

# MINC-11

MNCDA DIAGNOSTIC  
CVMNDA0

AH-B095A-MC

COPYRIGHT © 1978

FICHE 1 OF 1

DEC 1978

**digital**

MADE IN USA

## IDENTIFICATION

SEQ 0001

PRODUCT CODE: AC-B094A-MC  
PRODUCT NAME: CVMNDAO MNCDA (D/A) DIAGNOSTIC TEST  
DATE: AUGUST 1978  
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1978  
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
4.1	PROGRAM START
5.0	SOFTWARE SWITCH REGISTER
5.1	OPTIONS
5.2	CONTROL
6.0	ERROR REPORTING
6.1	ERROR COMMENT
6.2	ERROR DATA
7.0	MISCELLANEOUS
7.1	MNCDA BUS ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	MULTIPLE MNCDA INTERFACE TESTING
7.5	RESTRICTIONS
8.0	EXECUTION TIME
9.0	PROGRAM TEST DESCRIPTIONS
9.1	LOGIC TEST
9.2	RAMP LOOP
9.3	STATIC CALIBRATION
9.4	DYNAMIC CALIBRATION
9.5	LOGIC TEST WITH TESTER SUPPORT
10.0	LISTING

1.0 ABSTRACT

THE MNCDA DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE. ADDITIONAL SUB-SECTIONS ARE PROVIDED TO VERIFY THE ANALOG OUTPUT. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5.

2.0 REQUIREMENTS2.1 EQUIPMENT

1. PDP11/03 COMPUTER OR LSI-11 PROCESSOR
2. DLV11 WITH I/O TERMINAL (LA36, VT100, ETC.)
3. MNCDA (D/A) OPTION

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE <XXDP>

1. ENSURE THAT THE DIAGNOSTIC LOAD MEDIA IS INSTALLED IN DRIVE 0.
2. BOOT THE MEDIA BY TYPING '173000G' IF IN THE ODT MICRO-CODE STATE OR CYCLING THE POWER 'ON-OFF' SWITCH.
3. UPON SUCCESSFUL BOOTING OF THE LOAD MEDIA, THE XXDP MONITOR WILL IDENTIFY ITSELF AND INFORM THE OPERATOR OF THE OPERATING OPTIONS THAT MAYBE SELECTED.
4. THE OPERATOR SHOULD TYPE 'R VMND??' FOLLOWED BY A 'RETURN' THE XXDP MONITOR WILL LOAD THE PROGRAM INTO MEMORY AND START THE PROGRAM AT LOCATION 200.

4.0 STARTING PROCEDURE

1. THE PROGRAM WILL RESPOND BY TYPING THE PROGRAM TITLE.
2. THE PROGRAM WILL NOW ASK FOR AN INITIAL SWITCH REGISTER VALUE TO BE STORED IN THE SOFTWARE SWITCH REGISTER.
3. THE PROGRAM WILL NOW DISPLAY THE MENU OF TEST OR LOOP SELECTION. THE OPERATOR SELECTS THE TESTS BY TYPING SELECTED CHARACTER FOLLOWED BY DEPRESSING THE 'RETURN' KEY.

4.1 PROGRAM START

200	STARTING ADDRESS OF THE PROGRAM
204	RESTART ADDRESS OF THE PROGRAM
210	STARTING ADDRESS FOR THE OPTION TESTER.

## 5.0 SOFTWARE SWITCH REGISTER

## 5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW12=1	010000	INHIBIT SIZING THE NUMBER OF MNCDA'S
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <7-0>

## 5.2 CONTROL

1. THE TEST OR LOOP MAYBE STOPPED BY TYPING THE 'CONTROL & C' KEYS. THIS OPERATION WILL STOP THE PROGRAM AND ENABLE THE OPERATOR TO SELECT DIFFERENT PROGRAM COMMANDS.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CONTROL & G' KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.
3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING 'P' (CONTINUE).
4. IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL 'CONTROL & G OR C' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE BY THE PROGRAM.

## 6.0 ERROR REPORTING

## 6.1 ERROR COMMENT

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

## 6.2 ERROR DATA

*UNIT	UNIT NUMBER
*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*BUSADR	MNCDA BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

\*ALWAYS REPORTED

7.0 MISCELLANEOUS  
-----

## 7.1 MNCDA BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' (LOC. 1244) IF BASE BUS ADDRESS IS NOT 171060.

\*NOTE: USE THE 'B' PROGRAM COMMAND TO MODIFY THIS LOCATION AFTER PROGRAM LOAD.

## 7.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REQUIRES 8K OR MORE). THIS DIAGNOSTIC DOES SUPPORT 'APT' BUT HAS NOT BEEN RUN UNDER IT.

## 7.3 POWER FAIL

A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).

## 7.4 MULTIPLE MNCDA INTERFACE TESTING

THIS PROGRAM DOES 'AUTO-SIZE' THE NUMBER OF MNCDA'S CONNECTED. THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 MNCDA INTERFACES WITH CONTIGUOUS BUS ADDRESSES. THE 'AUTO-SIZE' CAN BE INHIBITED BY THE OPERATOR SETTING BIT 15 OF LOCATION '\$ENV (LOC. 1214) OR SETTING SWITCH REGISTER BIT 12 TO A ONE. USE THE 'B' PROGRAM COMMAND TO LOAD THE BASE ADDRESS.

## 7.5 RESTRICTIONS

NONE

8.0 EXECUTION TIME  
-----

EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS TO ABOUT 20 SECONDS WITH ITERATIONS ENABLED WITH ONE MNCDA CONNECTED. AN END PASS MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS. END OF PASS WILL ALSO REPORT TOTAL ERROR COUNT AND ANY UNIT'S THAT HAD ERRORED.



9.0 PROGRAM TEST DESCRIPTIONS  
-----

## 9.1 L LOGIC TESTS

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE MNCDA DAC CONTROL. A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING. THE PROGRAM WILL AUTO-SIZE UP TO 8 MNCDA'S TO BE TESTED.

## 9.2 R = RAMP LOOP

THIS LOOP IS PROVIDED A METHOD FOR THE OPERATOR TO INSPECT AND VERIFY ANALOG OPERATION OF ALL DAC BITS. THE LOOP ALSO ENABLES THE OPERATOR TO VERIFY THAT NO TWO DAC OUTPUTS ARE INTERCONNECTED.

## 9.3 S - STATIC CALIBRATION LOOP

THIS LOOP PROVIDES THE OPERATOR WITH A SIMPLE LOOP FOR VERIFYING THE INDIVIDUAL DAC BITS AND THE OPERATION OF DAC #3 DIGITAL OUTPUT BITS. THE VALUE OF THE SWITCH REGISTER IS LOADED INTO ALL DAC'S AND THE OUTPUT VOLTAGE CAN BE MONITORED.

## 9.4 D DYNAMIC CALIBRATION LOOP

THIS PROVIDES THE OPERATOR WITH A LOOP THAT LOADS THE VALUE OF THE SWITCH REGISTER INTO THE DAC'S AND THEN AFTER A DELAY CLEARS THE DAC REGISTERS. THIS PROVIDES A SWITCHING PATTERN BETWEEN THE SELECTED VOLTAGE AND 0.

## 9.5 L = LOGIC TEST WITH TESTER SUPPORT (SA 210)

INITIAL PERFORMS THE LOGIC TESTS AND THEN EMPLOYS A KNOWN GOOD A TO D CONVERTER TO AID IN ADJUSTING THE POT'S ON THE MNCDA BOARD. THE OPERATOR IS INFORMED AS TO WHICH POT TO ADJUST AND WHICH D TO A CONVERTER IS TESTED.

10.0 LISTING  
-----

18 BASIC DEFINITIONS  
23 OPERATIONAL SWITCH SETTINGS  
24 TRAP CATCHER  
(1) STARTING ADDRESS(ES)  
31 ACT11 HOOKS  
33 APT PARAMETER BLOCK  
34 COMMON TAGS  
(2) APT MAILBOX-ETABLE  
(1) ERROR POINTER TABLE  
157 INITIALIZE THE COMMON TAGS  
190 TYPE PROGRAM NAME  
(2) GET VALUE FOR SOFTWARE SWITCH REGISTER  
236 SUBROUTINE TO CHANGE BASE ADDRESS OF DEVICE  
248 DETERMINE THE NUMBER OF MNCDA ON THIS SYSTEM

TEST #	DESCRIPTION
-----	-----
295	T1 VERIFY CORRECT I.D. VALUES
296	T2 TEST THAT THE MNCDA RESPONDS TO THE CPU
297	T3 TEST THAT DAC0 REGISTER CAN BE CLEARED
299	T4 TEST THAT DAC0 REGISTER CAN BE LOADED WITH #7777
310	T5 TEST THAT DAC0 REGISTER CAN HOLD A FLOATING 1 PATTERN
322	T6 TEST THAT DAC0 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
330	T7 TEST THE 'SUB' INSTRUCTION WORKS ON DAC0
338	T10 TEST THAT DAC1 REGISTER CAN BE CLEARED
348	T11 TEST THAT DAC #1 REGISTER CAN BE LOADED WITH #7777
349	T12 TEST THAT DAC #1 REGISTER CAN HOLD A FLOATING 1 PATTERN
350	T13 TEST THAT DAC1 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
357	T14 TEST THE 'SUB' INSTRUCTION WORKS ON DAC1
365	T15 TEST THAT THE DAC #2 REGISTER CAN BE CLEARED
374	T16 TEST THAT THE DAC #2 REGISTER CAN BE LOADED WITH #7777
375	T17 TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN
377	T20 TEST THAT DAC2 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
385	T21 TEST THE 'SUB' INSTRUCTION WORKS ON DAC2
393	T22 TEST THAT THE DAC #3 REGISTER CAN BE CLEARED
403	T23 TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777
405	T24 TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN
407	T25 TEST THAT DAC3 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
415	T26 TEST THE 'SUB' INSTRUCTION WORKS ON DAC3
423	T27 TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA
434	T30 TEST THAT RESET CLEARS DAC #0 REGISTER
435	T31 TEST THAT RESET CLEARS DAC #1 REGISTER
437	T32 TEST THAT RESET CLEARS DAC #2 REGISTER
466	T33 TEST THAT RESET CLEARS DAC #3 REGISTER
478	T34 DETERMINE IF MORE MNCDA'S REMAIN TO BE TESTED
489	T35 DETERMINE IF RUNNING ON THE HARDWARE TESTER (IF NOT REPORT END OF PASS)
498	T36 TEST THAT DAC #3 OUTPUT BITS (0-3) FUNCTION
510	T37 VERIFY THE MNCDA +15 SUPPLY
557	T40 VERIFY THE MNCDA -15 SUPPLY
563	T41 DAC0 OFFSET ADJUSTMENT
588	T42 DAC0 GAIN ADJUSTMENT
597	T43 DAC0 CALIBRATION
606	T44 DAC0 2.5 VOLT SWITCH CHECK
(4)	
(4)	
(4)	



(4)	T45	DAC0 10.0 VOLT SWITCH CHECK
(4)	T46	DAC0 UNIPOLAR SWITCH CHECK
608	T47	DAC1 OFFSET ADJUSTMENT
(4)	T50	DAC1 GAIN ADJUSTMENT
(4)	T51	DAC1 CALIBRATION
(4)	T52	DAC1 2.5 VOLT SWITCH CHECK
(4)	T53	DAC1 10.0 VOLT SWITCH CHECK
(4)	T54	DAC1 UNIPOLAR SWITCH CHECK
610	T55	DAC2 OFFSET ADJUSTMENT
(4)	T56	DAC2 GAIN ADJUSTMENT
(4)	T57	DAC2 CALIBRATION
(4)	T60	DAC2 2.5 VOLT SWITCH CHECK
(4)	T61	DAC2 10.0 VOLT SWITCH CHECK
(4)	T62	DAC2 UNIPOLAR SWITCH CHECK
612	T63	DAC3 OFFSET ADJUSTMENT
(4)	T64	DAC3 GAIN ADJUSTMENT
(4)	T65	DAC3 CALIBRATION
(4)	T66	DAC3 2.5 VOLT SWITCH CHECK
(4)	T67	DAC3 10.0 VOLT SWITCH CHECK
(4)	T70	DAC3 UNIPOLAR SWITCH CHECK
614		END OF PASS ROUTINE
630		SUBROUTINE TO ADJUST THE DAC'S OFFSET POTS
663		SUBROUTINE TO ADJUST THE GAIN ADJUSTMENT POTS
691		SUBROUTINE TO TEST THE D/A CALIBRATION
714		SUBROUTINE TO VERIFY THE 2.5 VOLT SWITCH POSITION
736		SUBROUTINE TO VERIFY THE 10.0 VOLT SWITCH POSITION
758		SUBROUTINE TO VERIFY THE UNIPOLAR SWITCH POSITION
785		FIELD TEST MODULE DIGITAL OUTPUT LAMP LOOP
801		SUBROUTINE TO CONVERT CHANNEL N ON THE TESTER A/D
825		SUBROUTINE TO LOOP UNTIL OPERATOR TYPES THE 'RETURN' KEY
832		SUBROUTINE TO COMPARE TWO LOCATIONS BY THE SPREAD
852		DAC ADJUSTMENT ROUTINES
853		-----
855		FULL SCALE RAMP ON EACH RAMP
875		CONTROL CHARACTER DETECTOR
892		STATIC DAC CALIBRATION
904		DYNAMIC DAC CALIBRATION
938		MISC. SUB-ROUTINES, ASCII MESSAGES AND SOFTWARE HANDLERS
943		ASCII MESSAGES
1034		BINARY TO ASCII AND TYPE ROUTINE
1035		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1037		SCOPE HANDLER ROUTINE
1050		ERROR HANDLER ROUTINE
1051		ERROR MESSAGE TIMEOUT ROUTINE
1052		POWER DOWN AND UP ROUTINES
1056		BINARY TO OCTAL (ASCII) AND TYPE
1059		TYPE ROUTINE
1060		TTY INPUT ROUTINE
1062		READ AN OCTAL NUMBER FROM THE TTY
1063		APT COMMUNICATIONS ROUTINE
1066		TRAP DECODER
(3)		TRAP TABLE

```

17      .TITLE CVMND-A MNCAA  DIAGNOSTIC
(1)    .*COPYRIGHT (C) 1978
(1)    .*DIGITAL EQUIPMENT CORP.
(1)    .*MAYNARD, MASS. 01754
(1)    .*
(1)    .*PROGRAM BY RAYMOND SHOOP
(1)    .*
(1)    .*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)    .*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)    .*
18      .SBTTL BASIC DEFINITIONS
(1)
(1)    .*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)    001100  STACK= 1100
(1)    .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
(1)    .EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)    .*MISCELLANEOUS DEFINITIONS
(1)    000011  HT= 11          ;;CODE FOR HORIZONTAL TAB
(1)    000012  LF= 12          ;;CODE FOR LINE FEED
(1)    000015  CR= 15          ;;CODE FOR CARRIAGE RETURN
(1)    000200  CRLF= 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)    177776  PS= 177776     ;;PROCESSOR STATUS WORD
(1)    .EQUIV PS,PSW
(1)    177774  STKLMT= 177774 ;;STACK LIMIT REGISTER
(1)    177772  PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)    177570  DSWR= 177570   ;;HARDWARE SWITCH REGISTER
(1)    177570  DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)
(1)    .*GENERAL PURPOSE REGISTER DEFINITIONS
(1)    000000  R0= %0          ;;GENERAL REGISTER
(1)    000001  R1= %1          ;;GENERAL REGISTER
(1)    000002  R2= %2          ;;GENERAL REGISTER
(1)    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  PR1= 40         ;;PRIORITY LEVEL 1
(1)    000100  PR2= 100       ;;PRIORITY LEVEL 2
(1)    000140  PR3= 140       ;;PRIORITY LEVEL 3
(1)    000200  PR4= 200       ;;PRIORITY LEVEL 4
(1)    000240  PR5= 240       ;;PRIORITY LEVEL 5
(1)    000300  PR6= 300       ;;PRIORITY LEVEL 6
(1)    000340  PR7= 340       ;;PRIORITY LEVEL 7
(1)
(1)    .*'SWITCH REGISTER' SWITCH DEFINITIONS
(1)    100000  SW15= 100000
(1)    040000  SW14= 40000

```

```
(1) 020000 SW13= 20000
(1) 010000 SW12= 10000
(1) 004000 SW11= 4000
(1) 002000 SW10= 2000
(1) 001000 SW09= 1000
(1) 000400 SW08= 400
(1) 000200 SW07= 200
(1) 000100 SW06= 100
(1) 000040 SW05= 40
(1) 000020 SW04= 20
(1) 000010 SW03= 10
(1) 000004 SW02= 4
(1) 000002 SW01= 2
(1) 000001 SW00= 1
(1) .EQUIV SW09,SW9
(1) .EQUIV SW08,SW8
(1) .EQUIV SW07,SW7
(1) .EQUIV SW06,SW6
(1) .EQUIV SW05,SW5
(1) .EQUIV SW04,SW4
(1) .EQUIV SW03,SW3
(1) .EQUIV SW02,SW2
(1) .EQUIV SW01,SW1
(1) .EQUIV SW00,SW0
(1)
(1) ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1) 100000 BIT15= 100000
(1) 040000 BIT14= 40000
(1) 020000 BIT13= 20000
(1) 010000 BIT12= 10000
(1) 004000 BIT11= 4000
(1) 002000 BIT10= 2000
(1) 001000 BIT09= 1000
(1) 000400 BIT08= 400
(1) 000200 BIT07= 200
(1) 000100 BIT06= 100
(1) 000040 BIT05= 40
(1) 000020 BIT04= 20
(1) 000010 BIT03= 10
(1) 000004 BIT02= 4
(1) 000002 BIT01= 2
(1) 000001 BIT00= 1
(1) .EQUIV BIT09,BIT9
(1) .EQUIV BIT08,BIT8
(1) .EQUIV BIT07,BIT7
(1) .EQUIV BIT06,BIT6
(1) .EQUIV BIT05,BIT5
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0
(1)
(1) ;*BASIC 'CPU' TRAP VECTOR ADDRESSES
```

(1)	000004	ERRVEC= 4	::TIME OUT AND OTHER ERRORS
(1)	000010	RESVEC= 10	::RESERVED AND ILLEGAL INSTRUCTIONS
(1)	000014	TBITVEC=14	::'T' BIT
(1)	000014	TRTVEC= 14	::TRACE TRAP
(1)	000014	BPTVEC= 14	::BREAKPOINT TRAP (BPT)
(1)	000020	IOTVEC= 20	::INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)	000024	PWRVEC= 24	::POWER FAIL
(1)	000030	EMTVEC= 30	::EMULATOR TRAP (EMT) **ERROR**
(1)	000034	TRAPVEC=34	::'TRAP' TRAP
(1)	000060	TKVEC= 60	::TTY KEYBOARD VECTOR
(1)	000064	TPVEC= 64	::TTY PRINTER VECTOR
(1)	000240	PIRQVEC-240	::PROGRAM INTERRUPT REQUEST VECTOR
19			
20	171060	ABASE-171060	

```

22
23      .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              INHIBIT SIZING # OF MNCAA'S
(1)      : *      11              INHIBIT ITERATIONS
(1)      : *      9              LOOP ON ERROR
(1)      : *      8              LOOP ON TEST IN SWR<7:0>
24      .SBTTL TRAP CATCHER
(1)
(1)      000000      .=0
(1)      : *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ''+2,HALT''
(1)      : *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)      : *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)      000174      .=174
(1) 000174 000000  DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000  SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
(1)      .SBTTL STARTING ADDRESS(ES)
(1) 000200 000137 001502  JMP @WBEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
25 000204 000137 001466  JMP RESTRT      ;RESTART ADDRESS
26 000210 000137 001474  JMP TESTER      ;JUMP TO TESTER SA.
27
28
29 000100 000104 000200 000002  .=100
      104,200,2      ;B EVENT SAFE GUARD
  
```

```

31      .SBTTL  ACT11 HOOKS
(1)
(2)      ;:*****
(1)      ;HOOKS REQUIRED BY ACT11
(1)      $SVPC=      ;SAVE PC
(1)      .=46
(1) 000046 000046  $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)      .=52
(1) 000052 000000  .WORD 0      ;;2)SET LOC.52 TO ZERO
(1)      .=$SVPC      ;; RESTORE PC
(1)      .=1000
32      001000
33

```

```

(1)      .SBTTL  APT PARAMETER BLOCK
(2)      ;:*****
(1)      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      ;:*****
(1)      .SX-      ;;SAVE CURRENT LOCATION
(1)      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200  200      ;;FOR APT START UP
(1)      .-44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000  $APTHDR ;;POINT TO APT HEADER BLOCK
(1)      .=$X      ;;RESET LOCATION COUNTER

```

```

(2)      ;:*****
(1)      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)      ;INTERFACE SPEC.

