$EOP    1156#  2012
.$ERKO   1156#  2066
.$ERRT   1158#  2068
.$INLP    652#  2081
.$MMAC    141#
.$OUTL    610#  2081
.$PARM   1157#
.$POWE   1157#  2072
.$RDOC   1158#  2076
.$READ   1157#  2077
.$SAVE   1157#
.$SCOP   1157#  2065
.$SPAC   1157#
.$SWDO   1157#
.$TLKW    511#  2081
.$TOUT   1252#  2081
.$TRAP   1157#  2079
.$TYPD   1158#  2013
.$TYPE   1156#  1157#   2075
.$TYPO   1156#  2070

. ABS.  042300     000     CON   RW    ABS   GBL   D
        000000     001     CON   RW    REL   LCL   I
```

```
 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 CRLPFC,CRLPFC/CRF=CRLPAB.MAC,CRLPFC.P11
 RUN-TIME: 24 14 1 SECONDS
 RUN-TIME RATIO: 83/40=2.0
 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   .$$$.
```

```
541                                                     ;WRITE INTO FAST PATH.
542  042076                                 BR      CMNOP           ;RETURN.
(2)  042076  100420                         .WORD   .$$$.
543
544
545                                         ;
546                                         ;THIS ROUTINE TAKES AN A/D SAMPLE.
547                                         ;CALL= CMND 6 IN BSEL2
548                                         ;THESE REGS. MUST BE SET UP IN ADVANCE.
549                                         ;BSEL 3 MUST CONTAIN READ CODE FOR A/D BUFFER.
550                                         ;BSEL 4,5 MUST CONTAIN  A/D CSR SETTING.
551                                         ;BSEL 6 MUST CONTAIN WRITE CODE FOR A/D CSR
552                                         ;BSEL 7 MUST CONTAIN READ CODE FOR A/D CSR
553                                         ; BSEL 3,6,7 WILL REMAIN  UNEFFECTED.
554                                         ; BSEL 4,5 WILL CONTAIN A/D SAMPLE.
555                                         ;BSEL2 WILL CONTAIN CODE 377 WHEN DONE.
556
566
567  042100                                 WTMC    SAMP
(4)  042100  020640                         .WORD   .$$$.
(3)  042102  103040                         .WORD   .$$$.
568  042104                                 MOVE    INP1 <6>,OUTO <11>      ;SEND A/D WRITE CODE.
(3)  042104  122151                         .WORD   .$$$.
569  042106                                 WTMC    SAMP1
(4)  042106  020640                         .WORD   .$$$.
(3)  042110  103043                         .WORD   .$$$.
570  042112                                 MOVE    INP1 <4>,OUTO <11>      ;SEND LOW BYTE CSR INFO.
(3)  042112  122111                         .WORD   .$$$.
571  042114                                 WTMC    SAMP2
(4)  042114  020640                         .WORD   .$$$.
(3)  042116  103046                         .WORD   .$$$.
572  042120                                 MOVE    INP1 <5>,OUTO <11>      ;SEND HIGH BYTE CSR INFO.
(3)  042120  122131                         .WORD   .$$$.
573  042122                                 WTMC    SLOOP
(4)  042122  020640                         .WORD   .$$$.
(3)  042124  103051                         .WORD   .$$$.
574  042126                                 MOVE    INP1 <7>,OUTO <11>      ;SEND READ CODE TO GET A/D CSR.
(3)  042126  122171                         .WORD   .$$$.
575  042130                                 WTMC    SAMP3
(4)  042130  020640                         .WORD   .$$$.
(3)  042132  103054                         .WORD   .$$$.
576  042134                                 WTMM    SLOOP1
(4)  042134  020640                         .WORD   .$$$.
(4)  042136  061620                         .WORD   .$$$.
(3)  042140  103056                         .WORD   .$$$.
577  042142                                 MOVE    INP0 <11>,BREG
(3)  042142  020620                         .WORD   .$$$.
578  042144                                 MOVE    BREG,SPAD <0>
(3)  042144  063220                         .WORD   .$$$.
579  042146                                 WTMM    SLOOP2
(4)  042146  020640                         .WORD   .$$$.
(4)  042150  061620                         .WORD   .$$$.
(3)  042152  103063                         .WORD   .$$$.
580  042154                                 MOVE    INP0 <11>,BREG
(3)  042154  020620                         .WORD   .$$$.
581  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          .D0   = 000400          .SUBWC= 000040
MARINC= 014000          SLOOP3  042170          .BR   = 000400          .F0   = 000020          .SUB2C= 000360
MARLD = 010000          SLOOP4  042200          .BSBRG= 160000          .INC  = 000060          .S0   = 020000
MARLDX= 004000          STARTU  042016          .BSIMM= 100000          .LORN = 000240          .XOR  = 000320
PAGE0 = 000000          WRCMND  042074          .BSMEM= 140000          .MINUS= 000360          .$$$. = 100420
PAGE1 = 001000          WRSILO  042064          .BZ   = 001400          .M0   = 004000          ..LOC = 042040
PAGE2 = 002000          $$$SER= 000001          .C0   = 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 = 000200

. 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: 16/7=2.2
 CORE USED:  35K  (69 PAGES)
```