```

```

(1) 001000  $APTHD:
(1) 001000 000000 $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001170 $MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000030 $STSM: .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 000031 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```





(2)	001204	000000	\$MSGAD: .WORD	AMSGAD	::MESSAGE ADDRESS
(2)	001206	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
(2)	001210		\$ETABLE:		::APT ENVIRONMENT TABLE
(2)	001210	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
(2)	001211	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
(2)	001212	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
(2)	001214	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
(2)	001216	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)	001220	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001221	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)	001222	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			:*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001224	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001225	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
(2)	001226	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001230	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001231	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
(2)	001232	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001234	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001235	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
(2)	001236	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001240	000000	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001242	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001244	171060	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001246	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
(2)	001250	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001252		\$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) 001252 $ERRTB:
35
36
37 ;ITEM 1
38 001252 010432 EM1 ;BUT TIME-OUT WHEN REF. A DAC ADDRESS
39 001254 012154 DH2 ;ERRPC BUSADR
40 001256 013650 DT1 ;$ERRPC $BDDAT
41 001260 013770 DF0
42 ;ITEM 2
43 001262 010531 EM2 ;DAC #0 REGISTER IN ERROR
44 001264 012114 DH1 ;ERRPC BUSADR GOOD BAD
45 001266 013660 DT2 ;$ERRPC DAC0 $GDDAT $BDDAT
46 001270 013770 DF0
47 ;ITEM 3
48 001272 010574 EM3 ;DAC #1 REGISTER IN ERROR
49 001274 012114 DH1 ;ERRPC BUSADR GOOD BAD
50 001276 013674 DT3 ;$ERRPC DAC1 $GDDAT $BDDAT
51 001300 013770 DF0
52 ;ITEM 4
53 001302 010637 EM4 ;DAC #2 REGISTER IN ERROR
54 001304 012114 DH1 ;ERRPC BUSADR GOOD BAD
55 001306 013710 DT4 ;$ERRPC DAC2 $GDDAT $BDDAT
56 001310 013770 DF0
57 ;ITEM 5
58 001312 010702 EM5 ;DAC #3 REGISTER IN ERROR
59 001314 012114 DH1 ;ERRPC BUSADR GOOD BAD
60 001316 013724 DT5 ;$ERRPC DAC3 $GDDAT $BDDAT
61 001320 013770 DF0
62 ;ITEM 6
63 001322 010745 EM6 ;SELECTED DAC OFFSET POT IS NOT ADJUSTED CORRECTLY
64 001324 012176 DH6 ;ERRPC BUSADR EXPECT WAS SPREAD
65 001326 013740 DT6 ;$ERRPC DACBAD $GDDAT $BDDAT SPREAD
66 001330 013770 DF0
67 ;ITEM 7
68 001332 011034 EM7 ;SELECTED DAC GAIN POT IS NOT ADJUSTED CORRECTLY
69 001334 012176 DH6 ;ERRPC BUSADR EXPECT WAS SPREAD
70 001336 013740 DT6 ;$ERRPC DACBAD $GDDAT $BDDAT SPREAD
71 001340 013770 DF0

```

73					:ITEM 10	
74	001342	011121			EM10	:SELECTED DAC HAS A LINEARITY PROBLEM
75	001344	012176			DH6	:ERRPC BUSADR EXPECT WAS SPREAD
76	001346	013740			DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
77	001350	013770			DF0	
78					:ITEM 11	
79	001352	011174			EM11	:+15 VOLT SUPPLY IS INCORRECT
80	001354	012176			DH6	:ERRPC BUSADR EXPECT WAS SPREAD
81	001356	013740			DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
82	001360	013770			DF0	
83					:ITEM 12	
84	001362	011237			EM12	:-15 VOLT SUPPLY IS INCORRECT
85	001364	012176			DH6	:ERRPC BUSADR EXPECT WAS SPREAD
86	001366	013740			DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
87	001370	013770			DF0	
88					:ITEM 13	
89	001372	011302			EM13	:DAC #3 DIGITAL OUTPUT BITS IN ERROR
90	001374	012114			DH1	:ERRPC BUSADR GOOD BAD
91	001376	013724			DT5	:\$ERRPC DAC3 \$GDDAT \$BDDAT
92	001400	013770			DF0	
93					:ITEM 14	
94	001402	000000	000000	000000	0,0,0,0	:CALL IS NOT USED
95					:ITEM 15	
96	001412	011402			EM15	:INCORRECT I.D. VALUE CODE
97	001414	012114			DH1	:ERRPC BUSADR GOOD BAD
98	001416	013660			DT2	:\$ERRPC DAC0 \$GDDAT \$BDDAT
99	001420	013770			DF0	
100					:ITEM 16	
101	001422	011442			EM16	:SELECTED DAC HAS A FRONT PANEL SWITCH PROBLEM
102	001424	012176			DH6	:ERRPC BUSADR EXPECT WAS SPREAD
103	001426	013740			DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
104	001430	013770			DF0	
105					:ITEM 17	
106	001432	011520	012242	013756	EM17,DH17,DT17,DF0	:EXISTING UNIT NOW FAILS TO RESPOND
107						
108	001442	000010			VADDR: 10	:OFFSET TO NEXT MNCDA ADDRESS
109	001444	000000			EVER: 0	:ADDITIONAL UNIT CONTROL WORD
110	001446	171060			DAC0: ABASE	
111	001450	171062			DAC1: ABASE+2	
112	001452	171064			DAC2: ABASE+4	
113	001454	171066			DAC3: ABASE+6	
114						
115	001456	167772			TSTR2: 167772	:SNOW WHITE TESTER ADDRESSES
116	001460	167774			DRIN: 167774	
117	001462	170400			ADCS: 170400	:KNOWN GOOD A/D ADDRESSES
118	001464	170402			ADBR: 170402	

```

148 001466 005237 001444      RESTRT: INC      EVER          ;INDICATE RESTART
149 001472 000411              BR      RSTRT
150 001474 005237 010422      TESTER: INC     WFTST         ;INDICATE TESTER MODE
151 001500 000402              BR      BEGIN1
152 001502 005037 010422      BEGIN: CLR     WFTST
153 001506 005037 010410      BEGIN1: CLR    TEMP
154 001512 005037 001444      CLR     EVER
155 001516 000005      RSTRT: RESET
157      .SBTTL INITIALIZE THE COMMON TAGS
(1)      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001520 012706 001100      MOV     # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 001524 005026              CLR     (R6)+           ;;CLEAR MEMORY LOCATION
(1) 001526 022706 001140      CMP     #SWR,R6 ;;DONE?
(1) 001532 001374              BNE    -6               ;;LOOP BACK IF NO
(1) 001534 012706 001100      MOV     #STACK,SP      ;;SETUP THE STACK POINTER
(1)      ;;INITIALIZE A FEW VECTORS
(1) 001540 012737 014300 000020      MOV     # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001546 012737 000340 000022      MOV     #340,@#IOTVEC+2 ;;LEVEL 7
(1) 001554 012737 014564 000030      MOV     # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001562 012737 000340 000032      MOV     #340,@#EMTVEC+2 ;;LEVEL 7
(1) 001570 012737 020022 000034      MOV     # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001576 012737 000340 000036      MOV     #340,@#TRAPVEC+2;LEVEL 7
(1) 001604 012737 015172 000024      MOV     # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 001612 012737 000340 000026      MOV     #340,@#PWRVEC+2 ;;LEVEL 7
(1) 001620 013737 006470 006462      MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
(1) 001626 005037 001160      CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001632 005037 001162      CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001636 112737 000001 001115      MOV     #1,$SERMAX      ;;ALLOW ONE ERROR PER TEST
(1) 001644 012737 001644 001106      MOV     #,$SLPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001652 012737 001652 001110      MOV     #,$SLPERR       ;;SETUP THE ERROR LOOP ADDRESS
(2)      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)      ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001660 013746 000004              MOV     @#ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
(2) 001664 012737 001720 000004      MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
(2) 001672 012737 177570 001140      MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001700 012737 177570 001142      MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
(2) 001706 022777 177777 177224      CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
(2) 001714 001012              BNE    66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)      ;;AND THE HARDWARE SWR IS NOT = -1
(2) 001716 000403              BR     65$              ;;BRANCH IF NO TIMEOUT
(2) 001720 012716 001726      64$: MOV     #65$,(SP)     ;;SET UP FOR TRAP RETURN
(2) 001724 000002              RTI
(2) 001726 012737 000176 001140      65$: MOV     #SWREG,SWR    ;;POINT TO SOFTWARE SWR
(2) 001734 012737 000174 001142      MOV     #DISPREG,DISPLAY
(2) 001742 012637 000004      66$: MOV     (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 001746 005037 001176              CLR     $PASS           ;;CLEAR PASS COUNT
(2) 001752 132737 000200 001211      BIT    #APTSIZE,$ENVM   ;;TEST USER SIZE UNDER APT
(2) 001760 001403              BEQ    67$              ;;YES,USE NON-APT SWITCH
(2) 001762 012737 001212 001140      MOV     # $SWREG,SWR    ;;NO,USE APT SWITCH REGISTER
(2) 001770      67$:
158 001770 005046              CLR     -(SP)           ;LOWER PRIORITY
159 001772 012746 002000      MOV     #1$,-(SP)
160 001776 000002              RTI

```

```

162 ;OVERLAY THE FIRST 4 LOC. OF THE $TYPE ROUTINE TO ENABLE LEVEL 0 INTERRUPT
163 002000 012737 005046 015640 1$: MOV #5046,$TYPE ;CLR -(SP)
164 002006 012737 012746 015642 MOV #12746,$TYPE+2 ;MOV # $TYPE+12,-(SP)
165 002014 012737 015652 015644 MOV # $TYPE+12,$TYPE+4 ;
166 002022 012737 000002 015646 MOV #RTI,$TYPE+6 ;RTI
167 002030 004737 016170 JSR PC,$TKINT ;INIT THE KEYBOARD INTR. HANDLER
168 002034 005037 010404 CLR BADUNT ;RESET BAD INDICATOR
169 002040 000137 002114 JMP INIT1
170 ;SUBROUTINE TO LOAD DEVICE ADDRESSES LOCATIONS
171 002044 013700 001244 LDTRAP: MOV $BASE,R0 ;GET BASE ADDRESS
172 002050 010037 001446 MOV R0,DAC0 ;LOAD X ADDRESS
173 002054 010037 001450 MOV R0,DAC1 ;LOAD Y ADDRESS
174 002060 010037 001452 MOV R0,DAC2 ;LOAD DAC #2
175 002064 010037 001454 MOV R0,DAC3 ;LOAD DAC #3
176 002070 062737 000002 001450 ADD #2,DAC1
177 002076 062737 000004 001452 ADD #4,DAC2
178 002104 062737 000006 001454 ADD #6,DAC3
179 002112 000207 RTS PC ;EXIT
180
181 002114 004737 002044 INIT1: JSR PC,LDTRAP ;FIX BUS ADDRESSES
182 002120 005737 010410 TST TEMP ;TEST IF START OR RESTART
183 002124 001073 BNE MTEST1 ;RESTART
184 002126 005737 010422 1$: TST WFTST ;TEST IF ON TESTER
185 002132 001406 BEQ 2$ ;BR IF NOT
186 002134 104401 TYPE
187 002136 011647 MSGSW ;TELL OPERATOR ABOUT TESTER SWITCHES
188 002140 104401 011723 TYPE, FRONT ;INFORM THE OPEATOR ABOUT TESTER SWITCHES
189 002144 004737 010020 JSR PC,CSPACE ;WAIT FOR OPERATOR
190 002150 2$:
(1) .SBTTL TYPE PROGRAM NAME
(1) ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002150 005227 177777 INC #-1 ;;FIRST TIME?
(1) 002154 001050 BNE 64$ ;;BRANCH IF NO
(1) 002156 022737 006522 000042 CMP # $ENDAD,@#42 ;;ACT-11?
(1) 002164 001444 BEQ 64$ ;;BRANCH IF YES
(1) 002166 104401 002234 TYPE ,65$ ;;TYPE ASCIZ STRING
(2) .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002172 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002176 001012 BNE 66$ ;;BRANCH IF YES
(2) 002200 123727 001210 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
(2) 002206 001406 BEQ 66$ ;;BRANCH IF YES
(2) 002210 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
(2) 002216 001005 BNE 67$ ;;BRANCH IF NO
(2) 002220 104407 GTSWR ;;GET SOFT-SWR SETTINGS
(2) 002222 000403 BR 67$
(2) 002224 112737 000001 001134 66$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
(2) 002232 67$:
(1) 002232 000421 BR 64$ ;;GET OVER THE ASCIZ
(1) ;;65$: .ASCIZ <CRLF>#CVMND-A MNCAA (D/A) DIAGNOSTIC#<CRLF>
(1) 64$:
191 002276 005737 001134 TST $AUTOB ;TEST IF UNDER A MONITOR
192 002302 001402 BEQ MTEST ;BR IF NO
193 002304 000137 002600 JMP LOGIC ;RUN LOGIC TEST

```

```

195      ;.SBTTL KEYBOARD COMMAND DECODER
196 002310 104401 013137      MTEST: TYPE, PRIME0      ; TELL THE OPER. THE TEST AVAIL.
197 002314 000005      MTEST1: RESET
198 002316 052777 000100 176620  BIS #BIT6,@$TKS      ;REENABLE KRB INTR.
199 002324 005037 001176      CLR $PASS      ;CLEAR PASS COUNTER
200 002330 005037 001112      CLR $ERTTL      ;CLEAR TOTAL ERROR COUNT
201 002334 005037 001444      CLR EVER      ;CLEAR INITIAL UNIT TYPEOUT
202 002340 004737 002044      JSR PC,LDTRAP      ;ENSURE BASE ADDRESS IS LOADED
203 002344 104401 013516      TYPE, DOT      ;ASK THE OPERATOR FOR TEST CHAR.
204 002350 104412      RDLIN      ;GET OPER. INPUT
205 002352 013637 002534      MOV @($P)+,RUNIT      ;SAVE THE FIRST CHARACTER
206 002356 142737 000040 002534  BICB #40,RUNIT      ;MASK OFF LOWER CASE BIT
207 002364 122737 000102 002534  CMPB #'B,RUNIT      ;TEST IF 'B'
208 002372 001002      BNE 1$      ;BR IF NOT
209 002374 000137 002540      JMP BASEXC      ;CHANGE BASE ADDRESS
210 002400 122737 000104 002534 1$:  CMPB #'D,RUNIT      ;TEST IF 'D'
211 002406 001002      BNE 2$      ;BR IF NOT
212 002410 000137 010314      JMP DYNCAL      ;DO DYNAMIC CALIB.
213 002414 122737 000110 002534 2$:  CMPB #'H,RUNIT      ;TEST IF 'H'
214 002422 001002      BNE 3$      ;BR IF NOT
215 002424 000137 002310      JMP MTEST      ;RETYPE THE LIST
216 002430 122737 000114 002534 3$:  CMPB #'L,RUNIT      ;TEST IF 'L'
217 002436 001002      BNE 4$      ;BR IF NOT
218 002440 000137 002600      JMP LOGIC      ;DO LOGIC TEST
219 002444 122737 000117 002534 4$:  CMPB #'O,RUNIT      ;TEST IF 'O'
220 002452 001002      BNE 5$      ;BR IF NOT
221 002454 000137 007630      JMP LAMPS      ;DO LAMP OUTPUT PATTERN
222 002460 122737 000122 002534 5$:  CMPB #'R,RUNIT      ;TEST IF 'R'
223 002466 001002      BNE 6$      ;BR IF NOT
224 002470 000137 010102      JMP FULRMP      ;DO RAMP TEST
225 002474 122737 000123 002534 6$:  CMPB #'S,RUNIT      ;TEST IF 'S'
226 002502 001002      BNE 7$      ;BR IF NOT
227 002504 000137 010252      JMP STATIC      ;DO STATIC CALIB.
228 002510 122737 000107 002534 7$:  CMPB #'G,RUNIT      ;TEST IF 'G'
229 002516 001002      BNE 77$      ;BR IF NOT
230 002520 104407      GTSWR      ;GET SWITCHES
231 002522 000674      BR MTEST1      ;RETYPE THE DOT
232 002524 104401 001164      77$: TYPE, $QUES      ;OPER. HAS FAT FINGERS
233 002530 000137 002314      JMP MTEST1      ;RETYPE 'DOT'
234 002534 000000      RUNIT: 0      ;OPERATOR INPUT CHARACTER
235 002536 000000      UNITBD: 0      ;BAD UNIT NUMBER
236      ;.SBTTL SUBROUTINE TO CHANGE BASE ADDRESS OF DEVICE
237 002540 104401 013606      BASEXC: TYPE, PRIMBA      ;TELL OPERATOR THE HEADER
238 002544 013746 001244      MOV $BASE,-($P)      ;PUSH ON STACK
239 002550 104402      TYPOC      ;TELL OPERATOR CURRENT VALUE
240 002552 104401 013643      TYPE, PRIMBB      ;FINISH THE MESSAGE
241 002556 104413      RDOCT      ;GET OPER. INPUT
242 002560 012600      MOV ($P)+,R0      ;GET #
243 002562 001404      BEQ 1$      ;BR IF C/R
244 002564 010037 001244      MOV R0,$BASE      ;LOAD NEW VALUE
245 002570 004737 002044      JSR PC,LDTRAP      ;RELOAD ADDRESS VALUES
246 002574 000137 002314      1$: JMP MTEST1      ;RETYPE THE DOT

```

```

248 .SBTTL DETERMINE THE NUMBER OF MNCDAs ON THIS SYSTEM
249 002600 013737 001244 001126 LOGIC: MOV $BASE,$BDDAT ;GET THE BASE ADDRESS
250 002606 005037 010406 CLR MASKNM
251 002612 005037 001202 CLR $UNIT ;CLEAR UNIT #
252 002616 012737 002672 000004 MOV #2$,ERRVEC ;LOAD TRAP RETURN
253 002624 005777 176276 1$: TST @BDDAT ;TEST IF ADDR EXISTS
254 002630 063737 001442 001126 ADD VADDR,$BDDAT ;UPDATE THE BUS ADDRESS
255 002636 005237 001202 INC $UNIT ;UPDATE UNIT COUNT
256 002642 005737 001210 TST $ENV ;TEST IF 'DO NOT SIZE'
257 002646 100423 BMI 3$ ;BR IF NO SIZEING
258 002650 032777 010000 176262 BIT #SW12,@SWR ;TEST IF SW12 IS SET
259 002656 001017 BNE 3$ ;BR AND DONOT SIZE # OF UNITS
260 002660 022737 000010 001202 CMP #8,$UNIT ;TEST IF MAX. NUMBER
261 002666 001356 BNE 1$ ;BR IF NOT
262 002670 000412 BR 3$ ;BR IF MAX.
263 002672 022626 2$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
264 002674 005737 001202 TST $UNIT ;TEST IF ANY EXIST
265 002700 001006 BNE 3$ ;BR IF SOME ARE THERE
266 002702 005737 000042 TST @#42 ;TEST IF XXDP CHAIN MODE
267 002706 001003 BNE 3$ ;BR IF YES
268 002710 104001 ERROR 1 ;BASE ADDRESS CAUSED AN BUS TRAP
269 002712 000137 006434 JMP $EOP ;GO TO END OF PASS
270 002716 005737 001444 3$: TST EVER ;TEST IF # HAS BEEN REPORTED
271 002722 100426 BMI 4$ ;BR IF IT HAS
272 002724 005737 010422 TST WFTST ;TEST IF TESTER MODE
273 002730 001014 BNE 6$ ;BR IF TESTER
274 002732 104401 TYPE
275 002734 012046 FOUND1 ;TELL OPERATOR THE # OF MNCDAs
276 002736 013746 001202 MOV $UNIT,-(SP)
277 002742 104405 TYPDS ;TELL OPER. THE DEC. VALUE
278 002744 104401 TYPE
279 002746 012072 FOUND2
280 002750 005737 001202 TST $UNIT ;ANY UNITS
281 002754 001002 BNE 6$ ;BR IF SOME
282 002756 000137 006434 JMP $EOP ;REPORT EOP
283 002762 013737 001202 001444 6$: MOV $UNIT,EVER ;SAVE THE # OF MNCDAs FOR LATER
284 002770 052737 100000 001444 BIS #BIT15,EVER ;SET 'REPORTED # FLAG'
285 002776 000405 BR 5$ ;
286 003000 123737 001444 001202 4$: CMPB EVER,$UNIT ;TEST IF ANY HAVE GONE AWAY
287 003006 001401 BEQ 5$ ;BR IF ALL ARE STILL HERE
288 003010 104017 ERROR 17 ;EXISTING UNIT FAILED TO RESPOND NOW
289 003012 005037 001202 5$: CLR $UNIT ;RESET UNIT POINTER
290 003016 005037 001200 CLR $DEVCT ;MAKE APT HAPPY
291 003022 004737 002044 JSR PC,LDTRAP ;LOAD BUS ADDRESSES
292 003026 012737 000001 010406 MOV #BIT0,MASKNM ;LOAD MASK NUMBER IF ERROR
299 ;*****
(3) ;*TEST 1 VERIFY CORRECT I.D. VALUES
(3) ;*****
(2) TST1: SCOPE
300 003036 005737 010422 TST WFTST ;CHECK IF ON TESTER
301 003042 001420 BEQ TST2 ;BR IF NOT ON TESTER
302 003044 005077 176406 CLR @TSTR2 ;ENSURE TESTER MODE
303 003050 012737 000100 001124 MOV #100,$GDDAT ;LOAD EXPECTED I.D. VALUE
304 003056 017737 176376 001126 MOV @DRIN,$BDDAT ;READ I.D. LINES

```



305	003064	042737	177417	001126	BIC	#177417,\$BDDAT	:MASK OUT OTHER BITS
306	003072	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE
307	003100	001401			BEQ	1ST2	::BR IF SAME
308	003102	104015			ERROR	15	:INCORRECT I.D. VALUE

```

310          :*****
(3)          :*TEST 2      TEST THAT THE MNCDA RESPONDS TO THE CPU
(3)          :*****
(2) 003104 000004 TST2:  SCOPE
311 003106 012737 003136 000004  MOV    #1$,ERRVEC      ;LOAD BUS TRAP RETURN
312 003114 005777 176326          TST    @DAC0           ;TEST DAC #0
313 003120 005777 176324          TST    @DAC1           ;TEST DAC #1
314 003124 005777 176322          TST    @DAC2           ;TEST DAC #2
315 003130 005777 176320          TST    @DAC3           ;TEST DAC #3
316 003134 000407          BR     2$              ;BR AND RESTORE LOC. 4
317 003136 022626 1$:  CMP    (SP)+,(SP)+    ;CLEAN THE STACK
318 003140 104001          ERROR  1              ;ERROR, BUS TIMEOUT WHEN ADDRESSING THE MNCDA
319 003142 012737 000006 000004  MOV    #6,ERRVEC      ;LOAD LOC 4
320 003150 000137 005334          JMP    REMAIN          ;TEST IF ANY OTHER'S
321 003154 012737 000006 000004  2$:  MOV    #6,ERRVEC      ;LOAD RETURN
322          :*****
(3)          :*TEST 3      TEST THAT DACO REGISTER CAN BE CLEARED
(3)          :*****
(2) 003162 000004 TST3:  SCOPE
323 003164 005037 001124          CLR    $GDDAT          ;LOAD EXPECTED
324 003170 013777 001124 176250  MOV    $GDDAT,@DAC0    ;LOAD REG
325 003176 017737 176244 001126  MOV    @DAC0,$BDDAT    ;READ REG
326 003204 023737 001124 001126  CMP    $GDDAT,$BDDAT   ;COMPARE
327 003212 001401          BEQ    TST4            ;;BR IF EQUAL
328 003214 104002          ERROR  2              ;ERROR, DACO REGISTER NOT = 0
329
330          :*****
(3)          :*TEST 4      TEST THAT DACO REGISTER CAN BE LOADED WITH #7777
(3)          :*****
(2) 003216 000004 TST4:  SCOPE
331 003220 012737 007777 001124  MOV    #7777,$GDDAT    ;LOAD EXPECTED
332 003226 013777 001124 176212  MOV    $GDDAT,@DAC0    ;LOAD REG
333 003234 017737 176206 001126  MOV    @DAC0,$BDDAT    ;READ REG
334 003242 023737 001124 001126  CMP    $GDDAT,$BDDAT   ;COMPARE
335 003250 001401          BEQ    TST5            ;;BR IF EQUAL
336 003252 104002          ERROR  2              ;ERROR, DACO REGISTER NOT = 7777
337
338          :*****
(3)          :*TEST 5      TEST THAT DACO REGISTER CAN HOLD A FLOATING 1 PATTERN
(3)          :*****
(2) 003254 000004 TST5:  SCOPE
(1) 003256 012737 000100 001160  MOV    #100,$TIMES     ;;DO 100 ITERATIONS
339 003264 012737 004000 001124  MOV    #BIT11,$GDDAT   ;LOAD EXPECTED
340 003272 013777 001124 176146  1$:  MOV    $GDDAT,@DAC0    ;LOAD DACO REGISTER
341 003300 017737 176142 001126  MOV    @DAC0,$BDDAT    ;READ THE REGISTER
342 003306 023737 001124 001126  CMP    $GDDAT,$BDDAT   ;COMPARE THE DATA
343 003314 001401          BEQ    2$              ;;BR IF SAME
344 003316 104002          ERROR  2              ;ERROR, DACO REGISTER FAILED TO HOLD A FLOATING
345 003320 006237 001124  2$:  ASR    $GDDAT          ;CHANGE THE DATA
346 003324 001362          BNE    1$              ;BR AND TEST MORE DATA

```

```
348
(4)
(4)
(3) 003326 000004
(2) 003330 012737 000100 001160
(1) 003336 012737 004000 001124
(1) 003344 013777 001124 176074
(1) 003352 017737 176070 001126
(1) 003360 023737 001124 001126
(2) 003366 001407
(1) 003370 017737 176052 001126
(1) 003376 104002
(1) 003400 013777 001124 176040
(1) 003406 006277 176034
(1) 003412 006237 001124
(1) 003416 001355

*****
*TEST 6 TEST THAT DACO CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
*****
TST6: SCOPE
      MOV #100,$TIMES ;;DO 100 ITERATIONS
      MOV #BIT11,$GDDAT ;LOAD EXPECTED
      MOV $GDDAT,@DACO ;LOAD DACO
1$:   MOV @DACO,$BDDAT ;READ THE REGISTER
      CMP $GDDAT,$BDDAT ;COMPARE THE GOOD TO DACO
      BEQ 2$ ;;BR IF THE SAME
      MOV @DACO,$BDDAT ;SAVE FOR TYPEOUT
      ERROR 2 ;DACO FAILED TO HOLD A FLOATING 1 PATTERN
      MOV $GDDAT,@DACO ;LOAD DACO AGAIN
2$:   ASR @DACO ;CHANGE THE DATA
      ASR $GDDAT ;CHANGE THE EXPECTED
      BNE 1$ ;BR IF MORE DATA

*****
*TEST 7 TEST THE 'SUB' INSTRUCTION WORKS ON DACO
*****
TST7: SCOPE
      MOV #10,$TIMES ;;DO 10 ITERATIONS
      MOV #7777,$GDDAT ;LOAD EXPECTED
      MOV $GDDAT,@DACO ;LOAD DACO
1$:   SUB #1,$GDDAT ;SUB A VALUE
      SUB #1,@DACO ;FROM EXPECTED AND DACO
      MOV @DACO,$BDDAT ;READ THE REGISTER
      CMP $GDDAT,$BDDAT ;COMPARE
      BEQ 2$ ;;BR IF SAME
      ERROR 2 ;THE SUB INSTRUCTION FAILED ON DACO
      MOV $GDDAT,@DACO ;LOAD THE REGISTER AGAIN
2$:   TST $GDDAT ;TEST FOR MORE DATA
      BNE 1$ ;;BR IF MORE DATA

*****
*TEST 10 TEST THAT DAC1 REGISTER CAN BE CLEARED
*****
TST10: SCOPE
      CLR $GDDAT ;LOAD EXPECTED
      MOV $GDDAT,@DAC1 ;LOAD DAC1
      MOV @DAC1,$BDDAT ;READ REG
      CMP $GDDAT,$BDDAT ;COMPARE
      BEQ TST11 ;;BR IF EQUAL
      ERROR 3 ;ERROR, DAC1 REGISTER NOT = 0

*****
*TEST 11 TEST THAT DAC #1 REGISTER CAN BE LOADED WITH #7777
*****
TST11: SCOPE
      MOV #7777,$GDDAT ;LOAD EXPECTED
      MOV $GDDAT,@DAC1 ;LOAD DAC #1
      MOV @DAC1,$BDDAT ;READ REG
      CMP $GDDAT,$BDDAT ;COMPARE
      BEQ TST12 ;;BR IF EQUAL
      ERROR 3 ;ERROR, DAC1 REGISTER NOT = 7777

349
(4)
(4)
(3) 003420 000004
(2) 003422 012737 000010 001160
(1) 003430 012737 007777 001124
(1) 003436 013777 001124 176002
(1) 003444 162737 000001 001124
(1) 003452 162777 000001 175766
(1) 003460 017737 175762 001126
(1) 003466 023737 001124 001126
(2) 003474 001404
(1) 003476 104002
(1) 003500 013777 001124 175740
(1) 003506 005737 001124
(2) 003512 001354

350
(3)
(3)
(2) 003514 000004
351 003516 005037 001124
352 003522 013777 001124 175720
353 003530 017737 175714 001126
354 003536 023737 001124 001126
355 003544 001401
356 003546 104003

357
(3)
(3)
(2) 003550 000004
358 003552 012737 007777 001124
359 003560 013777 001124 175662
360 003566 017737 175656 001126
361 003574 023737 001124 001126
362 003602 001401
363 003604 104003
```

```
365      ::*****  
(3)      ::*TEST 12      TEST THAT DAC #1 REGISTER CAN HOLD A FLOATING 1 PATTERN  
(3)      ::*****  
(2) 003606 000004  
(1) 003610 012737 000100 001160  
366 003616 012737 004000 001124  
367 003624 013777 001124 175616 1$:  MOV    #100,$TIMES      ;;DO 100 ITERATIONS  
368 003632 017737 175612 001126 1$:  MOV    #BIT11,$GDDAT      ;LOAD EXPECTED  
369 003640 023737 001124 001126 1$:  MOV    $GDDAT,@DAC1      ;LOAD THE REGISTER  
370 003646 001401 175612 001126 1$:  MOV    @DAC1,$BDDAT      ;READ THE REGISTER  
371 003650 104003 001124 001126 1$:  CMP    $GDDAT,$BDDAT      ;COMPARE THE DATA  
372 003652 006237 001124 001126 1$:  BEQ    2$                ;;BR IF DATA IS SAME  
373 003656 001362 001124 001126 1$:  ERROR  3                ;ERROR, DAC #1 REGISTER FAILED TO HOLD A FLOATIN  
374      2$:  ASR    $GDDAT      ;CHANGE THE DATA  
375      2$:  BNE    1$                ;BR AND TEST MORE DATA  
(4)      ::*****  
(4)      ::*TEST 13      TEST THAT DAC1 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)  
(4)      ::*****  
(3) 003660 000004  
(2) 003662 012737 000100 001160  
(1) 003670 012737 004000 001124  
(1) 003676 013777 001124 175544  
(1) 003704 017737 175540 001126 1$:  MOV    #100,$TIMES      ;;DO 100 ITERATIONS  
(1) 003712 023737 001124 001126 1$:  MOV    #BIT11,$GDDAT      ;LOAD EXPECTED  
(2) 003720 001407 175540 001126 1$:  MOV    $GDDAT,@DAC1      ;LOAD DAC1  
(1) 003722 017737 175522 001126 1$:  MOV    @DAC1,$BDDAT      ;READ THE REGISTER  
(1) 003730 104003 001124 175510 1$:  CMP    $GDDAT,$BDDAT      ;COMPARE THE GOOD TO DAC1  
(1) 003732 013777 001124 175510 1$:  BEQ    2$                ;;BR IF THE SAME  
(1) 003740 006277 175504 001124 1$:  MOV    @DAC1,$BDDAT      ;SAVE FOR TYPEOUT  
(1) 003744 006237 001124 001124 1$:  ERROR  3                ;DAC1 FAILED TO HOLD A FLOATING 1 PATTERN  
(1) 003750 001355 001124 001124 1$:  MOV    $GDDAT,@DAC1      ;LOAD DAC1 AGAIN  
375      2$:  ASR    @DAC1      ;CHANGE THE DATA  
(4)      2$:  ASR    $GDDAT      ;CHANGE THE EXPECTED  
(4)      2$:  BNE    1$                ;BR IF MORE DATA  
(3) 003752 000004  
(2) 003754 012737 000010 001160  
(1) 003762 012737 007777 001124  
(1) 003770 013777 001124 175452  
(1) 003776 162737 000001 001124 1$:  MOV    #10,$TIMES      ;;DO 10 ITERATIONS  
(1) 004004 162777 000001 175436 1$:  MOV    #7777,$GDDAT      ;LOAD EXPECTED  
(1) 004012 017737 175432 001126 1$:  MOV    $GDDAT,@DAC1      ;LOAD DAC1  
(1) 004020 023737 001124 001126 1$:  SUB    #1,$GDDAT      ;SUB A VALUE  
(2) 004026 001404 001124 001126 1$:  SUB    #1,@DAC1      ;FROM EXPECTED AND DAC1  
(1) 004030 104003 001124 175410 1$:  MOV    @DAC1,$BDDAT      ;READ THE REGISTER  
(1) 004032 013777 001124 175410 1$:  CMP    $GDDAT,$BDDAT      ;COMPARE  
(1) 004040 005737 001124 001124 1$:  BEQ    2$                ;;BR IF SAME  
(2) 004044 001354 001124 001124 1$:  ERROR  3                ;THE SUB INSTRUCTION FAILED ON DAC1  
375      2$:  MOV    $GDDAT,@DAC1      ;LOAD THE REGISTER AGAIN  
(4)      2$:  TST    $GDDAT      ;TEST FOR MORE DATA  
(4)      2$:  BNE    1$                ;BR IF MORE DATA
```

```

377
(3)
(3)
(2) 004046 000004
378 004050 005037 001124
379 004054 013777 001124 175370
380 004062 017737 175364 001126
381 004070 023737 001124 001126
382 004076 001401
383 004100 104004
384
385
(3)
(3)
(2) 004102 000004
386 004104 012737 007777 001124
387 004112 013777 001124 175332
388 004120 017737 175326 001126
389 004126 023737 001124 001126
390 004134 001401
391 004136 104004
392
393
(3)
(3)
(2) 004140 000004
(1) 004142 012737 000100 001160
394 004150 012737 004000 001124
395 004156 013777 001124 175266
396 004164 017737 175262 001126
397 004172 023737 001124 001126
398 004200 001401
399 004202 104004
400 004204 006237 001124
401 004210 001362
402
403
(4)
(4)
(3) 004212 000004
(2) 004214 012737 000100 001160
(1) 004222 012737 004000 001124
(1) 004230 013777 001124 175214
(1) 004236 017737 175210 001126
(1) 004244 023737 001124 001126
(2) 004252 001407
(1) 004254 017737 175172 001126
(1) 004262 104004
(1) 004264 013777 001124 175160
(1) 004272 006277 175154
(1) 004276 006237 001124
(1) 004302 001355

*****
*TEST 15 TEST THAT THE DAC #2 REGISTER CAN BE CLEARED
*****
TST15: SCOPE
CLR $GDDAT ;LOAD EXPECTED
MOV $GDDAT,@DAC2 ;LOAD RE
MOV @DAC2,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST16 ;;BR IF EQUAL
ERROR 4 ;ERROR, DAC #2 REGISTER NOT = 0

*****
*TEST 16 TEST THAT THE DAC #2 REGISTER CAN BE LOADED WITH #7777
*****
TST16: SCOPE
MOV #7777,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@DAC2 ;LOAD REG
MOV @DAC2,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST17 ;;BR IF EQUAL
ERROR 4 ;ERROR, DAC #2 REGISTER NOT = 7777

*****
*TEST 17 TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN
*****
TST17: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #BIT11,$GDDAT ;LOAD EXPECTED
1$: MOV $GDDAT,@DAC2 ;LOAD DAC2 REGISTER
MOV @DAC2,$BDDAT ;READ THE REGISTER
CMP $GDDAT,$BDDAT ;COMPARE THE DATA
BEQ 2$ ;;BR IF SAME
ERROR 4 ;ERROR, DAC #2 REGISTER FAILED TO HOLD A FLOATIN
2$: ASR $GDDAT ;CHANGE THE DATA
BNE 1$ ;BR AND TEST MORE DATA

*****
*TEST 20 TEST THAT DAC2 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
*****
TST20: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #BIT11,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@DAC2 ;LOAD DAC2
1$: MOV @DAC2,$BDDAT ;READ THE REGISTER
CMP $GDDAT,$BDDAT ;COMPARE THE GOOD TO DAC2
BEQ 2$ ;;BR IF THE SAME
MOV @DAC2,$BDDAT ;SAVE FOR TYPEOUT
ERROR 4 ;DAC2 FAILED TO HOLD A FLOATING 1 PATTERN
MOV $GDDAT,@DAC2 ;LOAD DAC2 AGAIN
2$: ASR @DAC2 ;CHANGE THE DATA
ASR $GDDAT ;CHANGE THE EXPECTED
BNE 1$ ;BR IF MORE DATA
  
```

```
405      ::*****  
(4)      :*TEST 21      TEST THE 'SUB' INSTRUCTION WORKS ON DAC2  
(4)      :*****  
(3) 004304 000004  
(2) 004306 012737 000010 001160 TST21: SCOPE  
(1) 004314 012737 007777 001124      MOV #10,$TIMES      ;;DO 10 ITERATIONS  
(1) 004322 013777 001124 175122      MOV #7777,$GDDAT      ;LOAD EXPECTED  
(1) 004330 162737 000001 001124 1$:  MOV $GDDAT,@DAC2      ;LOAD DAC2  
(1) 004336 162777 000001 175106      SUB #1,$GDDAT      ;SUB A VALUE  
(1) 004344 017737 175102 001126      SUB #1,@DAC2      ;FROM EXPECTED AND DAC2  
(1) 004352 023737 001124 001126      MOV @DAC2,$BDDAT      ;READ THE REGISTER  
(2) 004360 001404      BEQ 2$      ;;BR IF SAME  
(1) 004362 104004      ERROR 4      ;THE SUB INSTRUCTION FAILED ON DAC2  
(1) 004364 013777 001124 175060      MOV $GDDAT,@DAC2      ;LOAD THE REGISTER AGAIN  
(1) 004372 005737 001124 2$:  TST $GDDAT      ;TEST FOR MORE DATA  
(2) 004376 001354      BNE 1$      ;;BR IF MORE DATA  
406  
407      ::*****  
(3)      :*TEST 22      TEST THAT THE DAC #3 REGISTER CAN BE CLEARED  
(3)      :*****  
(2) 004400 000004  
408 004402 005037 001124 TST22: SCOPE  
409 004406 013777 001124 175040      CLR $GDDAT      ;LOAD EXPECTED  
410 004414 017737 175034 001126      MOV $GDDAT,@DAC3      ;LOAD REG  
411 004422 023737 001124 001126      MOV @DAC3,$BDDAT      ;READ REG  
412 004430 001401      CMP $GDDAT,$BDDAT      ;COMPARE  
413 004432 104005      BEQ TST23      ;;BR IF EQUAL  
414      ERROR 5      ;ERROR, DAC #3 REGISTER NOT = 0  
415  
416      ::*****  
(3)      :*TEST 23      TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777  
(3)      :*****  
(2) 004434 000004  
416 004436 012737 007777 001124 TST23: SCOPE  
417 004444 013777 001124 175002      MOV #7777,$GDDAT      ;LOAD EXPECTED  
418 004452 017737 174776 001126      MOV $GDDAT,@DAC3      ;LOAD REG  
419 004460 023737 001124 001126      MOV @DAC3,$BDDAT      ;READ REG  
420 004466 001401      CMP $GDDAT,$BDDAT      ;COMPARE  
421 004470 104005      BEQ TST24      ;;BR IF EQUAL  
422      ERROR 5      ;ERROR, DAC #3 REGISTER NOT = 7777  
423  
424      ::*****  
(3)      :*TEST 24      TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN  
(3)      :*****  
(2) 004472 000004  
424 004474 012737 000100 001160 TST24: SCOPE  
425 004502 012737 004000 001124      MOV #100,$TIMES      ;;DO 100 ITERATIONS  
426 004510 013777 001124 174736      MOV #BIT11,$GDDAT      ;LOAD EXPECTED  
427 004524 023737 001124 001126 1$:  MOV $GDDAT,@DAC3      ;LOAD DAC #3 REGISTER  
428 004532 001401      MOV @DAC3,$BDDAT      ;READ THE REGISTER  
429 004534 104005      CMP $GDDAT,$BDDAT      ;COMPARE THE DATA  
430 004536 006237 001124 2$:  BEQ 2$      ;;BR IF SAME  
431 004542 001362      ERROR 5      ;ERROR, DAC #3 REGISTER FAILED TO HOLD A FLOATIN  
432      ASR $GDDAT      ;CHANGE THE DATA  
433      BNE 1$      ;BR AND TEST MORE DATA
```

```
434
(4)
(4)
(3) 004544 000004
(2) 004546 012737 000100 001160
(1) 004554 012737 004000 001124
(1) 004562 013777 001124 174664
(1) 004570 017737 174660 001126
(1) 004576 023737 001124 001126
(2) 004604 001407
(1) 004606 017737 174642 001126
(1) 004614 104005
(1) 004616 013777 001124 174630
(1) 004624 006277 174624
(1) 004630 006237 001124
(1) 004634 001355

*****
:*TEST 25 TEST THAT DAC3 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
*****
TST25: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #BIT11,$GDDAT ;:LOAD EXPECTED
MOV $GDDAT,@DAC3 ;:LOAD DAC3
1$: MOV @DAC3,$BDDAT ;:READ THE REGISTER
CMP $GDDAT,$BDDAT ;:COMPARE THE GOOD TO DAC3
BEQ 2$ ;:BR IF THE SAME
MOV @DAC3,$BDDAT ;:SAVE FOR TYPEOUT
ERROR 5 ;:DAC3 FAILED TO HOLD A FLOATING 1 PATTERN
MOV $GDDAT,@DAC3 ;:LOAD DAC3 AGAIN
2$: ASR @DAC3 ;:CHANGE THE DATA
ASR $GDDAT ;:CHANGE THE EXPECTED
BNE 1$ ;:BR IF MORE DATA

*****
:*TEST 26 TEST THE 'SUB' INSTRUCTION WORKS ON DAC3
*****
TST26: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #7777,$GDDAT ;:LOAD EXPECTED
MOV $GDDAT,@DAC3 ;:LOAD DAC3
1$: SUB #1,$GDDAT ;:SUB A VALUE
SUB #1,@DAC3 ;:FROM EXPECTED AND DAC3
MOV @DAC3,$BDDAT ;:READ THE REGISTER
CMP $GDDAT,$BDDAT ;:COMPARE
BEQ 2$ ;:BR IF SAME
ERROR 5 ;:THE SUB INSTRUCTION FAILED ON DAC3
MOV $GDDAT,@DAC3 ;:LOAD THE REGISTER AGAIN
(1) 004716 013777 001124 174530
(1) 004724 005737 001124
(2) 004730 001354
2$: TST $GDDAT ;:TEST FOR MORE DATA
BNE 1$ ;:BR IF MORE DATA
```



```
437 :*****
(3) :*TEST 27 TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA
(3) :*****
(2) 004732 000004 TST27: SCOPE
438 004734 012777 001111 174504 MOV #1111,@DAC0 ;LOAD DAC #0
439 004742 012777 002222 174500 MOV #2222,@DAC1 ;LOAD DAC #1
440 004750 012777 004444 174474 MOV #4444,@DAC2 ;LOAD DAC #2
441 004756 012777 007777 174470 MOV #7777,@DAC3 ;LOAD DAC #3
442 004764 012737 001111 001124 MOV #1111,$GDDAT ;LOAD EXPECTED
443 004772 017737 174450 001126 MOV @DAC0,$BDDAT ;READ REG
444 005000 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
445 005006 001401 BEQ 1$ ;;BR IF EQUAL
446 005010 104002 ERROR 2 ;ERROR, SELECTED DAC #0 IN ERROR
447
448 005012 012737 002222 001124 1$: MOV #2222,$GDDAT ;LOAD EXPECTED
449 005020 017737 174424 001126 MOV @DAC1,$BDDAT ;READ REG
450 005026 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
451 005034 001401 BEQ 2$ ;;BR IF EQUAL
452 005036 104003 ERROR 3 ;ERROR, SELECTED DAC #1 IN ERROR
453
454 005040 012737 004444 001124 2$: MOV #4444,$GDDAT ;LOAD EXPECTED
455 005046 017737 174400 001126 MOV @DAC2,$BDDAT ;READ REG
456 005054 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
457 005062 001401 BEQ 3$ ;;BR IF SAME
458 005064 104004 ERROR 4 ;ERROR, SELECTED DAC #2 IN ERROR
459
460 005066 012737 007777 001124 3$: MOV #7777,$GDDAT ;LOAD EXPECTED
461 005074 017737 174354 001126 MOV @DAC3,$BDDAT ;READ REG
462 005102 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
463 005110 001401 BEQ TST30 ;;BR IF SAME
464 005112 104005 ERROR 5 ;ERROR, SELECTED DAC #3 IN ERROR
```

```

466
(3)
(3)
(2) 005114 000004
467 005116 005737 001176
468 005122 001104
469 005124 012777 177777 174314
470 005132 005037 001124
471 005136 000005
472 005140 052777 000100 173776
473 005146 017737 174274 001126
474 005154 023737 001124 001126
475 005162 001401
476 005164 104002
477
478
(3)
(3)
(2) 005166 000004
479 005170 012777 177777 174252
480 005176 005037 001124
481 005202 000005
482 005204 052777 000100 173732
483 005212 017737 174232 001126
484 005220 023737 001124 001126
485 005226 001401
486 005230 104003
487
488
489
(3)
(3)
(2) 005232 000004
490 005234 012777 177777 174210
491 005242 005037 001124
492 005246 000005
493 005250 052777 000100 173666
494 005256 017737 174170 001126
495 005264 001401
496 005266 104004
497
498
(3)
(3)
(2) 005270 000004
499 005272 012777 177777 174154
500 005300 005037 001124
501 005304 012737 000001 010410
502 005312 000005
503 005314 052777 000100 173622
504 005322 017737 174126 001126
505 005330 001401
506 005332 104005
507

```

```

:*****
:*TEST 30      TEST THAT RESET CLEARS DAC #0 REGISTER
:*****
TST30:  SCOPE
        TST      $PASS          ;TEST PASS COUNTER
        BNE      REMAIN        ;BYPASS RESET TEST AFTER 1ST PASS
        MOV      #-1,@DAC0
        CLR      $GDDAT        ;LOAD EXPECTED
        RESET
        BIS      #BIT6,@$TKS    ;RE-ENABLE TKS INTR.
        MOV      @DAC0,$BDDAT   ;READ REG
        CMP      $GDDAT,$BDDAT  ;COMPARE
        BEQ      TST31         ;;BR IF EQUAL
        ERROR    2             ;ERROR, RESET FAILED TO CLEAR DAC #0

:*****
:*TEST 31      TEST THAT RESET CLEARS DAC #1 REGISTER
:*****
TST31:  SCOPE
        MOV      #-1,@DAC1
        CLR      $GDDAT        ;LOAD EXPECTED
        RESET
        BIS      #BIT6,@$TKS    ;RE-ENABLE TKS INTR.
        MOV      @DAC1,$BDDAT   ;READ REG
        CMP      $GDDAT,$BDDAT  ;COMPARE
        BEQ      TST32         ;;BR IF EQUAL
        ERROR    3             ;ERROR, RESET FAILED TO CLEAR DAC #1

:*****
:*TEST 32      TEST THAT RESET CLEARS DAC #2 REGISTER
:*****
TST32:  SCOPE
        MOV      #-1,@DAC2
        CLR      $GDDAT        ;LOAD THE REGISTER
        RESET                  ;CLEAR EXPECTED
        BIS      #BIT6,@$TKS    ;RE-ENABLE TKS INTR.
        MOV      @DAC2,$BDDAT   ;READ THE REGISTER
        BEQ      TST33         ;;BR IF CLEARED
        ERROR    4             ;ERROR, RESET FAILED TO CLEAR DAC #2

:*****
:*TEST 33      TEST THAT RESET CLEARS DAC #3 REGISTER
:*****
TST33:  SCOPE
        MOV      #-1,@DAC3
        CLR      $GDDAT        ;LOAD THE REGISTER
        MOV      #1,TEMP        ;CLEAR THE EXPECTED
        RESET
        BIS      #BIT6,@$TKS    ;RE-ENABLE TKS ENABLE
        MOV      @DAC3,$BDDAT   ;READ THE REGISTER
        BEQ      TST34         ;;BR IF CLEARED
        ERROR    5             ;ERROR, RESET FAILED TO CLEAR DAC #3

```

509 005334

510

(3)

(3)

(2) 005334 000004

(1) 005336 012737 000001 001160

511 005344 005237 001202

512 005350 123737 001202 001444

513 005356 001426

514 005360 005237 001200

515 005364 063737 001442 001446

516 005372 063737 001442 001450

517 005400 063737 001442 001452

518 005406 063737 001442 001454

519 005414 006337 010406

520 005420 005037 001102

521 005424 005046

522 005426 012746 003034

523 005432 000002

524

557

(3)

(3)

(2) 005434 000004

(1) 005436 012737 000001 001160

558 005444 005737 010422

559 005450 001002

560 005452 000137 006434

561

REMAIN:

::\*\*\*\*\*  
:\*TEST 34 DETERMINE IF MORE MNCA'S REMAIN TO BE TESTED  
:\*\*\*\*\*

TST34: SCOPE

MOV #1,\$TIMES ;:DO 1 ITERATION  
INC \$UNIT ;:UPDATE UNIT #  
CMPB \$UNIT,EVER ;:TEST IF MORE  
BEQ TST35 ;:BR IF NOT  
INC \$DEVCT ;:APT UNIT #  
ADD VADDR,DAC0 ;:UPDATE BUS ADDRESS  
ADD VADDR,DAC1  
ADD VADDR,DAC2  
ADD VADDR,DAC3  
ASL MASKNM ;:CHANGE THE ERROR FLAG BIT  
CLR \$STNM  
CLR -(SP)  
MOV #TST1,-(SP)  
RTI ;:LOWER PRIORITY AND RESTART TEST 1

::\*\*\*\*\*  
:\*TEST 35 DETERMINE IF RUNNING ON THE HARDWARE TESTER (IF NOT REPORT END OF PA  
:\*\*\*\*\*

TST35: SCOPE

MOV #1,\$TIMES ;:DO 1 ITERATION  
TST WFTST ;:TEST IF ON TESTER  
BNE TST36 ;:BR TO TEST  
JMP \$EOP

```

563
(3)
(3)
(2) 005456 000004
(1) 005460 012737 000001 001160
564 005466 012737 000010 010412
565 005474 012737 004000 010414
566 005502 012737 100000 010416
567 005510 013777 010412 173736
568 005516 017737 173736 001126
569 005524 042737 170377 001126
570 005532 013737 010414 001124
571 005540 023737 001124 001126
572 005546 001401
573 005550 104013
574 005552 017737 173702 001126
575 005560 042737 007777 001126
576 005566 013737 010416 001124
577 005574 005137 001124
578 005600 042737 007777 001124
579 005606 023737 001124 001126
580 005614 001401
581 005616 104013
582 005620 000257
583 005622 006037 010416
584 005626 006237 010414
585 005632 006237 010412
586 005636 001324
587
588
(3)
(3)
(2) 005640 000004
(1) 005642 012737 000001 001160
589 005650 013737 010424 001124
590 005656 004537 007700
591 005662 000012
592 005664 013737 010426 010420
593 005672 004737 010032
594 005676 000401
595 005700 104011
596
597
(3)
(3)
(2) 005702 000004
(1) 005704 012737 000001 001160
598 005712 013737 010430 001124
599 005720 004537 007700
600 005724 000011
601 005726 013737 010426 010420
602 005734 004737 010032
603 005740 000401
604 005742 104012

*****
*TEST 36 TEST THAT DAC #3 OUTPUT BITS (0-3) FUNCTION
*****
TST36: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #BIT3,$TEMP ;LOAD DAC PATTERN
MOV #BIT11,$TEMP1 ;LOAD EXPECTED PATTERN
MOV #BIT15,$TEMP2 ;LOAD OTHER (INVERTED) EXPECTED PATTERN
1$: MOV $TEMP,@DAC3 ;LOAD DAC REGISTER
MOV @DRIN,$BDDAT ;READ THE REGISTER
BIC #170377,$BDDAT ;MASK OFF OTHER BITS
MOV $TEMP1,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ;;BR IF THE SAME
ERROR 13 ;DAC #3 DIGITAL OUTPUT BITS IN ERROR
2$: MOV @DRIN,$BDDAT ;GET DATA AGAIN
BIC #7777,$BDDAT ;MASK OFF OTHER BITS
MOV $TEMP2,$GDDAT ;LOAD OTHER EXPECTED VALUE
COM $GDDAT ;INVERT DATA
BIC #7777,$GDDAT ;AND MASK OFF OTHER BITS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 3$ ;;BR IF SAME
ERROR 13 ;DAC 3 DIGITAL OUTPUT BITS IN ERROR
3$: CCC ;ADJUST EXPECTED
ROR $TEMP2 ;
ASR $TEMP1 ;
ASR $TEMP ;ADJUST LOADED PATTERN
BNE 1$

*****
*TEST 37 VERIFY THE MNCDA +15 SUPPLY
*****
TST37: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV V5744,$GDDAT ;LOAD EXPECTED
JSR R5,CONVRT ;SAMPLE THE CHANNEL
12
MOV V144,SPREAD ;LOAD TOLERANCE
JSR PC,COMPAR ;TEST IT
BR TST40 ;;BR
ERROR 11 ;+15 VOLT SUPPLY IS WRONG

*****
*TEST 40 VERIFY THE MNCDA -15 SUPPLY
*****
TST40: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV V2034,$GDDAT ;LOAD EXPECTED
JSR R5,CONVRT ;SAMPLE THE CHANNEL
11
MOV V144,SPREAD ;LOAD TOLERANCE
JSR PC,COMPAR ;TEST IT
BR TST41 ;;BR
ERROR 12 ;-15 VOLT SUPPLY IS WRONG

```

```
606
(4)
(4)
(3) 005744 000004
(1) 005746 004537 006642
(1) 005752 001446
(1) 005754 012701
(1) 005756 012266
(1) 005760 000013
(5)
(4)
(4)
(3) 005762 000004
(1) 005764 004537 007002
(1) 005770 001446
(1) 005772 012305
(1) 005774 000013
(5)
(4)
(4)
(3) 005776 000004
(2) 006000 012737 000001 001160
(1) 006006 004537 007126
(1) 006012 001446
(1) 006014 000013
(5)
(4)
(4)
(3) 005016 000004
(1) 006020 004537 007234
(1) 006024 001446
(1) 006026 012701
(1) 006030 000013
(5)
(4)
(4)
(3) 006032 000004
(1) 006034 004537 007352
(1) 006040 001446
(1) 006042 012701
(1) 006044 000013
(5)
(4)
(4)
(3) 006046 000004
(1) 006050 004537 007470
(1) 006054 001446
(1) 006056 012701
(1) 006060 000013

*****
*TEST 41 DACO OFFSET ADJUSTMENT
*****
TST41: SCOPE
        JSR R5,OFFDAC ;LOAD AND EXECUTE DAC OFFSET ADJ.
        DACO ;DAC ADDRESS
        SELDO ;TYPEOUT ADDRESS
        ADJR19 ;RES. TO ADJUST
        13 ;RESULT CHANNEL #

*****
*TEST 42 DACO GAIN ADJUSTMENT
*****
TST42: SCOPE
        JSR R5,GAIDAC ;LOAD AND EXECUTE DAC GAIN ADJ.
        DACO ;DAC ADDRESS
        ADJR18 ;RES. TO ADJUST
        13 ;CHANNEL # FOR RESULTS

*****
*TEST 43 DACO CALIBRATION
*****
TST43: SCOPE
        MOV #1,$TIMES ;DO 1 ITERATION
        JSR R5,CALDAC ;LOAD AND EXECUTE CALIBRATION
        DACO ;DAC ADDRESS
        13 ;CHANNEL # FOR RESULTS

*****
*TEST 44 DACO 2.5 VOLT SWITCH CHECK
*****
TST44: SCOPE
        JSR R5,TWOVLT ;LOAD AND EXECUTE 2.5 VOLT TEST
        DACO ;DAC ADDRESS
        SELDO ;TYPEOUT POINTER
        13 ;CHANNEL # FOR RESULTS

*****
*TEST 45 DACO 10.0 VOLT SWITCH CHECK
*****
TST45: SCOPE
        JSR R5,TENVLT ;LOAD AND EXECUTE 10.0 VOLT TEST
        DACO ;DAC ADDRESS
        SELDO ;TYPEOUT POINTER
        13 ;CHANNEL # FOR RESULTS

*****
*TEST 46 DACO UNIPOLAR SWITCH CHECK
*****
TST46: SCOPE
        JSR R5,FIVUNI ;LOAD AND EXECUTE UNIPOLAR TEST
        DACO ;DAC ADDRESS
        SELDO ;TYPEOUT POINTER
        13 ;CHANNEL # FOR RESULTS
```

```
608
(4)
(4)
(3) 006062 000004
(1) 006064 004537 006642
(1) 006070 001450
(1) 006072 012710
(1) 006074 012324
(1) 006076 000014
(5)
(4)
(4)
(3) 006100 000004
(1) 006102 004537 007002
(1) 006106 001450
(1) 006110 012343
(1) 006112 000014
(5)
(4)
(4)
(3) 006114 000004
(2) 006116 012737 000001 001160
(1) 006124 004537 007126
(1) 006130 001450
(1) 006132 000014
(5)
(4)
(4)
(3) 006134 000004
(1) 006136 004537 007234
(1) 006142 001450
(1) 006144 012710
(1) 006146 000014
(5)
(4)
(4)
(3) 006150 000004
(1) 006152 004537 007352
(1) 006156 001450
(1) 006160 012710
(1) 006162 000014
(5)
(4)
(4)
(3) 006164 000004
(1) 006166 004537 007470
(1) 006172 001450
(1) 006174 012710
(1) 006176 000014

*****
*TEST 47 DAC1 OFFSET ADJUSTMENT
*****
TST47: SCOPE
        JSR R5,OFFDAC ;LOAD AND EXECUTE DAC OFFSET ADJ.
        DAC1 ;DAC ADDRESS
        SELD1 ;TYPEOUT ADDRESS
        ADJR21 ;RES. TO ADJUST
        14 ;RESULT CHANNEL #

*****
*TEST 50 DAC1 GAIN ADJUSTMENT
*****
TST50: SCOPE
        JSR R5,GAIDAC ;LOAD AND EXECUTE DAC GAIN ADJ.
        DAC1 ;DAC ADDRESS
        ADJR20 ;RES. TO ADJUST
        14 ;CHANNEL # FOR RESULTS

*****
*TEST 51 DAC1 CALIBRATION
*****
TST51: SCOPE
        MOV #1,$TIMES ;;DO 1 ITERATION
        JSR R5,CALDAC ;LOAD AND EXECUTE CALIBRATION
        DAC1 ;DAC ADDRESS
        14 ;CHANNEL # FOR RESULTS

*****
*TEST 52 DAC1 2.5 VOLT SWITCH CHECK
*****
TST52: SCOPE
        JSR R5,TWOVLT ;LOAD AND EXECUTE 2.5 VOLT TEST
        DAC1 ;DAC ADDRESS
        SELD1 ;TYPEOUT POINTER
        14 ;CHANNEL # FOR RESULTS

*****
*TEST 53 DAC1 10.0 VOLT SWITCH CHECK
*****
TST53: SCOPE
        JSR R5,TENVLT ;LOAD AND EXECUTE 10.0 VOLT TEST
        DAC1 ;DAC ADDRESS
        SELD1 ;TYPEOUT POINTER
        14 ;CHANNEL # FOR RESULTS

*****
*TEST 54 DAC1 UNIPOLAR SWITCH CHECK
*****
TST54: SCOPE
        JSR R5,FIVUNI ;LOAD AND EXECUTE UNIPOLAR TEST
        DAC1 ;DAC ADDRESS
        SELD1 ;TYPEOUT POINTER
        14 ;CHANNEL # FOR RESULTS
```

```
610
(4)
(4)
(3) 006200 000004
(1) 006202 004537 006642
(1) 006206 001452
(1) 006210 012717
(1) 006212 012362
(1) 006214 000015
(5)
(4)
(4)
(3) 006216 000004
(1) 006220 004537 007002
(1) 006224 001452
(1) 006226 012401
(1) 006230 000015
(5)
(4)
(4)
(3) 006232 000004
(2) 006234 012737 000001 001160
(1) 006242 004537 007126
(1) 006246 001452
(1) 006250 000015
(5)
(4)
(4)
(3) 006252 000004
(1) 006254 004537 007234
(1) 006260 001452
(1) 006262 012717
(1) 006264 000015
(5)
(4)
(4)
(3) 006266 000004
(1) 006270 004537 007352
(1) 006274 001452
(1) 006276 012717
(1) 006300 000015
(5)
(4)
(4)
(3) 006302 000004
(1) 006304 004537 007470
(1) 006310 001452
(1) 006312 012717
(1) 006314 000015

*****
*TEST 55 DAC2 OFFSET ADJUSTMENT
*****
TST55: SCOPE
        JSR R5,OFFDAC ;LOAD AND EXECUTE DAC OFFSET ADJ.
        DAC2 ;DAC ADDRESS
        SELD2 ;TYPEOUT ADDRESS
        ADJR42 ;RES. TO ADJUST
        15 ;RESULT CHANNEL #

*****
*TEST 56 DAC2 GAIN ADJUSTMENT
*****
TST56: SCOPE
        JSR R5,GAIDAC ;LOAD AND EXECUTE DAC GAIN ADJ.
        DAC2 ;DAC ADDRESS
        ADJR41 ;RES. TO ADJUST
        15 ;CHANNEL # FOR RESULTS

*****
*TEST 57 DAC2 CALIBRATION
*****
TST57: SCOPE
        MOV #1,$TIMES ;DO 1 ITERATION
        JSR R5,CALDAC ;LOAD AND EXECUTE CALIBRATION
        DAC2 ;DAC ADDRESS
        15 ;CHANNEL # FOR RESULTS

*****
*TEST 60 DAC2 2.5 VOLT SWITCH CHECK
*****
TST60: SCOPE
        JSR R5,TWOVLT ;LOAD AND EXECUTE 2.5 VOLT TEST
        DAC2 ;DAC ADDRESS
        SELD2 ;TYPEOUT POINTER
        15 ;CHANNEL # FOR RESULTS

*****
*TEST 61 DAC2 10.0 VOLT SWITCH CHECK
*****
TST61: SCOPE
        JSR R5,TENVLT ;LOAD AND EXECUTE 10.0 VOLT TEST
        DAC2 ;DAC ADDRESS
        SELD2 ;TYPEOUT POINTER
        15 ;CHANNEL # FOR RESULTS

*****
*TEST 62 DAC2 UNIPOLAR SWITCH CHECK
*****
TST62: SCOPE
        JSR R5,FIVUNI ;LOAD AND EXECUTE UNIPOLAR TEST
        DAC2 ;DAC ADDRESS
        SELD2 ;TYPEOUT POINTER
        15 ;CHANNEL # FOR RESULTS
```



```
612
(4)
(4)
(3) 006316 000004
(1) 006320 004537 006642
(1) 006324 001454
(1) 006326 012726
(1) 006330 012420
(1) 006332 000016
(5)
(4)
(4)
(3) 006334 000004
(1) 006336 004537 007002
(1) 006342 001454
(1) 006344 012437
(1) 006346 000016
(5)
(4)
(4)
(3) 006350 000004
(2) 006352 012737 000001 001160
(1) 006360 004537 007126
(1) 006364 001454
(1) 006366 000016
(5)
(4)
(4)
(3) 006370 000004
(1) 006372 004537 007234
(1) 006376 001454
(1) 006400 012726
(1) 006402 000016
(5)
(4)
(4)
(3) 006404 000004
(1) 006406 004537 007352
(1) 006412 001454
(1) 006414 012726
(1) 006416 000016
(5)
(4)
(4)
(3) 006420 000004
(1) 006422 004537 007470
(1) 006426 001454
(1) 006430 012726
(1) 006432 000016

*****
*TEST 63 DAC3 OFFSET ADJUSTMENT
*****
TST63: SCOPE
        JSR R5,OFFDAC ;LOAD AND EXECUTE DAC OFFSET ADJ.
        DAC3 ;DAC ADDRESS
        SELD3 ;TYPEOUT ADDRESS
        ADJR44 ;RES. TO ADJUST
        16 ;RESULT CHANNEL #

*****
*TEST 64 DAC3 GAIN ADJUSTMENT
*****
TST64: SCOPE
        JSR R5,GAIDAC ;LOAD AND EXECUTE DAC GAIN ADJ.
        DAC3 ;DAC ADDRESS
        ADJR43 ;RES. TO ADJUST
        16 ;CHANNEL # FOR RESULTS

*****
*TEST 65 DAC3 CALIBRATION
*****
TST65: SCOPE
        MOV #1,$TIMES ;:DO 1 ITERATION
        JSR R5,CALDAC ;LOAD AND EXECUTE CALIBRATION
        DAC3 ;DAC ADDRESS
        16 ;CHANNEL # FOR RESULTS

*****
*TEST 66 DAC3 2.5 VOLT SWITCH CHECK
*****
TST66: SCOPE
        JSR R5,TWOVLT ;LOAD AND EXECUTE 2.5 VOLT TEST
        DAC3 ;DAC ADDRESS
        SELD3 ;TYPEOUT POINTER
        16 ;CHANNEL # FOR RESULTS

*****
*TEST 67 DAC3 10.0 VOLT SWITCH CHECK
*****
TST67: SCOPE
        JSR R5,TENVLT ;LOAD AND EXECUTE 10.0 VOLT TEST
        DAC3 ;DAC ADDRESS
        SELD3 ;TYPEOUT POINTER
        16 ;CHANNEL # FOR RESULTS

*****
*TEST 70 DAC3 UNIPOLAR SWITCH CHECK
*****
TST70: SCOPE
        JSR R5,FIVUNI ;LOAD AND EXECUTE UNIPOLAR TEST
        DAC3 ;DAC ADDRESS
        SELD3 ;TYPEOUT POINTER
        16 ;CHANNEL # FOR RESULTS
```

```

614          .SBTTL  END OF PASS ROUTINE
(1)
(2)          ::*****
(1)          :*INCREMENT THE PASS NUMBER ($PASS)
(1)          :*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
(1)          :*IF THERES A MONITOR GO TO IT
(1)          :*IF THERE ISN'T JUMP TO INIT?
(1)
(1) 006434          $EOP:
(1) 006434          000004          SCOPE
(1) 006436          005037          001102          CLR          $TSTNM          ;;ZERO THE TEST NUMBER
(1) 006442          005037          001160          CLR          $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
(1) 006446          005237          001176          INC          $PASS          ;;INCREMENT THE PASS NUMBER
(1) 006452          042737          100000          001176          BIC          #100000,$PASS          ;;DON'T ALLOW A NEG. NUMBER
(1) 006460          005327          DEC          (PC)+          ;;LOOP?
(1) 006462          000001          $EOPCT: .WORD          1
(1) 006464          003022          BGT          $DOAGN          ;;YES
(1) 006466          012737          MOV          (PC)+,@(PC)+          ;;RESTORE COUNTER
(1) 006470          000001          $ENDCT: .WORD          1
(1) 006472          006462          $EOPCT
(1) 006474          104401          006541          TYPE          $SENDMG          ;;TYPE 'END PASS #'
(2) 006500          013746          001176          MOV          $PASS,-(SP)          ;;SAVE $PASS FOR TYPEOUT
(2) 006504          104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 006506          104401          006536          TYPE          $ENULL          ;;TYPE A NULL CHARACTER
(1) 006512          013700          000042          $GET42: MOV          @#42,R0          ;;GET MONITOR ADDRESS
(1) 006516          001405          BEQ          $DOAGN          ;;BRANCH IF NO MONITOR
(1) 006520          000005          RESET          ;;CLEAR THE WORLD
(1) 006522          004710          $ENDAD: JSR          PC,(R0)          ;;GO TO MONITOR
(1) 006524          000240          NOP          ;;SAVE ROOM
(1) 006526          000240          NOP          ;;FOR
(1) 006530          000240          NOP          ;;ACT11
(1) 006532          $DOAGN:
(1) 006532          000137          JMP          @(PC)+          ;;RETURN
(1) 006534          006556          $RTNAD: .WORD          INIT7
(1) 006536          377          377          000          $ENULL: .BYTE          -1,-1,0          ;;NULL CHARACTER STRING
(1) 006541          015          042412          042116          $SENDMG: .ASCIIZ          <15><12>/END PASS #/
(1) 006546          050040          051501          020123
(1) 006554          000043
615
616 006556          052777          000100          172360          INIT7: BIS          #BIT6,@$TKS          ;RE-ENABLE TKS INTR.
617 006564          005737          001112          TST          $ERTTL          ;TEST IF ANY ERRORS
618 006570          001416          BEQ          1$          ;BR IF NONE
619 006572          104401          012002          TYPE,          ERRTOT
620 006576          013746          001112          MOV          $ERTTL,-(SP)
621 006602          104405          TYPDS
622 006604          022737          000001          010406          CMP          #1,MASKNM          ;TEST IF ADDITIONAL UNITS
623 006612          001405          BEQ          1$          ;BR IF NONE
624 006614          104401          012031          TYPE,          MESGD          ;INFORM OPER. ABOUT OTHER UNIT ERRORS
625 006620          013746          010404          MOV          BADUNT,-(SP)          ;;SAVE BADUNT FOR TYPEOUT
(1) 006624          104406          TYPBN          ;;GO TYPE--BINARY ASCII
626 006626          104401          006536          1$: TYPE,          $ENULL          ;ENSURE THE OUTPUT MADE IT
627 006632          004737          010174          JSR          PC,CTRLCG          ;TEST FOR CONTROL CHAR.
628 006636          000137          002600          JMP          LOGIC          ;TEST IT AGAIN

```

```

630          .SBTTL  SUBROUTINE TO ADJUST THE DAC'S OFFSET POTS
631
632 006642 012537 007000 OFFDAC: MOV      (R5)+,10$          ;GET BUS ADDRESS
633 006646 017737 000126 007000 MOV      @10$,10$          ;
634 006654 013737 007000 010402 MOV      10$,DACBAD      ;LOAD BUS ADDRESS IF ERROR
635 006662 012537 006712 MOV      (R5)+,11$      ;GET POINTER TO ASCII MESSAGE
636 006666 012537 006730 MOV      (R5)+,12$      ;GET POINTER TO RES. MESSAGE
637 006672 012537 006750 MOV      (R5)+,13$      ;GET AND SAVE CHANNEL #
638 006676 005737 001176 TST      $PASS          ;TEST IF FIRST PASS
639 006702 001035 BNE      2$            ;EXIT IF NOT FIRST PASS
640 006704 104401 TYPE          ;TELL OPERATOR TO SELECT DAC N
641 006706 012456 SELDX
642 006710 104401 TYPE
643 006712 012701 11$: SELDO
644 006714 104401 TYPE
645 006716 013003 N51200 ;TELL OPERATOR ABOUT THE EXPECTED VOLTAGE
646 006720 012777 000000 000052 MOV      #0000,@10$      ;LOAD THE SELECTED DAC TO NULL
647 006726 104401 TYPE          ;TELL OPERATOR TO ADJUST RXX
648 006730 012266 12$: ADJR19
649 006732 004737 010020 1$: JSR      PC,CSPACE      ;WAIT UNTIL THE IS READY
650 006736 012737 000000 001124 MOV      #0000,$GDDAT    ;LOAD EXPECTED VALUE
651 006744 004537 007700 JSR      R5,CONVRT      ;SAMPLE THE CHANNEL
652 006750 000013 13$: 13
653 006752 012737 000004 010420 MOV      #4,SPREAD
654 006760 004737 010032 JSR      PC,COMPAR      ;TEST RESULTS
655 006764 000404 BR        2$            ;:BR IF WITHIN THE LIMIT
656 006766 104006 ERROR 6 ;SELECTED DAC OFFSET POT WAS NOT ADJUSTED INCORRECTLY
657 006770 104401 TYPE
658 006772 012735 TRYAGN
659 006774 000756 BR        1$            ;LOOP AGAIN
660 006776 000205 2$: RTS      R5          ;EXIT
661 007000 000000 10$: 0
662
663          .SBTTL  SUBROUTINE TO ADJUST THE GAIN ADJUSTMENT POTS
664
665 007002 012537 007124 GAIDAC: MOV      (R5)+,10$          ;GET BUS ADDRESS
666 007006 017737 000112 007124 MOV      @10$,10$          ;
667 007014 013737 007124 010402 MOV      10$,DACBAD      ;LOAD BUS ADDRESS IF ERROR
668 007022 012537 007054 MOV      (R5)+,11$      ;GET ASCII RES. ADDRESS
669 007026 012537 007074 MOV      (R5)+,12$      ;GET CHANNEL #
670 007032 005737 001176 TST      $PASS          ;TEST IF FIRST PASS
671 007036 001031 BNE      2$            ;BR IF NOT
672 007040 104401 TYPE          ;TELL OPERATOR ABOUT THE EXPECTED VOLTAGE
673 007042 013061 P51175
674 007044 012777 007777 000052 MOV      #7777,@10$      ;LOAD THE DAC
675 007052 104401 TYPE          ;TELL OPERATOR WHICH RXX TO ADJUST
676 007054 012305 11$: ADJR18
677 007056 004737 010020 1$: JSR      PC,CSPACE      ;WAIT FOR OPERATOR
678 007062 012737 007777 001124 MOV      #7777,$GDDAT    ;LOAD EXPECTED
679 007070 004537 007700 JSR      R5,CONVRT      ;CONVERT THE VALUE
680 007074 000013 12$: 13
681 007076 012737 000004 010420 MOV      #4,SPREAD      ;LOAD LIMIT
682 007104 004737 010032 JSR      PC,COMPAR      ;TEST RESULTS
683 007110 000404 BR        2$            ;:BR IF WITHIN LIMITS

```

```

684 007112 104007          ERROR 7          ;SELECTED DAC GAIN POT WAS NOT ADJUSTED PROPERLY
685 007114 104401          TYPE
686 007116 012735          TRYAGN
687 007120 000756          BR          1$
688 007122 000205          2$: RTS          R5          ;EXIT
689 007124 000000          10$: 0
690
691          .SBTTL SUBROUTINE TO TEST THE D/A CALIBRATION
692
693 007126 012537 007232 CALDAC: MOV          (R5)+,10$          ;GET BUS ADDRESS
694 007132 017737 000074 007232 MOV          @10$,10$          ;
695 007140 013737 007232 010402 MOV          10$,DACBAD          ;LOAD BUS ADDRESS IF ERROR
696 007146 012537 007172 MOV          (R5)+,11$          ;GET CHANNEL #
697
698 007152 012777 007400 000052 MOV          #7400,@10$          ;LOAD THE DAC
699 007160 012737 007400 001124 MOV          #7400,$GDDAT          ;LOAD THE EXPECTED VALUE
700
701 007166 004537 007700          1$: JSR          R5,CONVRT          ;SAMPLE THE CHANNEL
702 007172 000013          11$: 13
703
704 007174 012737 000005 010420 MOV          #5,SPREAD          ;LOAD TOLERANCE
705 007202 004737 010032 JSR          PC,COMPAR          ;TEST THE RESULTS
706 007206 000401 BR          2$          ;;BR
707 007210 104010 ERROR          10          ;NON-LINEARITY IN DAC DETECTED
708 007212 162777 000400 000012 2$: SUB          #400,@10$          ;ADJUST THE CONTENTS
709 007220 162737 000400 001124 SUB          #400,$GDDAT          ;ADJUST THE EXPECTED
710 007226 001357 BNE          1$          ;;BR IF NOT DONE
711 007230 000205 RTS          R5          ;EXIT
712 007232 000000          10$: 0
713
714          .SBTTL SUBROUTINE TO VERIFY THE 2.5 VOLT SWITCH POSITION
715 007234 012537 007350 TWOVLT: MOV          (R5)+,10$          ;SAVE DAC ADDRESS
716 007240 017737 000104 007350 MOV          @10$,10$          ;GET ACTUAL ADDRESS
717 007246 013737 007350 010402 MOV          10$,DACBAD          ;SAVE ADDRESS FOR TYPEOUT
718 007254 012537 007300 MOV          (R5)+,11$          ;GET ASCII MESSAGE
719 007260 012537 007326 MOV          (R5)+,12$          ;GET TESTER CHANNEL
720 007264 005737 001176 TST          $PASS          ;TEST IF FIRST PASS
721 007270 001026 BNE          2$          ;BR IF NOT
722 007272 104401 012515 TYPE, TWOTXT          ;TELL OPERATOR TO CHANGE SWITCH
723 007276 104401 TYPE
724 007300 012701          11$: SELDO          ;ON DAC N
725 007302 004737 010020 JSR          PC,CSPACE          ;WAIT FOR OPERATOR
726 007306 012777 007777 000034 1$: MOV          #7777,@10$          ;LOAD DAC OUTPUT
727 007314 012737 006000 001124 MOV          #6000,$GDDAT          ;LOAD EXPECTED
728 007322 004537 007700 JSR          R5,CONVRT          ;CONVERT DATA ON CH
729 007326 000013          12$: 13
730 007330 012737 000004 010420 MOV          #4,SPREAD          ;LOAD TOLERANCE
731 007336 004737 010032 JSR          PC,COMPAR          ;TEST IF WITHIN LIMITS
732 007342 000401 BR          2$
733 007344 104016 ERROR          16          ;FRONT PANEL SWITCH ERROR
734 007346 000205          2$: RTS          R5          ;EXIT
735 007350 000000          10$: 0
736          .SBTTL SUBROUTINE TO VERIFY THE 10.0 VOLT SWITCH POSITION
737 007352 012537 007466 TENVLT: MOV          (R5)+,10$          ;GET DAC ADDRESS

```

```

738 007356 017737 000104 007466      MOV      @10$,10$      ;GET READ ADDRESS
739 007364 013737 007466 010402      MOV      10$,DACBAD  ;SAVE FOR TYPEOUT
740 007372 012537 007416      MOV      (R5)+,11$   ;GET TEXT POINTER
741 007376 012537 007444      MOV      (R5)+,12$   ;GET TESTER CHANNEL
742 007402 005737 001176      TST      $PASS      ;TEST IF FIRST PASS
743 007406 001026      BNE      2$         ;BR IF NOT
744 007410 104401 012553      TYPE,    TENTXT
745 007414 104401
746 007416 012701      11$:    SELDO
747 007420 004737 010020      JSR      PC,CSPACE
748 007424 012777 004777 000034 1$:    MOV      #4777,@10$  ;LOAD DAC OUTPUT
749 007432 012737 006000 001124      MOV      #6000,$GDDAT ;LOAD EXPECTED
750 007440 004537 007700      JSR      R5,CONVRT   ;CONVERT DATA ON CH
751 007444 000013      12$:    13
752 007446 012737 000015 010420      MOV      #15,SPREAD  ;LOAD TOLERANCE
753 007454 004737 010032      JSR      PC,COMPAR   ;TEST IF WITHIN LIMITS
754 007460 000401      BR      2$         ;BR IF NO ERROR
755 007462 104016      ERROR   16        ;FRONT PANEL SWITCH ERROR
756 007464 000205      2$:    RTS      R5      ;EXIT
757 007466 000000      10$:    0
758      .SBTTL  SUBROUTINE TO VERIFY THE UNIPOLAR SWITCH POSITION
759 007470 012537 007626      FIVUNI: MOV      (R5)+,10$   ;GET DAC ADDRESS
760 007474 017737 000126 007626      MOV      @10$,10$   ;GET REAL ADDRESS
761 007502 013737 007626 010402      MOV      10$,DACBAD ;SAVE FOR TYPEOUT
762 007510 012537 007542      MOV      (R5)+,11$   ;GET TEXT POINTER
763 007514 013737 007542 007616      MOV      11$,13$    ;
764 007522 012537 007570      MOV      (R5)+,12$   ;GET TESTER CHANNEL
765 007526 005737 001176      TST      $PASS      ;TEST IF FIRST PASS
766 007532 001034      BNE      3$         ;BR IF NOT
767 007534 104401 012610      TYPE,    UNITXT
768 007540 104401
769 007542 012701      11$:    SELDO
770 007544 004737 010020      JSR      PC,CSPACE  ;WAIT FOR OPERATOR
771 007550 012777 001777 000050 1$:    MOV      #1777,@10$ ;LOAD DAC OUTPUT
772 007556 012737 006000 001124      MOV      #6000,$GDDAT ;LOAD EXPECTED
773 007564 004537 007700      JSR      R5,CONVRT   ;CONVERT DATA ON CH
774 007570 000013      12$:    13
775 007572 012737 000015 010420      MOV      #15,SPREAD  ;LOAD TOLERANCE
776 007600 004737 010032      JSR      PC,COMPAR   ;TEST IF WITHIN LIMITS
777 007604 000401      BR      2$         ;BR IF NO ERROR
778 007606 104016      ERROR   16        ;FRONT PANEL SWITCH ERROR
779 007610 104401 012645      2$:    TYPE,    BIPTXT ;TELL OPERATOR TO CHANGE SWITCH
780 007614 104401      TYPE,    ;ON SELECTED DAC
781 007616 012701      13$:    SELDO
782 007620 004737 010020      JSR      PC,CSPACE  ;WAIT FOR OPERATOR
783 007624 000205      3$:    RTS      R5      ;EXIT
784 007626 000000      10$:    0
785      .SBTTL  FIELD TEST MODULE DIGITAL OUTPUT LAMP LOOP
786 007630 012706 001100      LAMPS: MOV      #STACK,SP  ;PRIME THE STACK
787 007634 004737 002044      JSR      PC,LDTRAP  ;LOAD TRAP AND ADDRESSES
788 007640 012702 000010 1$:    MOV      #BIT3,R2  ;LOAD DATA PATTERN
789 007644 012700 000001 2$:    MOV      #1,R0     ;LOAD DELAY
790 007650 005001      CLR      R1
791 007652 010277 171576      MOV      R2,@DAC3   ;LOAD DATA OUTPUT

```

```

792 007656 005301          3$:  DEC      R1
793 007660 001376          BNE     3$           ;DELAY
794 007662 005300          DEC     R0
795 007664 001374          BNE     3$
796 007666 006202          ASR     R2           ;CHANGE THE DATA
797 007670 001365          BNE     2$
798 007672 004737 010174   JSR     PC,CTRLCG   ;TEST FOR CONTROL CHAR.
799 007676 000760          BR      1$
800
801          .SBTTL  SUBROUTINE TO CONVERT CHANNEL N ON THE TESTER A/D
802
803 007700 012537 010012   CONVRT: MOV     (R5)+,10$           ;GET THE CHANNEL #
804 007704 000337 010012   SWAB   10$
805 007710 042737 170377 010012   BIC     #170377,10$           ;MASK OUT OTHER BITS
806 007716 013777 010012 171536   MOV     10$,@ADCS           ;SELECT CHANNEL
807 007724 005037 010014   CLR     11$
808 007730 012737 000200 010016   MOV     #BIT7,12$           ;LOAD SHIFT COUNTER
809 007736 105277 171520   1$:    INCB   @ADCS           ;CONVERT CHANNEL
810 007742 105777 171514   2$:    TSTB   @ADCS           ;WAIT FOR DONF
811 007746 100375          BPL     2$
812 007750 067737 171510 010014   ADD     @ADBR,11$           ;UPDATE CONVERSION
813 007756 006237 010016   ASR     12$
814 007762 001365          BNE     1$           ;FINISHED ?
815 007764 000257          CCC
816 007766 006037 010014   ROR     11$
817 007772 006237 010014   ASR     11$
818 007776 006237 010014   ASR     11$           ;JUSTIFY DATA
819 010002 013737 010014 001126   MOV     11$,SBDAT           ;LOAD ACTUAL <ADJUSTED>
820 010010 000205          RTS     R5           ;EXIT
821 010012 000000          10$:    0
822 010014 000000          11$:    0
823 010016 000000          12$:    0
824
825          .SBTTL  SUBROUTINE TO LOOP UNTIL OPERATOR TYPES THE 'RETURN' KEY
826
827 010020 104401 011601   CSPACE: TYPE,  LDSPAC           ;TELL OPERATOR TO HIT RETURN KEY
828 010024 104412          RDLIN           ;WAIT FOR INPUT
829 010026 005726          TST     (SP)+           ;CLEAN STACK
830 010030 000207          RTS     PC           ;EXIT

```

```

832          .SBTTL  SUBROUTINE TO COMPARE TWO LOCATIONS BY THE SPREAD
833
834 010032 010046          COMPAR: MOV    R0,-(SP)          ;SAVE R0
835 010034 010146          MOV    R1,-(SP)          ;SAVE R1
836 010036 013700 001124  MOV    $GDDAT,R0        ;GET EXPECTED VALUE
837 010042 013701 001126  MOV    $BDDAT,R1        ;GET THE UNKNOWN
838 010046 160100          SUB    R1,R0          ;SUBTRACT
839 010050 100001          BPL    8$
840 010052 005400          NEG    R0
841 010054 020037 010420  8$:  CMP    R0,SPREAD        ;TEST IF DIFFERENCE IF > THAN SPREAD
842 010060 003405          BLE    10$
843 010062 012601          9$:  MOV    (SP)+,R1        ;RESTORE R1
844 010064 012600          MOV    (SP)+,R0        ;RESTORE R0
845 010066 062716 000002  ADD    #2,(SP)        ;MAKE AN ERROR EXIT
846 010072 000207          RTS    PC          ;EXIT
847 010074 012601          10$: MOV    (SP)+,R1
848 010076 012600          MOV    (SP)+,R0
849 010100 000207          RTS    PC          ;EXIT FOR GOOD LIMIT TEST
850
851          .SBTTL  FULL SCALE RAMP ON EACH RAMP
852
853 010102 012706 001100  FULRMP: MOV    #STACK,SP        ;LOAD POINTER
854 010106 004737 002044  JSR    PC,LDTRAP        ;LOAD BUS ADDRESS
855 010112 013700 001446  1$:  MOV    DAC0,R0        ;GET BUS ADDRESS
856 010116 004737 010160  JSR    PC,10$          ;LOAD THE RAMP ON DAC #1
857 010122 013700 001450  MOV    DAC1,R0        ;GET BUS ADDRESS
858 010126 004737 010160  JSR    PC,10$          ;LOAD THE RAMP ON DAC #1
859 010132 013700 001452  MOV    DAC2,R0        ;GET BUS ADDRESS
860 010136 004737 010160  JSR    PC,10$          ;LOAD THE RAMP ON DAC #2
861 010142 013700 001454  MOV    DAC3,R0        ;GET THE BUS ADDRESS
862 010146 004737 010160  JSR    PC,10$          ;LOAD THE RAMP ON DAC #3
863 010152 004737 010174  JSR    PC,CTRLCG        ;TEST FOR CONTROL CHAR.
864 010156 000755          BR    1$            ;BR BACK
865 010160 005010          10$: CLR    (R0)        ;CLEAR DAC
866 010162 062710 000010  11$: ADD    #10,(R0)        ;UPDATE THE DATA
867 010166 005710          TST    (R0)        ;TEST IF DONE
868 010170 001374          BNE    11$        ;BR IF NOT
869 010172 000207          RTS    PC          ;EXIT
870
871          .SBTTL  CONTROL CHARACTER DETECTOR
872
873 010174 105777 170744  CTRLCG: TSTB  @STKS        ;TEST FOR INPUT FLAG
874 010200 100022          BPL    2$            ;BR IF NOT
875 010202 017737 170740 010250  MOV    @STKB,CTRCHA    ;SAVE CHARACTER
876 010210 042737 177640 010250  BIC    #177640,CTRCHA  ;MASK OFF BITS
877 010216 022737 000003 010250  CMP    #3,CTRCHA        ;TEST IF CTRL C
878 010224 001003          BNE    1$            ;BR IF NOT
879 010226 005726          TST    (SP)+        ;CLEAN THE STACK
880 010230 000137 0023'4  JMP    MTEST1        ;AND RETYPE THE 'DOT'
881 010234 022737 000007 010250  1$:  CMP    #7,CTRCHA        ;TEST IF CTRL G
882 010242 001001          BNE    2$            ;BR IF NOT
883 010244 104407          GTSWR          ;GET SWR FROM OPERATOR
884 010246 000207          RTS    PC          ;EXIT
885 010250 000000          CTRLCHA: 0          ;CHAR. THE OPER. TYPED DUNRING RUNNING

```

```

891
892          .SBTTL  STATIC DAC CALIBRATION
893
894 010252 012706 001100  STATIC: MOV    #STACK,SP      ;LOAD STACK POINTER
895 010256 004737 002044      JSR    PC,LDTRAP      ;LOAD BUS ADDRESSES
896 010262 004737 010174 1$:  JSR    PC,CTRLCG      ;TEST FOR CTRL G OR C
897 010266 017700 170646      MOV    @SWR,R0        ;READ SWITCHES
898 010272 010077 171150      MOV    R0,@DAC0      ;LOAD DAC #0
899 010276 010077 171146      MOV    R0,@DAC1      ;LOAD DAC #1
900 010302 010077 171144      MOV    R0,@DAC2      ;LOAD DAC #2
901 010306 010077 171142      MOV    R0,@DAC3      ;LOAD DAC #3
902 010312 000763      BR    1$
903
904          .SBTTL  DYNAMIC DAC CALIBRATION
905
906 010314 012706 001100  DYNCAL: MOV    #STACK,SP      ;LOAD STACK POINTER
907 010320 004737 002044      JSR    PC,LDTRAP      ;LOAD BUS ADDRESSES
908 010324 004737 010174 1$:  JSR    PC,CTRLCG      ;TEST FOR CTRL G OR C
909 010330 017700 170604      MOV    @SWR,R0        ;READ SWR
910 010334 004737 010350      JSR    PC,10$         ;LOAD THE SWR VALUE TO ALL DACS
911 010340 005000      CLR    R0            ;CLEAR R0
912 010342 004737 010350      JSR    PC,10$         ;LOAD ALL DAC'S WITH 0
913 010346 000766      BR    1$
914
915 010350 010077 171072 10$:  MOV    R0,@DAC0      ;LOAD DAC #0
916 010354 010077 171070      MOV    R0,@DAC1      ;LOAD DAC #1
917 010360 010077 171066      MOV    R0,@DAC2      ;LOAD DAC #2
918 010364 010077 171064      MOV    R0,@DAC3      ;LOAD DAC #3
919 010370 012700 000020      MOV    #20,R0        ;LOAD DELAY COUNTER
920 010374 005300 11$:  DEC    R0            ;DELAY
921 010376 100376      BPL   11$            ;WAIT
922 010400 000207      RTS    PC            ;EXIT
923
924 010402 171060  DACBAD: ABASE
925 010404 000000  BADUNT: 0
926 010406 000001  MASKNM: BIT0
927 010410 000000  TEMP: 0
928 010412 000000  $TEMP: 0
929 010414 000000  $TEMP1: 0
930 010416 000000  $TEMP2: 0
931 010420 000000  SPREAD: 0
932 010422 000000  WFTST: 0
933 010424 005744  V5744: 5744
934 010426 000144  V144: 144
935 010430 002034  V2034: 2034
936
940

```



```

942
943
944
945 010432 047115 040503 020101 EM1: .ASCIZ \MNCAA (D/A) DOES NOT EXIST <BUS ERROR> CHECK ADDRESS SWITCHES\
    010440 042050 040457 004451
    010446 047504 051505 047040
    010454 052117 042440 044530
    010462 052123 036040 052502
    010470 020123 051105 047522
    010476 037122 020040 044103
    010504 041505 020113 042101
    010512 051104 051505 020123
    010520 053523 052111 044103
    010526 051505 000
946 010531 115 041516 040501 EM2: .ASCIZ \MNCAA (D/A) DAC0 REGISTER IN ERROR\
    010536 024040 027504 024501
    010544 042011 041501 020060
    010552 042522 044507 052123
    010560 051105 044440 020116
    010566 051105 047522 000122
947 010574 047115 040503 020101 EM3: .ASCIZ \MNCAA (D/A) DAC1 REGISTER IN ERROR\
    010602 042050 040457 004451
    010610 040504 030503 051040
    010616 043505 051511 042524
    010624 020122 047111 042440
    010632 051122 051117 000
948 010637 115 041516 040501 EM4: .ASCIZ \MNCAA (D/A) DAC2 REGISTER IN ERROR\
    010644 024040 027504 024501
    010652 042011 041501 020062
    010660 042522 044507 052123
    010666 051105 044440 020116
    010674 051105 047522 000122
949 010702 047115 040503 020101 EM5: .ASCIZ \MNCAA (D/A) DAC3 REGISTER IN ERROR\
    010710 042050 040457 004451
    010716 040504 031503 051040
    010724 043505 051511 042524
    010732 020122 047111 042440
    010740 051122 051117 000
950 010745 115 041516 040501 EM6: .ASCIZ /MNCAA SELECTED DAC OFFSET POT WAS ADJUSTED INCORRECTLY/
    010752 051411 046105 041505
    010760 042524 020104 040504
    010766 020103 043117 051506
    010774 052105 050040 052117
    011002 053440 051501 040440
    011010 045104 051525 042524
    011016 020104 047111 047503
    011024 051122 041505 046124
    011032 000131
951 011034 047115 040503 004501 EM7: .ASCIZ /MNCAA SELECTED DAC GAIN POT WAS ADJUSTED INCORRECTLY/
    011042 042523 042514 052103
    011050 042105 042040 041501
    011056 043440 044501 020116
    011064 047520 020124 040527
    011072 020123 042101 052512

```

	011100	052123	042105	044440			
	011106	041516	051117	042522			
	011114	052103	054514	000			
952	011121	115	041516	040501	EM10:	.ASCIZ /MNCAA	SELECTED DAC HAS A LINEARITY PROBLEM/
	011126	051411	046105	041505			
	011134	042524	020104	040504			
	011142	020103	040510	020123			
	011150	020101	044514	042516			
	011156	051101	052111	020131			
	011164	051120	041117	042514			
	011172	000115					
953	011174	047115	040503	004501	EM11:	.ASCIZ /MNCAA	+15 VOLT SUPPLY IS INCORRECT/
	011202	030453	020065	047526			
	011210	052114	051440	050125			
	011216	046120	020131	051511			
	011224	044440	041516	051117			
	011232	042522	052103	000			
954	011237	115	041516	040501	EM12:	.ASCIZ /MNCAA	-15 VOLT SUPPLY IS INCORRECT/
	011244	026411	032461	053040			
	011252	046117	020124	052523			
	011260	050120	054514	044440			
	011266	020123	047111	047503			
	011274	051122	041505	000124			
955	011302	047115	040503	004501	EM13:	.ASCIZ /MNCAA	DAC #3 DIGITAL OUTPUT BITS IN ERROR/
	011310	040504	020103	031443			
	011316	042040	043511	052111			
	011324	046101	047440	052125			
	011332	052520	020124	044502			
	011340	051524	044440	020116			
	011346	051105	047522	000122			
956	011354	046120	040505	042523	EM14:	.ASCIZ /PLEASE	ADJUST THE POT/
	011362	040440	045104	051525			
	011370	020124	044124	020105			
	011376	047520	000124				
957	011402	047115	040503	004501	EM15:	.ASCIZ /MNCAA	INCORRECT I.D. VALUE CODE/
	011410	047111	047503	051122			
	011416	041505	020124	027111			
	011424	027104	053040	046101			
	011432	042525	041440	042117			
	011440	000105					
958	011442	042523	042514	052103	EM16:	.ASCIZ /SELECTED	DAC HAS A FRONT PANEL SWITCH PROBLEM/
	011450	042105	042040	041501			
	011456	044040	051501	040440			
	011464	043040	047522	052116			
	011472	050040	047101	046105			
	011500	051440	044527	041524			
	011506	020110	051120	041117			
	011514	042514	000115				
959	011520	047115	040503	020101	EM17:	.ASCIZ \MNCAA	(D/A) EXISTING MNCAA NOW FAIL'S TO RESPOND\
	011526	042050	040457	004451			
	011534	054105	051511	044524			
	011542	043516	046440	041516			
	011550	040501	047040	053517			
	011556	043040	044501	023514			

011564	020123	047524	051040		
011572	051505	047520	042116		
011600	000				
960 011601	015	004412	042504	LDSPAC: .ASCIZ	<15><12>/
011606	051120	051505	020123		DEPRESS THE 'RETURN' KEY WHEN DONE/
011614	044124	020105	051042		
011622	052105	051125	021116		
011630	045440	054505	053440		
011636	042510	020116	047504		
011644	042516	000			
961 011647	015	003412	047503	MSGSW: .ASCIZ	<15><12><7>/CONNECT TESTER MINC-AA DWARF TO M.U.T./<15><12>
011654	047116	041505	020124		
011662	042524	052123	051105		
011670	046440	047111	026503		
011676	040501	042040	040527		
011704	043122	052040	020117		
011712	027115	027125	027124		
011720	005015	000			
962 011723	123	052105	040440	FRONT: .ASCIZ	\SET ALL MINC-AA SWITCHES TO '+/-' AND 5 VOLT\<15><12>
011730	046114	046440	047111		
011736	026503	040501	051440		
011744	044527	041524	042510		
011752	020123	047524	021040		
011760	027453	021055	040440		
011766	042116	032440	053040		
011774	046117	006524	000012		
963 012002	035440	047524	040524	ERRTOT: .ASCIZ	/ ;TOTAL ERROR COUNT = /
012010	020114	051105	047522		
012016	020122	047503	047125		
012024	020124	020075	000		
964 012031	040	041073	042101	MESGD: .ASCIZ	/ ;BAD UNITS /
012036	052440	044516	051524		
012044	000040				
965 012046	015	012		FOUND1: .BYTE	15,12
966 012050	051120	043517	040522	.ASCIZ	/PROGRAM DETECTED /
012056	020115	042504	042524		
012064	052103	042105	000040		
967 012072	046440	041516	040501	FOUND2: .ASCIZ	\ MNCAA (D/A)'S \
012100	024040	027504	024501		
012106	051447	020040	000040		
968 012114	047125	052111	042411	DH1: .ASCIZ	/UNIT ERRPC BUSADR EXPECT WAS/
012122	051122	041520	041011		
012130	051525	042101	020122		
012136	042440	050130	041505		
012144	020124	020040	040527		
012152	000123				
969 012154	047125	052111	042411	DH2: .ASCIZ	/UNIT ERRPC BUSADR/
012162	051122	041520	041011		
012170	051525	042101	000122		
970 012176	047125	052111	042411	DH6: .ASCIZ	/UNIT ERRPC BUSADR EXPECT WAS SPREAD/
012204	051122	041520	041011		
012212	051525	042101	004522		
012220	054105	042520	052103		
012226	053411	051501	051411		

971	012234	051120	040505	000104					
	012242	047125	052111	042411	DH17:	.ASCIZ	/UNIT	ERRPC	WERE ARE/
	012250	051122	041520	053411					
	012256	051105	004505	051101					
	012264	000105							
975	012266	040440	045104	051525	ADJR19:	.ASCIZ	/ ADJUST	R19	/
(1)	012274	020124	030522	020071					
(1)	012302	020040	000						
976	012305	040	042101	052512	ADJR18:	.ASCIZ	/ ADJUST	R18	/
(1)	012312	052123	051040	034061					
(1)	012320	020040	000040						
977	012324	040440	045104	051525	ADJR21:	.ASCIZ	/ ADJUST	R21	/
(1)	012332	020124	031122	020061					
(1)	012340	020040	000						
978	012343	040	042101	052512	ADJR20:	.ASCIZ	/ ADJUST	R20	/
(1)	012350	052123	051040	030062					
(1)	012356	020040	000040						
979	012362	040440	045104	051525	ADJR42:	.ASCIZ	/ ADJUST	R42	/
(1)	012370	020124	032122	020062					
(1)	012376	020040	000						
980	012401	040	042101	052512	ADJR41:	.ASCIZ	/ ADJUST	R41	/
(1)	012406	052123	051040	030464					
(1)	012414	020040	000040						
981	012420	040440	045104	051525	ADJR44:	.ASCIZ	/ ADJUST	R44	/
(1)	012426	020124	032122	020064					
(1)	012434	020040	000						
982	012437	040	042101	052512	ADJR43:	.ASCIZ	/ ADJUST	R43	/
(1)	012444	052123	051040	031464					
(1)	012452	020040	000040						
983	012456	015	012		SELDX:	.BYTE	15,12		
984	012460	047503	047116	041505		.ASCIZ	/CONNECT	DVM TO THE OUTPUT OF/	
	012466	020124	053104	020115					
	012474	047524	052040	042510					
	012502	047440	052125	052520					
	012510	020124	043117	000					
985	012515	015	051412	046105	TWOTXT:	.ASCIZ	<15><12>\SELECT	+/- AND 2.5 VOLT ON \	
	012522	041505	020124	027453					
	012530	020055	047101	020104					
	012536	027062	020065	047526					
	012544	052114	047440	020116					
	012552	000							
986	012553	015	051412	046105	TENTXT:	.ASCIZ	<15><12>\SELECT	+/- AND 10 VOLT ON \	
	012560	041505	020124	027453					
	012566	020055	047101	020104					
	012574	030061	053040	046117					
	012602	020124	047117	000040					
987	012610	005015	042523	042514	UNITXT:	.ASCIZ	<15><12>\SELECT	+ AND 0-10 VOLT ON \	
	012616	052103	025440	040440					
	012624	042116	030040	030455					
	012632	020060	047526	052114					
	012640	047440	020116	000					
988	012645	015	051412	046105	BIPTXT:	.ASCIZ	<15><12>\SELECT	+/- AND 5 VOLT ON \	
	012652	041505	020124	027453					
	012660	020055	047101	020104					

012666	020065	047526	052114		
012674	047440	020116	000		
989					
993	012701	040	040504	030103	SELDO: .ASCIZ / DAC0 /
(1)	012706	000040			
994	012710	042040	041501	020061	SEL1: .ASCIZ / DAC1 /
(1)	012716	000			
995	012717	040	040504	031103	SEL2: .ASCIZ / DAC2 /
(1)	012724	000040			
996	012726	042040	041501	020063	SEL3: .ASCIZ / DAC3 /
(1)	012734	000			
997	012735	015	040412	045104	TRYAGN: .ASCIZ <15><12>/ADJUST THAT SAME POT AGAIN PLEASE/<15><12>
	012742	051525	020124	044124	
	012750	052101	051440	046501	
	012756	020105	047520	020124	
	012764	043501	044501	020116	
	012772	046120	040505	042523	
	013000	005015	000		
998	013003	015	012		N51200: .BYTE 15,12
999	013005	105	050130	041505	.ASCIZ /EXPECTED OUTPUT VOLTAGE IS -5.12000 VOLTS /
	013012	042524	020104	052517	
	013020	050124	052125	053040	
	013026	046117	040524	042507	
	013034	044440	020123	032455	
	013042	030456	030062	030060	
	013050	053040	046117	051524	
	013056	020040	000		
1000	013061	015	012		P51175: .BYTE 15,12
1001	013063	105	050130	041505	.ASCIZ /EXPECTED OUTPUT VOLTAGE IS +5.11750 VOLTS /
	013070	042524	020104	052517	
	013076	050124	052125	053040	
	013104	046117	040524	042507	
	013112	044440	020123	032453	
	013120	030456	033461	030065	
	013126	053040	046117	051524	
	013134	020040	000		
1002	013137	015	012		PRIME0: .BYTE 15,12
1003	013141	114	036440	046040	.ASCII \L = LOGIC TEST\
	013146	043517	041511	052040	
	013154	051505	124		
1004	013157	015	012		.BYTE 15,12
1005	013161	122	036440	051040	.ASCII \R RAMP OUTPUT LOOP\
	013166	046501	020120	052517	
	013174	050124	052125	046040	
	013202	047517	120		
1006	013205	015	012		.BYTE 15,12
1007	013207	123	036440	051440	.ASCII \S STATIC CALIBRATION LOOP\
	013214	040524	044524	020103	
	013222	040503	044514	051102	
	013230	052101	047511	020116	
	013236	047514	050117		
1008	013242	015	012		.BYTE 15,12
1009	013244	020104	020075	054504	.ASCII \D - DYNAMIC CALIBRATION LOOP\
	013252	040516	044515	020103	

	013260	040503	044514	051102	
	013266	052101	047511	020116	
	013274	047514	050117		
1010	013300	015	012		.BYTE 15,12
1011	013302	020102	020075	040502	.ASCII \B - BASE ADDRESS CHANGE\
	013310	042523	040440	042104	
	013316	042522	051523	041440	
	013324	040510	043516	105	
1012	013331	015	012		.BYTE 15,12
1013	013333	117	036440	047440	.ASCII \O = OUTPUT TEST MODULE LED LOOP\
	013340	052125	052520	020124	
	013346	042524	052123	046440	
	013354	042117	046125	020105	
	013362	042514	020104	047514	
	013370	050117			
1014	013372	015	012		.BYTE 15,12
1015	013374	020107	020075	042507	.ASCII \G = GET NEW SWITCH REGISTER VALUE\
	013402	020124	042516	020127	
	013410	053523	052111	044103	
	013416	051040	043505	051511	
	013424	042524	020122	040526	
	013432	052514	105		
1016	013435	015	012		.BYTE 15,12
1017	013437	110	036440	044040	.ASCII \H = HELP THE OPERATOR AND RETYPE THIS LIST \
	013444	046105	020120	044124	
	013452	020105	050117	051105	
	013460	052101	051117	040440	
	013466	042116	051040	052105	
	013474	050131	020105	044124	
	013502	051511	046040	051511	
	013510	020124	020040	000040	
1018	013516	015	012		.BYTE 15,12
1019	013520	054524	042520	052040	.ASCII \I = TEST CHARACTER THEN DEPRESS 'RETURN KEY' /
	013526	042510	021040	042524	
	013534	052123	041440	040510	
	013542	040522	052103	051105	
	013550	020042	044124	047105	
	013556	042040	050105	042522	
	013564	051523	021040	042522	
	013572	052524	047122	045440	
	013600	054505	020042	000040	
1020	013606	015	012		.BYTE 15,12
1021	013610	047115	040503	020101	.ASCII \MNCAA (D/A) BASE ADDRESS <\
	013616	042050	040457	020051	
	013624	040502	042523	040440	
	013632	042104	042522	051523	
	013640	036040	000		
1022	013643	076	037440	000040	.ASCII \> ? \
1023					.EVEN
1024	013650	002536	001116	001126	DT1: UNITBD,\$ERRPC,\$BDDAT,0
	013656	000000			
1025	013660	002536	001116	001446	DT2: UNITBD,\$ERRPC,DACO,\$GDDAT,\$BDDAT,0
	013666	001124	001126	000000	
1026	013674	002536	001116	001450	DT3: UNITBD,\$ERRPC,DAC1,\$GDDAT,\$BDDAT,0

CVMND-A MNCAA  
CVMNDA.P11

DIAGNOSTIC  
ASCII MESSAGES

MACY11 27(654) 19-SEP-78 08:58 <sup>L 4</sup> PAGE 27-6

SEQ 0050

1027	013702	001124	001126	000000					
	013710	002536	001116	001452	DT4:	UNITBD,\$ERRPC,DAC2,\$GDDAT,\$BDDAT,0			
	013716	001124	001126	000000					
1028	013724	002536	001116	001454	DT5:	UNITBD,\$ERRPC,DAC3,\$GDDAT,\$BDDAT,0			
	013732	001124	001126	000000					
1029	013740	002536	001116	010402	DT6:	UNITBD,\$ERRPC,DACBAD,\$GDDAT,\$BDDAT,SPREAD,0			
	013746	001124	001126	010420					
	013754	000000							
1030	013756	002536	001116	001202	DT17:	UNITBD,\$ERRPC,\$UNIT,EVER,0			
	013764	001444	000000						
1031	013770	000	000	000	DF0:	.BYTE 0,0,0,0,0,0,0,0			
	013773	000	000	000					
	013776	000	000						

```

1033
1034      .SBTTL  BINARY TO ASCII: AND TYPE ROUTINE
(1)
(2)      ;:*****
(1)      ;:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
(1)      ;:BINARY-ASCII NUMBER AND TYPE IT.
(1)      ;:CALL:
(1)      ;:*      MOV      NUMBER,-(SP)      ;;NUMBER TO BE TYPED
(1)      ;:*      TYPBN      ;;TYPE IT
(1)
(1) 014000 010146      $TYPBN: MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
(1) 014002 016601 000006      MOV      6(SP),R1      ;;GET THE INPUT NUMBER
(1) 014006 000261      SEC      ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1) 014010 112737 000060 014052 1$:  MOVB     #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
(1) 014016 006101      ROL      R1      ;;GET THIS BIT
(1) 014020 001406      BEQ      2$      ;;DONE?
(1) 014022 105537 014052      ADCB     $BIN      ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1) 014026 104401 014052      TYPE     , $BIN      ;;GO TYPE THIS BIT
(1) 014032 000241      CLC      ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
(1) 014034 000765      BR       1$      ;;GO DO THE NEXT BIT
(1) 014036 012601      MOV      (SP)+,R1      ;;POP THE STACK INTO R1
(1) 014040 016666 000002 000004      MOV      2(SP),4(SP)      ;;ADJUST THE STACK
(1) 014046 012616      MOV      (SP)+,(SP)
(1) 014050 000002      RTI      ;;RETURN TO USER
(1) 014052      000      000      $BIN: .BYTE 0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
1035      .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)
(2) 014054      $TYPDS:
(3) 014054 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 014056 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 014060 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(3) 014062 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
(3) 014064 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
(1) 014066 012746 020200      MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
(1) 014072 016605 000020      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
(1) 014076 100004      BPL      1$      ;;BR IF INPUT IS POS.
(1) 014100 005405      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
(1) 014102 112766 000055 000001 1$:  MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
(1) 014110 005000      CLR      R0      ;;ZERO THE CONSTANTS INDEX
(1) 014112 012703 014270      MOV      #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
(1) 014116 112723 000040      MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
(1) 014122 005002      CLR      R2      ;;CLEAR THE BCD NUMBER
(1) 014124 016001 014260      MOV      $DTBL(R0),R1      ;;GET THE CONSTANT
(1) 014130 160105      3$:  SUB      R1,R5      ;;FORM THIS BCD DIGIT

```



```

(1) 014132 002402          BLT      4$          ;;BR IF DONE
(1) 014134 005202          INC      R2          ;;INCREASE THE BCD DIGIT BY 1
(1) 014136 000774          BR       3$
(1) 014140 060105          4$: ADD     R1,R5      ;;ADD BACK THE CONSTANT
(1) 014142 005702          TST     R2          ;;CHECK IF BCD DIGIT=0
(1) 014144 001002          BNE     5$          ;;FALL THROUGH IF 0
(1) 014146 105716          TSTB   (SP)        ;;STILL DOING LEADING 0'S?
(1) 014150 100407          BMI     7$          ;;BR IF YES
(1) 014152 106316          5$: ASLB   (SP)        ;;MSD?
(1) 014154 103003          BCC     6$          ;;BR IF NO
(1) 014156 116663 000001 177777 MOVB   1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 014164 052702 000060          6$: BIS     #'0,R2   ;;MAKE THE BCD DIGIT ASCII
(1) 014170 052702 000040          7$: BIS     #' ,R2   ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 014174 110223          MOVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 014176 005720          TST    (R0)+       ;;JUST INCREMENTING
(1) 014200 020027 000010          CMP    R0,#10     ;;CHECK THE TABLE INDEX
(1) 014204 002746          BLT    2$          ;;GO DO THE NEXT DIGIT
(1) 014206 003002          BGT    8$          ;;GO TO EXIT
(1) 014210 010502          MOV    R5,R2      ;;GET THE LSD
(1) 014212 000764          BR     6$          ;;GO CHANGE TO ASCII
(1) 014214 105726          8$: TSTB   (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 014216 100003          BPL    9$          ;;BR IF NO
(1) 014220 116663 177777 177776 MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 014226 105013          9$: CLRB   (R3)     ;;SET THE TERMINATOR
(3) 014230 012605          MOV    (SP)+,R5   ;;POP STACK INTO R5
(3) 014232 012603          MOV    (SP)+,R3   ;;POP STACK INTO R3
(3) 014234 012602          MOV    (SP)+,R2   ;;POP STACK INTO R2
(3) 014236 012601          MOV    (SP)+,R1   ;;POP STACK INTO R1
(3) 014240 012600          MOV    (SP)+,R0   ;;POP STACK INTO R0
(1) 014242 104401 014270          TYPE   ,SDBLK     ;;NOW TYPE THE NUMBER
(1) 014246 016666 000002 000004 MOV    2(SP),4(SP) ;;ADJUST THE STACK
(1) 014254 012616          MOV    (SP)+,(SP)
(1) 014256 000002          RTI
(1) 014260 023420          $DTBL: 10000.    ;;RETURN TO USER
(1) 014262 001750          1000.
(1) 014264 000144          100.
(1) 014266 000012          10.
(1) 014270 000004          $DBLK: .BLKW   4

```

```

1037      .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)      :*SW11=1      INHIBIT ITERATIONS
(1)      :*SW09 1      LOOP ON ERROR
(1)      :*SW08-1     LOOP ON TEST IN SWR<7:0>
(1)      :*CALL
(1)      :*      SCOPE      ;;SCOPE=IOT
(1)
(1)      $SCOPE:
(1) 014300      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 014300 104410      JSR      PC,CTRLCG
(2) 014302 004737 010174      BIT      #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
(1) 014306 032777 040000 164624 1$:      BNE      $OVER      ;;YES IF SW14-1
(1) 014314 001114      :*****START OF CODE FOR THE XOR TESTER*****
(1) 014316 000416      $XTSTR: BR      6$      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)      :THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1) 014320 013746 000004      MOV      @ERRVEC,-(SP)      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 014324 012737 014344 000004      MOV      #5$,@ERRVEC      ;;SET FOR TIMEOUT
(1) 014332 005737 177060      TST      @177060      ;;TIME OUT ON XOR?
(1) 014336 012637 000004      MOV      (SP)+,@ERRVEC      ;;RESTORE THE ERROR VECTOR
(1) 014342 000463      BR      $$VLAD      ;;GO TO THE NEXT TEST
(1) 014344 022626      5$:      CMP      (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
(1) 014346 012637 000004      MOV      (SP)+,@ERRVEC      ;;RESTORE THE ERROR VECTOR
(1) 014352 000423      BR      7$      ;;LOOP ON THE PRESENT TEST
(1) 014354      6$:;*****END OF CODE FOR THE XOR TESTER*****
(1) 014354 032777 000400 164556      BIT      #BIT08,@SWR      ;;LOOP ON SPEC. TEST?
(1) 014362 001404      BEQ      2$      ;;BR IF NO
(1) 014364 127737 164550 001102      CMPB     @SWR,$TSTNM      ;;ON THE RIGHT TEST? SWR<7:0>
(1) 014372 001465      BEQ      $OVER      ;;BR IF YES
(1) 014374 105737 001103      2$:      TSTB     $ERFLG      ;;HAS AN ERROR OCCURRED?
(1) 014400 001421      BEQ      3$      ;;BR IF NO
(1) 014402 123737 001115 001103      CMPB     $ERMAX,$ERFLG      ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 014410 101015      BHI      3$      ;;BR IF NO
(1) 014412 032777 001000 164520      BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
(1) 014420 001404      BEQ      4$      ;;BR IF NO
(1) 014422 013737 001110 001106 7$:      MOV      $LPERR,$LPADR      ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 014430 000446      BR      $OVER
(1) 014432 105037 001103      4$:      CLRB     $ERFLG      ;;ZERO THE ERROR FLAG
(1) 014436 005037 001160      CLR      $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 014442 000415      BR      1$      ;;ESCAPE TO THE NEXT TEST
(1) 014444 032777 004000 164466 3$:      BIT      #BIT11,@SWR      ;;INHIBIT ITERATIONS?
(1) 014452 001011      BNE      1$      ;;BR IF YES
(1) 014454 005737 001176      TST      $PASS      ;;IF FIRST PASS OF PROGRAM
(1) 014460 001406      BEQ      1$      ;;      INHIBIT ITERATIONS
(1) 014462 005237 001104      INC      $ICNT      ;;INCREMENT ITERATION COUNT
(1) 014466 023737 001160 001104      CMP      $TIMES,$ICNT      ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 014474 002024      BGE      $OVER      ;;BR IF MORE ITERATION REQUIRED
(1) 014476 012737 000001 001104 1$:      MOV      #1,$ICNT      ;;REINITIALIZE THE ITERATION COUNTER

```

```

(1) 014504 013737 014562 001160      MOV      $MXCNT,$TIMES      ;;SET NUMBER OF ITERATIONS TO DO
(1) 014512 105237 001102      $SVLAD: INCB      $TSTNM      ;;COUNT TEST NUMBERS
(1) 014516 113737 001102 001174      MOV      $TSTNM,$TESTN    ;;SET TEST NUMBER IN APT MAILBOX
(1) 014524 011637 001106      MOV      (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
(1) 014530 011637 001110      MOV      (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
(1) 014534 005037 001162      CLR      $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 014540 112737 000001 001115      MOV      #1,$SERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 014546 013777 001102 164366      $OVER:  MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 014554 013716 001106      MOV      $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
(1) 014560 000002      RTI                          ;;FIXES PS
(1) 014562 003720      $MXCNT: 2000.                ;;MAX. NUMBER OF ITERATIONS
1050      .SBTTL  ERROR HANDLER ROUTINE
(1)
(2)      ;*****
(1)      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)      ;*AND GO TO $ERRTYP ON ERROR
(1)      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)      ;*SW15=1      HALT ON ERROR
(1)      ;*SW13=1      INHIBIT ERROR TYPEOUTS
(1)      ;*SW09=1      LOOP ON ERROR
(1)      ;*CALL
(1)      ;*      ERROR      N      ;;ERROR-EMT AND N-ERROR ITEM NUMBER
(1)
(1) 014564      $ERROR:
(1) 014564 104410      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(3) 014566 004737 010174      JSR      PC,CTRLCG      ;;TEST FOR CTRL C/G
(3) 014572 053737 010406 010404      BIS      MASKNM,BADUNT
(3) 014600 013737 010406 014630      MOV      MASKNM,11$      ;;GET CURRENT UNIT
(3) 014606 012737 000000 002536      MOV      #0,UNITBD      ;;PRIME THE UNIT #
(3) 014614 006237 014630      10$:     ASR      11$      ;;MOVE OUT
(3) 014620 001404      BEQ      12$      ;;BR WHEN DONE
(3) 014622 005237 002536      INC      UNITBD      ;;UPDATE VALUE
(3) 014626 000772      BR      10$
(3) 014630 000000      11$:     0
(3) 014632 000240      12$:     NOP
(1) 014634 105237 001103      7$:     INCB      $ERFLG      ;;SET THE ERROR FLAG
(1) 014640 001775      BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
(1) 014642 013777 001102 164272      MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 014650 005237 001112      INC      $ERTTL      ;;INC THE ERROR COUNT
(1) 014654 011637 001116      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 014660 162737 000002 001116      SUB      #2,$ERRPC
(1) 014666 117737 164224 001114      MOV      @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 014674 032777 020000 164236      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
(1) 014702 001004      BNE      20$      ;;SKIP TYPEOUTS
(1) 014704 004737 015016      JSR      PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(1) 014710 104401 001165      TYPE      ,$CRLF
(1) 014714
(1) 014714 122737 000001 001210      20$:     CMPB     #APTENV,$ENV      ;;RUNNING IN APT MODE
(1) 014722 001007      BNE      2$      ;;NO,SKIP APT ERROR REPORT
(1) 014724 113737 001114 014736      MOV      $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 014732 004737 017572      JSR      PC,$ATY4      ;;REPORT FATAL ERROR TO APT
(1) 014736 000
(1) 014737 000      21$:     .BYTE  0
           .BYTE  0

```

```

(1) 014740 000777          22$: BR          22$      ;;APT ERROR LOOP
(1) 014742 005777 164172  2$:  TST          @SWR      ;;HALT ON ERROR
(1) 014746 100002          3$:  BPL          3$      ;;SKIP IF CONTINUE
(1) 014750 000000          4$:  HALT          ;;HALT ON ERROR!
(1) 014752 104410          5$:  CKSWR         ;;TEST FOR CHANGE IN SOFT-SWR
(1) 014754 032777 001000 164156 3$:  BIT          #BIT09,@SWR  ;;LOOP ON ERROR SWITCH SET?
(1) 014762 001402          4$:  BEQ          4$      ;;BR IF NO
(1) 014764 013716 001110  4$:  MOV          $LPERR,(SP)  ;;FUDGE RETURN FOR LOOPING
(1) 014770 005737 001162  4$:  TST          $ESCAPE    ;;CHECK FOR AN ESCAPE ADDRESS
(1) 014774 001402          5$:  BEQ          5$      ;;BR IF NONE
(1) 014776 013716 001162  5$:  MOV          $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 015002          5$:          ;;
(1) 015002 022737 006522 000042 5$:  CMP          #SENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?
(1) 015010 001001          6$:  BNE          6$      ;;BRANCH IF NO
(1) 015012 000000          6$:  HALT          ;;YES
(1) 015014          6$:          ;;
(1) 015014 000002          6$:  RTI           ;;RETURN

```

1051 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

(1)
(2)
(1) ;*****
(1) ;*THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
(1) ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
(1) ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)

```

```

(1) 015016          $ERRTYP:
(1) 015016 104401 001165  TYPE          , $CRLF      ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 015022 010046          MOV          R0,-(SP)    ;;SAVE R0
(1) 015024 005000          CLR          RC           ;;PICKUP THE ITEM INDEX
(1) 015026 153700 001114  BISB          @#$ITEMB,R0
(1) 015032 001004          BNE          1$          ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                     ;;TYPE THE PC OF THE ERROR
(2) 015034 013746 001116  MOV          $ERRPC,-(SP)  ;;SAVE $ERRPC FOR TYPEOUT
(2)                                     ;;ERROR ADDRESS
(2) 015040 104402          TYPOC         ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 015042 000445          BR          10$         ;;GET OUT
(1) 015044 005300          1$: DEC          R0          ;;ADJUST THE INDEX SO THAT IT WILL
(1) 015046 006300          ASL          R0          ;; WORK FOR THE ERROR TABLE
(1) 015050 006300          ASL          R0
(1) 015052 006300          ASL          R0
(1) 015054 062700 001252  ADD          #$ERRTB,R0   ;;FORM TABLE POINTER
(1) 015060 012037 015070  MOV          (R0)+,2$    ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 015064 001404          BEQ          3$          ;;SKIP TYPEOUT IF NO POINTER
(1) 015066 104401          TYPE          ;;TYPE THE 'ERROR MESSAGE'
(1) 015070 000000          2$: .WORD          0     ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 015072 104401 001165  TYPE          , $CRLF    ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 015076 012037 015106  3$: MOV          (R0)+,4$  ;;PICKUP 'DATA HEADER' POINTER
(1) 015102 001404          BEQ          5$          ;;SKIP TYPEOUT IF 0
(1) 015104 104401          TYPE          ;;TYPE THE 'DATA HEADER'
(1) 015106 000000          4$: .WORD          0     ;;'DATA HEADER' POINTER GOES HERE
(1) 015110 104401 001165  TYPE          , $CRLF    ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 015114 010146          5$: MOV          R1,-(SP)   ;;SAVE R1
(1) 015116 012001          MOV          (R0)+,R1    ;;PICKUP 'DATA TABLE' POINTER
(1) 015120 001415          BEQ          9$          ;;BR IF NO DATA TO BE TYPED
(1) 015122 012000          MOV          (R0)+,R0   ;;PICKUP 'DATA FORMAT' POINTER

```

```

(1) 015124 105720      6$:  TSTB   (R0)+      ;; 'OCTAL' OR 'DECIMAL'
(1) 015126 001003      BNE    7$          ;; BR IF DECIMAL
(2) 015130 013146      MOV    @ (R1)+, -(SP) ;; SAVE @ (R1)+ FOR TYPEOUT
(2) 015132 104402      TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 015134 000402      BR     8$
(2) 015136          7$:  MOV    @ (R1)+, -(SP) ;; SAVE @ (R1)+ FOR TYPEOUT
(2) 015136 013146      TYPDS          ;; GO TYPE--DECIMAL ASCII WITH SIGN
(2) 015140 104405      TST    (R1)      ;; IS THERE ANOTHER NUMBER?
(1) 015142 005711      BEQ    9$          ;; BR IF NO
(1) 015144 001403      TYPE    ,11$     ;; TYPE TWO(2) SPACES
(1) 015146 104401 015166 BR     6$          ;; LOOP
(1) 015152 000764
(1) 015154 012601      9$:  MOV    (SP)+, R1   ;; RESTORE R1
(1) 015156 012600      10$: MOV    (SP)+, R0  ;; RESTORE R0
(1) 015160 104401 001165 TYPE    , $CRLF    ;; 'CARRIAGE RETURN' & 'LINE FEED'
(1) 015164 000207      RTS    PC         ;; RETURN
(1) 015166 020040 000 11$:  .ASCIZ / /        ;; TWO(2) SPACES
(1) 015172 015172      .EVEN
1052 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)
(1)
*****
: POWER DOWN ROUTINE
(1) 015172 012737 015336 000024 $PWRDN: MOV    # $ILLUP, @ #PWRVEC ;; SET FOR FAST UP
(1) 015200 012737 000340 000026      MOV    # 340, @ #PWRVEC+2 ;; PRIO:7
(3) 015206 010046      MOV    R0, -(SP)    ;; PUSH R0 ON STACK
(3) 015210 010146      MOV    R1, -(SP)    ;; PUSH R1 ON STACK
(3) 015212 010246      MOV    R2, -(SP)    ;; PUSH R2 ON STACK
(3) 015214 010346      MOV    R3, -(SP)    ;; PUSH R3 ON STACK
(3) 015216 010446      MOV    R4, -(SP)    ;; PUSH R4 ON STACK
(3) 015220 010546      MOV    R5, -(SP)    ;; PUSH R5 ON STACK
(3) 015222 017746 163712 MOV    @SWR, -(SP)   ;; PUSH @SWR ON STACK
(1) 015226 010637 015342 MOV    SP, $SAVR6    ;; SAVE SP
(1) 015232 012737 015244 000024 MOV    # $PWRUP, @ #PWRVEC ;; SET UP VECTOR
(1) 015240 000000      HALT
(1) 015242 000776      BR     .-2         ;; HANG UP
(1)
(2)
*****
: POWER UP ROUTINE
(1) 015244 012737 015336 000024 $PWRUP: MOV    # $ILLUP, @ #PWRVEC ;; SET FOR FAST DOWN
(1) 015252 013706 015342      MOV    $SAVR6, SP   ;; GET SP
(1) 015256 005037 015342      CLR    $SAVR6       ;; WAIT LOOP FOR THE TTY
(1) 015262 005237 015342      1$:  INC    $SAVR6      ;; WAIT FOR THE INC
(1) 015266 001375      BNE    1$           ;; OF WORD
(3) 015270 012677 163644      MOV    (SP)+, @SWR  ;; POP STACK INTO @SWR
(3) 015274 012605      MOV    (SP)+, R5    ;; POP STACK INTO R5
(3) 015276 012604      MOV    (SP)+, R4    ;; POP STACK INTO R4
(3) 015300 012603      MOV    (SP)+, R3    ;; POP STACK INTO R3
(3) 015302 012602      MOV    (SP)+, R2    ;; POP STACK INTO R2
(3) 015304 012601      MOV    (SP)+, R1    ;; POP STACK INTO R1
(3) 015306 012600      MOV    (SP)+, R0    ;; POP STACK INTO R0
(1) 015310 012737 015172 000024 MOV    # $PWRDN, @ #PWRVEC ;; SET UP THE POWER DOWN VECTOR
(1) 015316 012737 000340 000026 MOV    # 340, @ #PWRVEC+2 ;; PRIO:7
(1) 015324 104401      TYPE

```



```

(1) 015474 062704 000006      ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 015500 110437 015636      MOV      R4,$OMODE     ;;SAVE IT FOR USE
(1) 015504 113704 015635      MOV      $OFILL,R4     ;;GET THE ZERO FILL SWITCH
(1) 015510 016605 000012      MOV      12(SP),R5     ;;PICKUP THE INPUT NUMBER
(1) 015514 005003              CLR      R3            ;;CLEAR THE OUTPUT WORD
(1) 015516 006105              1$:      ROL      R5            ;;ROTATE MSB INTO 'C'
(1) 015520 000404              BR       3$            ;;GO DO MSB
(1) 015522 006105              2$:      ROL      R5            ;;FORM THIS DIGIT
(1) 015524 006105              ROL      R5
(1) 015526 006105              ROL      R5
(1) 015530 010503              MOV      R5,R3
(1) 015532 006103              3$:      ROL      R3            ;;GET LSB OF THIS DIGIT
(1) 015534 105337 015636      DECB    $OMODE         ;;TYPE THIS DIGIT?
(1) 015540 100016              BPL     7$            ;;BR IF NO
(1) 015542 042703 177770      BIC     #177770,R3     ;;GET RID OF JUNK
(1) 015546 001002              BNE     4$            ;;TEST FOR 0
(1) 015550 005704              TST     R4            ;;SUPPRESS THIS 0?
(1) 015552 001403              BEQ     5$            ;;BR IF YES
(1) 015554 005204              4$:      INC     R4            ;;DON'T SUPPRESS ANYMORE 0'S
(1) 015556 052703 000060      BIS     #'0,R3         ;;MAKE THIS DIGIT ASCII
(1) 015562 052703 000040      5$:      BIS     #' ,R3         ;;MAKE ASCII IF NOT ALREADY
(1) 015566 110337 015632      MOV     R3,8$          ;;SAVE FOR TYPING
(1) 015572 104401 015632      TYPE    8$            ;;GO TYPE THIS DIGIT
(1) 015576 105337 015634      7$:      DECB    $OCNT         ;;COUNT BY 1
(1) 015602 003347              BGT     2$            ;;BR IF MORE TO DO
(1) 015604 002402              BLT     6$            ;;BR IF DONE
(1) 015606 005204              INC     R4            ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 015610 000744              BR      2$            ;;GO DO THE LAST DIGIT
(1) 015612 012605              6$:      MOV     (SP)+,R5       ;;RESTORE R5
(1) 015614 012604              MOV     (SP)+,R4       ;;RESTORE R4
(1) 015616 012603              MOV     (SP)+,R3       ;;RESTORE R3
(1) 015620 016666 000002 000004  MOV     2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
(1) 015626 012616              MOV     (SP)+,(SP)
(1) 015630 000002              RTI                    ;;RETURN
(1) 015632 000          8$:      .BYTE  0              ;;STORAGE FOR ASCII DIGIT
(1) 015633 000          .BYTE  0              ;;TERMINATOR FOR TYPE ROUTINE
(1) 015634 000          $OCNT: .BYTE  0              ;;OCTAL DIGIT COUNTER
(1) 015635 000          $OFILL: .BYTE  0          ;;ZERO FILL SWITCH
(1) 015636 000000      $OMODE: .WORD  0          ;;NUMBER OF DIGITS TO TYPE
1057 ;*****CAUTION THE FIRST 4 LOC. ARE OVERLAYED TO LOWER INTERRUPT LEVEL
1058 ;THE OVERLAY OCCURS AFTER THE "SETUP" CODE
1059 .SBTTL TYPE ROUTINE
(1)
(2)
(1) ;*****
(1) ;*ROUTINE TO TYPE ASCII 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 ^N ASCII STRING
(1) ;*OR

```

```

(1)          : *      TYPE
(1)          : *      MESADR
(1)          : *
(1)
(1) 015640 105737 001157 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
(1) 015644 100002          BPL      1$          ;; BR IF YES
(1) 015646 000000          HALT          ;; HALT HERE IF NO TERMINAL
(1) 015650 000430          BR      3$          ;; LEAVE
(1) 015652 010046          '$:  MOV      RO,-(SP)      ;; SAVE RO
(1) 015654 017600 000002  MOV      @2(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
(1) 015660 122737 000001 001210  CMPB   #APTENV,$ENV      ;; RUNNING IN APT MODE
(1) 015666 001011          BNE      62$          ;; NO,GO CHECK FOR APT CONSOLE
(1) 015670 132737 000100 001211  BITB   #APTSPOOL,$ENVM      ;; SPOOL MESSAGE TO APT
(1) 015676 001405          BEQ      62$          ;; NO,GO CHECK FOR CONSOLE
(1) 015700 010037 015710  MOV      RO,61$          ;; SETUP MESSAGE ADDRESS FOR APT
(1) 015704 004737 017562  JSR     PC,$ATY3          ;; SPOOL MESSAGE TO APT
(1) 015710 000000          .WORD   0          ;; MESSAGE ADDRESS
(1) 015712 132737 000040 001211 61$:  BITB   #APTCSUP,$ENVM      ;; APT CONSOLE SUPPRESSED
(1) 015720 001003          BNE      60$          ;; YES,SKIP TYPE OUT
(1) 015722 112046          2$:  MOVB   (RO)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 015724 001005          BNE      4$          ;; BR IF IT ISN'T THE TERMINATOR
(1) 015726 005726          TST     (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
(1) 015730 012600          60$:  MOV      (SP)+,RO      ;; RESTORE RO
(1) 015732 062716 000002  3$:  ADD     #2,(SP)          ;; ADJUST RETURN PC
(1) 015736 000002          RTI          ;; RETURN
(1) 015740 122716 000011  4$:  CMPB   #HT,(SP)          ;; BRANCH IF <HT>
(1) 015744 001430          BEQ      8$          ;; BRANCH IF NOT <CRLF>
(1) 015746 122716 000200  CMPB   #CRLF,(SP)
(1) 015752 001006          BNE      5$          ;; POP <CR><LF> EQUIV
(1) 015754 005726          TST     (SP)+          ;; TYPE A CR AND LF
(1) 015756 104401          TYPE
(1) 015760 001165          $CRLF
(1) 015762 105037 016116  CLRB   $CHARCNT      ;; CLEAR CHARACTER COUNT
(1) 015766 000755          BR      2$          ;; GET NEXT CHARACTER
(1) 015770 004737 016052  5$:  JSR     PC,$TYPEC      ;; GO TYPE THIS CHARACTER
(1) 015774 123726 001156  6$:  CMPB   $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
(1) 016000 001350          BNE      2$          ;; IF NO GO GET NEXT CHAR.
(1) 016002 013746 001154  MOV     $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
(1)          ;; AND THE NULL CHAR.
(1) 016006 105366 000001  7$:  DECB   1(SP)          ;; DOES A NULL NEED TO BE TYPED?
(1) 016012 002770          BLT     6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 016014 004737 016052  JSR     PC,$TYPEC      ;; GO TYPE A NULL
(1) 016020 105337 016116  DECB   $CHARCNT      ;; DO NOT COUNT AS A COUNT
(1) 016024 000770          BR      7$          ;; LOOP
(1)
(1)
(1)          ;HORIZONTAL TAB PROCESSOR
(1)
(1) 016026 112716 000040  8$:  MOVB   #' ,(SP)          ;; REPLACE TAB WITH SPACE
(1) 016032 004737 016052  9$:  JSR     PC,$TYPEC      ;; TYPE A SPACE
(1) 016036 132737 000007 016116  BITB   #7,$CHARCNT      ;; BRANCH IF NOT AT
(1) 016044 001372          BNE      9$          ;; TAB STOP
(1) 016046 005726          TST     (SP)+          ;; POP SPACE OFF STACK
(1) 016050 000724          BR      2$          ;; GET NEXT CHARACTER
(1) 016052 105777 163072  $TYPEC: TSTB  @2$TPS      ;; WAIT UNTIL PRINTER IS READY

```



```

(1) 016056 100375          BPL      $TYPEC
(1) 016060 116677 000002 163064    MOVB     2(SP),D$TPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 016066 122766 000015 000002    CMPB     #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
(1) 016074 001003          BNE      1$           ;;BRANCH IF NO
(1) 016076 105037 016116          CLRB     $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
(1) 016102 000406          BR       $TYPEX      ;;EXIT
(1) 016104 122766 000012 000002 1$:   CMPB     #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
(1) 016112 001402          BEQ      $TYPEX      ;;BRANCH IF YES
(1) 016114 105227          INCB     (PC)+       ;;COUNT THE CHARACTER
(1) 016116 000000          $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
(1) 016120 000207          $TYPEX: RTS         PC

```

```

(1) 1060          .SBTTL  TTY INPUT ROUTINE
(1)
(2)          ;;*****
(1)          .ENABL  LSB
(1) 016122 000000          $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
(1) 016124 000000          $TKQIN: .WORD 0     ;;INPUT POINTER
(1) 016126 000000          $TKQOUT: .WORD 0    ;;OUTPUT POINTER
(1) 016130 000040          $TKQSRT: .BLKB 32.  ;;TTY KEYBOARD QUEUE
(1) 016170          $TKQEND=.

```

```

(1)          ;*TK INITIALIZE ROUTINE
(1)          ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
(1)          ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
(1)          ;
(1)          ;*CALL:
(1)          ;*      JSR      PC,$TKINT
(1)          ;*      RETURN
(1)          ;
(1) 016170 005037 016122          $TKINT: CLR      $TKCNT    ;;CLEAR COUNT OF ITEMS IN QUEUE
(1) 016174 012737 016130 016124    MOV      #TKQSRT,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
(1) 016202 013737 016124 016126    MOV      $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1) 016210 012737 016240 000060    MOV      #TKSRV,@TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
(1) 016216 012737 000200 000062    MOV      #200,@TKVEC+2  ;;'BR' LEVEL 4
(1) 016224 005777 162716          TST     @TKB           ;;CLEAR DONE FLAG
(1) 016230 012777 000100 162706    MOV      #100,@TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
(1) 016236 000207          RTS      PC           ;;RETURN TO CALLER

```

```

(1)          ;*TK SERVICE ROUTINE
(1)          ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
(1)          ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
(1)          ;*IT IN THE QUEUE.
(1)          ;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
(1)          ;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (MTEST1)
(1)          ;
(1) 016240 117746 162702          $TKSRV: MOVB     @TKB,-(SP)  ;;PICKUP THE CHARACTER
(1) 016244 042716 177600          BIC     #^C177,(SP)    ;;STRIP THE JUNK
(1) 016250 021627 000003          CMP     (SP),#3       ;;IS IT A CONTROL C?
(1) 016254 001007          BNE     1$           ;;BRANCH IF NO
(1) 016256 104401 017410          TYPE    ,SCNTLC      ;;TYPE A CONTROL-C (^C)
(1) 016262 004737 016170          JSR     PC,$TKINT    ;;INIT THE KEYBOARD
(1) 016266 005726          TST     (SP)+       ;;CLEAN UP STACK
(1) 016270 000137 002314          JMP     MTEST1       ;;CONTROL C RESTART

```

```
(1) 016274 021627 000007 1$: CMP (SP),#7 ;;IS IT A CONTROL G?
(1) 016300 001004 BNE 2$ ;;BRANCH IF NO
(1) 016302 022737 000176 001140 CMP #SWREG,SWR ;;IS SOFT-SWR SELECTED?
(1) 016310 001500 BEQ 6$ ;;GO TO SWR CHANGE
(1)
(1) 016312 2$:
(1) 016312 022737 000040 016122 CMP #32,,$STKCNT ;;IS THE QUEUE FULL?
(1) 016320 001004 BNE 3$ ;;BRANCH IF NO
(1) 016322 104401 017404 TYPE $BELL ;;RING THE TTY BELL
(1) 016326 005726 TST (SP)+ ;;CLEAN CHARACTER OFF OF STACK
(1) 016330 000451 BR 5$ ;;EXIT
(1) 016332 021627 000023 3$: CMP (SP),#23 ;;IS IT A CONTROL-S?
(1) 016336 001021 BNE 32$ ;;BRANCH IF NO
(1) 016340 005077 162600 CLR @STKS ;;DISABLE TTY KEYBOARD INTERRUPTS
(1) 016344 005726 TST (SP)+ ;;CLEAN CHAR OFF STACK
(1) 016346 105777 162572 31$: TSTB @STKS ;;WAIT FOR A CHAR
(1) 016352 100375 BPL 31$ ;;LOOP UNTIL ITS THERE
(1) 016354 117746 162566 MOVB @STKB,-(SP) ;;GET THE CHARACTER
(1) 016360 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 016364 022627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
(1) 016370 001366 BNE 31$ ;;BRANCH IF NO
(1) 016372 012777 000100 162544 MOV #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
(1) 016400 000002 RTI ;;RETURN
(1) 016402 005237 016122 32$: INC $STKCNT ;;COUNT THIS CHARACTER
(1) 016406 021627 000140 CMP (SP),#140 ;;IS IT UPPER CASE?
(1) 016412 002405 BLT 4$ ;;BRANCH IF YES
(1) 016414 021627 000175 CMP (SP),#175 ;;IS IT A SPECIAL CHAR?
(1) 016420 003002 BGT 4$ ;;BRANCH IF YES
(1) 016422 042716 000040 BIC #40,(SP) ;;MAKE IT UPPER CASE
(1) 016426 112677 177472 4$: MOVB (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
(1) 016432 005237 016124 INC $STKQIN ;;UPDATE THE POINTER
(1) 016436 023727 016124 016170 CMP $STKQIN,$STKQEND ;;GO OFF THE END?
(1) 016444 001003 BNE 5$ ;;BRANCH IF NO
(1) 016446 012737 016130 016124 MOV $STKQSRT,$STKQIN ;;RESET THE POINTER
(1) 016454 000002 5$: RTI ;;RETURN
```

```
(1)
(2)
(1) *****
(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
(1) *CALL WHEN OPERATING IN TTY INTERRUPT MODE.
```

```
(1) 016456 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED
(1) 016464 001124 BNE 15$ ;;EXIT IF NOT
(1) 016466 105777 162452 TSTB @STKS ;;IS A CHAR WAITING?
(1) 016472 100121 BPL 15$ ;;IF NOT, EXIT
(1) 016474 117746 162446 MOVB @STKB,-(SP) ;;YES
(1) 016500 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 016504 021627 000007 CMP (SP),#7 ;;IS IT A CONTROL-G?
(1) 016510 001300 BNE 2$ ;;IF NOT, PUT IT IN THE TTY QUEUE
(1) ;;AND EXIT
```

```
(1)
(2)
(1) *****
(1) *CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
(1) *ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
```



```

(1) 016760 042726 000060      BIC      #60,(SP)+      ;;STRIP-OFF ASCII
(1) 016764 005766 000002      TST      2(SP)         ;;IS THIS THE FIRST CHAR
(1) 016770 001403              BEQ      17$           ;;BRANCH IF YES
(1) 016772 006316              ASL      (SP)         ;;NO, SHIFT PRESENT
(1) 016774 006316              ASL      (SP)         ;;  CHAR OVER TO MAKE
(1) 016776 006316              ASL      (SP)         ;;  ROOM FOR NEW ONE.
(1) 017000 005266 000002      17$:    INC      2(SP)         ;;KEEP COUNT OF CHAR
(1) 017004 056616 177776      BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
(1) 017010 000667              BR       7$           ;;GET THE NEXT ONE
(1) 017012 104401 001164      18$:    TYPE     ,SQUES   ;;TYPE ?<CR><LF>
(1) 017016 000720              BR       20$         ;;SIMULATE CONTROL-U
(1)                               .DSABL  LSB
(1)
(1)
(2)                               ;:*****
(1)                               ;:THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                               ;:CALL:
(1)                               ;:*      RDCHR              ;;GET A CHARACTER FROM THE QUEUE
(1)                               ;:*      RETURN HERE       ;;CHARACTER IS ON THE STACK
(1)                               ;:*                               ;;WITH PARITY BIT STRIPPED OFF
(1)                               ;:
(1)
(1) 017020 011646              $RDCHR: MOV      (SP),-(SP) ;;PUSH DOWN THE PC AND
(1) 017022 016666 000004 000002  MOV      4(SP),2(SP)    ;;THE PS
(1) 017030 005066 000004      CLR      4(SP)         ;;GET READY FOR A CHARACTER
(2) 017034 005046              CLR      -(SP)         ;;PUT NEW PS ON STACK
(2) 017036 012746 017044      MOV      #64$,-(SP)   ;;PUT NEW PC ON STACK
(2) 017042 000002              RTI                    ;;POP NEW PC AND PS
(2) 017044              64$:
(1) 017044 005737 016122      1$:    TST      $TKCNT     ;;WAIT ON A CHARACTER
(1) 017050 001775              BEQ      1$           ;;
(1) 017052 005337 016122      DEC      $TKCNT     ;;DECREMENT THE COUNTER
(1) 017056 117766 177044 000004  MOVB     @TKQOUT,4(SP)  ;;GET ONE CHARACTER
(1) 017064 005237 016126      INC      $TKQOUT     ;;UPDATE THE POINTER
(1) 017070 023727 016126 016170  CMP      $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
(1) 017076 001003              BNE     2$           ;;BRANCH IF NO
(1) 017100 012737 016130 016126  MOV      #$TKQSRT,$TKQOUT ;;RESET THE POINTER
(1) 017106 000002              RTI                    ;;RETURN
(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) 017110 010346              $RDLIN: MOV      R3,-(SP) ;;SAVE R3
(1) 017112 005046              CLR      -(SP)         ;;CLEAR THE RUBOUT KEY
(1) 017114 012703 017344      1$:    MOV      #$TTYIN,R3 ;;GET ADDRESS
(1) 017120 022703 017404      2$:    CMP      #$TTYIN+32.,R3 ;;BUFFER FULL?
(1) 017124 101456              BLOS     4$           ;;BR IF YES
(1) 017126 104411              RDCHR    ;;GO READ ONE CHARACTER FROM THE TTY
(1) 017130 112613              MOVB     (SP)+,(R3)    ;;GET CHARACTER
(1) 017132 122713 000177      10$:   CMPB     #177,(R3)    ;;IS IT A RUBOUT
(1) 017136 001022              BNE     5$           ;;BR IF NO

```

```

(1) 017140 005716          TST      (SP)          ;; IS THIS THE FIRST RUBOUT?
(1) 017142 001007          BNE      6$           ;; BR IF NO
(1) 017144 112737 000134 017342  MOVB     #' \ ,9$      ;; TYPE A BACK SLASH
(1) 017152 104401 017342          TYPE     ,9$
(1) 017156 012716 177777          MOV      #-1,(SP)     ;; SET THE RUBOUT KEY
(1) 017162 005303          6$: DEC      R3          ;; BACKUP BY ONE
(1) 017164 020327 017344          CMP      R3,#$TTYIN  ;; STACK EMPTY?
(1) 017170 103434          BLO      4$           ;; BR IF YES
(1) 017172 111337 017342          MOVB     (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
(1) 017176 104401 017342          TYPE     ,9$          ;; GO TYPE
(1) 017202 000746          BR       2$           ;; GO READ ANOTHER CHAR.
(1) 017204 005716          5$: TST      (SP)          ;; RUBOUT KEY SET?
(1) 017206 001406          BEQ      7$           ;; BR IF NO
(1) 017210 112737 000134 017342  MOVB     #' \ ,9$      ;; TYPE A BACK SLASH
(1) 017216 104401 017342          TYPE     ,9$
(1) 017222 005016          CLR      (SP)         ;; CLEAR THE RUBOUT KEY
(1) 017224 122713 000025          7$: CMPB     #25,(R3)    ;; IS CHARACTER A CTRL U?
(1) 017230 001003          BNE      8$           ;; BR IF NO
(1) 017232 104401 017415          TYPE     ,SCNTLU     ;; TYPE A CONTROL 'U'
(1) 017236 000726          BR       1$           ;; GO START OVER
(1) 017240 122713 000022          8$: CMPB     #22,(R3)    ;; IS CHARACTER A '^R'?
(1) 017244 001011          BNE      3$           ;; BRANCH IF NO
(1) 017246 105013          CLRB     (R3)         ;; CLEAR THE CHARACTER
(1) 017250 104401 001165          TYPE     ,$CRLF      ;; TYPE A 'CR' & 'LF'
(1) 017254 104401 017344          TYPE     ,$TTYIN     ;; TYPE THE INPUT STRING
(1) 017260 000717          BR       2$           ;; GO PICKUP ANOTHER CHACTER
(1) 017262 104401 001164          4$: TYPE     ,SQUES     ;; TYPE A '?'
(1) 017266 000712          BR       1$           ;; CLEAR THE BUFFER AND LOOP
(1) 017270 111337 017342          3$: MOVB     (R3),9$    ;; ECHO THE CHARACTER
(1) 017274 104401 017342          TYPE     ,9$
(1) 017300 122723 000015          CMPB     #15,(R3)+   ;; CHECK FOR RETURN
(1) 017304 001305          BNE      2$           ;; LOOP IF NOT RETURN
(1) 017306 105063 177777          CLRB     -1(R3)      ;; CLEAR RETURN (THE 15)
(1) 017312 104401 001166          TYPE     ,SLF        ;; TYPE A LINE FEED
(1) 017316 005726          TST      (SP)+       ;; CLEAN RUBOUT KEY FROM THE STACK
(1) 017320 012603          MOV      (SP)+,R3    ;; RESTORE R3
(1) 017322 011646          MOV      (SP),-(SP)  ;; ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 017324 016666 000004 000002  MOV      4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 017332 012766 017344 000004  MOV      #$TTYIN,4(SP)
(1) 017340 000002          RTI
(1) 017342 000          9$: .BYTE     0          ;; RETURN
(1) 017343 000          .BYTE     0          ;; STORAGE FOR ASCII CHAR. TO TYPE
(1) 017344 000040          $TTYIN: .BLKB     32.  ;; TERMINATOR
(1) 017404 177607 000377          $BELL:  .ASCIZ    <207><377><377> ;; RESERVE 32. BYTES FOR TTY INPUT
(1) 017410 041536 005015 000          $CNTLC: .ASCIZ    / ^C / <15> <12> ;; CODE FOR BELL
(1) 017415 136 006525 000012  $CNTLU: .ASCIZ    / ^U / <15> <12> ;; CONTROL 'C'
(1) 017422 043536 005015 000          $CNTLG: .ASCIZ    / ^G / <15> <12> ;; CONTROL 'U'
(1) 017427 015 051412 051127  $MSWR:  .ASCIZ    <15><12>/SWR = / ;; CONTROL 'G'
(1) 017434 036440 000040          $MNEW:  .ASCIZ    / NEW = /
(1) 017440 020040 042516 020127
(1) 017446 020075 000
(1) 017452          .EVEN

```

1062  
(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
1063  
(1)  
(2)

```
.SBTTL READ AN OCTAL NUMBER FROM THE TTY  
*****  
*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND  
*CHANGE IT TO BINARY.  
*CALL:  
*      RDOCT          ;; READ AN OCTAL NUMBER  
*      RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK  
*                ;; HIGH ORDER BITS ARE IN $HIOCT  
SRDOCT: MOV      (SP),-(SP)      ;; PROVIDE SPACE FOR THE  
        MOV      4(SP),2(SP)    ;; INPUT NUMBER  
        MOV      R0,-(SP)       ;; PUSH R0 ON STACK  
        MOV      R1,-(SP)       ;; PUSH R1 ON STACK  
        MOV      R2,-(SP)       ;; PUSH R2 ON STACK  
1$:     RDLIN                     ;; READ AN ASCII LINE  
        MOV      (SP)+,R0       ;; GET ADDRESS OF 1ST CHARACTER  
        CLR      R1               ;; CLEAR DATA WORD  
        CLR      R2  
2$:     MOVB     (R0)+,-(SP)     ;; PICKUP THIS CHARACTER  
        BEQ      3$              ;; IF ZERO GET OUT  
        ASL      R1              ;; *2  
        ROL      R2  
        ASL      R1              ;; *4  
        ROL      R2  
        ASL      R1              ;; *8  
        ROL      R2  
        BIC      #*(7,(SP)       ;; STRIP THE ASCII JUNK  
        ADD      (SP)+,R1        ;; ADD IN THIS DIGIT  
        BR       2$              ;; LOOP  
3$:     TST      (SP)+           ;; CLEAN TERMINATOR FROM STACK  
        MOV      R1,12(SP)      ;; SAVE THE RESULT  
        MOV      R2,$HIOCT  
        MOV      (SP)+,R2       ;; POP STACK INTO R2  
        MOV      (SP)+,R1       ;; POP STACK INTO R1  
        MOV      (SP)+,R0       ;; POP STACK INTO R0  
        RTI                     ;; RETURN  
$HIOCT: .WORD    0              ;; HIGH ORDER BITS GO HERE  
.SBTTL APT COMMUNICATIONS ROUTINE  
*****
```

```
$ATY1:  MOVB     #1,$FFLG        ;; TO REPORT FATAL ERROR  
$ATY3:  MOVB     #1,$MFLG        ;; TO TYPE A MESSAGE  
        BR       $ATYC  
$ATY4:  MOVB     #1,$FFLG        ;; TO ONLY REPORT FATAL ERROR  
$ATYC:  MOV      R0,-(SP)       ;; PUSH R0 ON STACK  
        MOV      R1,-(SP)       ;; PUSH R1 ON STACK  
        TSTB     $MFLG           ;; SHOULD TYPE A MESSAGE?  
        BEQ      5$              ;; IF NOT: BR  
        CMPB     #APTENV,$ENV     ;; OPERATING UNDER APT?  
        BNE      3$              ;; IF NOT: BR  
        BITB     #APTSPOOL,$ENVM  ;; SHOULD SPOOL MESSAGES?  
        BEQ      3$              ;; IF NOT: BR
```

```

(1) 017632 017600 000004      MOV      @4(SP),R0      ;;GET MESSAGE ADDR.
(1) 017636 062766 000002 000004  ADD      #2,4(SP)      ;;BUMP RETURN ADDR.
(1) 017644 005737 001170      TST      $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
(1) 017650 001375      BNE      1$           ;;IF NOT: WAIT
(1) 017652 010037 001204      MOV      R0,$MSGAD     ;;PUT ADDR IN MAILBOX
(1) 017656 105720      TSTB    (R0)+         ;;FIND END OF MESSAGE
(1) 017660 001376      BNE      2$           ;;
(1) 017662 163700 001204      SUB      $MSGAD,R0     ;;SUB START OF MESSAGE
(1) 017666 006200      ASR      R0           ;;GET MESSAGE LNTH IN WORDS
(1) 017670 010037 001206      MOV      R0,$MSGGLT    ;;PUT LENGTH IN MAILBOX
(1) 017674 012737 000004 001170  MOV      #4,$MSGTYPE   ;;TELL APT TO TAKE MSG.
(1) 017702 000413      BR       5$           ;;
(1) 017704 017637 000004 017730 3$:  MOV      @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
(1) 017712 062766 000002 000004  ADD      #2,4(SP)     ;;BUMP RETURN ADDRESS
(3) 017720 013746 177776      MOV      177776,-(SP) ;;PUSH 177776 ON STACK
(1) 017724 004737 015640      JSR     PC,$TYPE      ;;CALL TYPE MACRO
(1) 017730 000000      4$:      .WORD      0
(1) 017732      5$:
(1) 017732 105737 020020      10$:    TSTB    $FFLG      ;;SHOULD REPORT FATAL ERROR?
(1) 017736 001416      BEQ     12$          ;;IF NOT: BR
(1) 017740 005737 001210      TST     $ENV         ;;RUNNING UNDER API?
(1) 017744 001413      BEQ     12$          ;;IF NOT: BR
(1) 017746 005737 001170      11$:    TST     $MSGTYPE    ;;FINISHED LAST MESSAGE?
(1) 017752 001375      BNE     11$          ;;IF NOT: WAIT
(1) 017754 017637 000004 001172  MOV      @4(SP),$FATAL ;;GET ERROR #
(1) 017762 062766 000002 000004  ADD      #2,4(SP)     ;;BUMP RETURN ADDR.
(1) 017770 005237 001170      INC     $MSGTYPE     ;;TELL APT TO TAKE ERROR
(1) 017774 105037 020020      12$:    CLRB    $FFLG      ;;CLEAR FATAL FLAG
(1) 020000 105037 020017      CLRB    $LFLG       ;;CLEAR LOG FLAG
(1) 020004 105037 020016      CLRB    $MFLG       ;;CLEAR MESSAGE FLAG
(3) 020010 012601      MOV     (SP)+,R1     ;;POP STACK INTO R1
(3) 020012 012600      MOV     (SP)+,R0     ;;POP STACK INTO R0
(1) 020014 000207      RTS     PC           ;;RETURN
(1) 020016      000      $MFLG: .BYTE      0   ;;MESSG. FLAG
(1) 020017      000      $LFLG: .BYTE      0   ;;LOG FLAG
(1) 020020      000      $FFLG: .BYTE      0   ;;FATAL FLAG
(1)      020022      .EVEN
(1)      000200      APTSIZE=200
(1)      000001      APTENV=001
(1)      000100      APTSPool=100
(1)      000040      APTCSUP=040

```

```

1066          .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) 020022 010046          $TRAP: MOV      R0,-(SP)          ;;SAVE R0
(1) 020024 016600 000002  MOV      2(SP),R0          ;;GET TRAP ADDRESS
(1) 020030 005740          TST      -(R0)             ;;BACKUP BY 2
(1) 020032 111000          MOVNB   (R0),R0           ;;GET RIGHT BYTE OF TRAP
(1) 020034 006300          ASL     R0                ;;POSITION FOR INDEXING
(1) 020036 016000 020056  MOV      $TRPAD(R0),R0    ;;INDEX TO TABLE
(1) 020042 000200          RTS      R0              ;;GO TO ROUTINE
(1)
(1)          ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
(1)
(1) 020044 011646          $TRAP2: MOV     (SP),-(SP)  ;;MOVE THE PC DOWN
(1) 020046 016666 000004 000002  MOV     4(SP),2(SP)      ;;MOVE THE PSW DOWN
(1) 020054 000002          RTI                     ;;RESTORE THE PSW
(3)
(3)          .SBTTL TRAP TABLE
(3)          ;;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)          ;;*BY THE 'TRAP' INSTRUCTION.
(3)
(3)          : ROUTINE
(3)          : -----
(3) $TRPAD: .WORD $TRAP2
(3) 020056 020044          $TYPE    ;;CALL=TYPE     TRAP+1(104401) TTY TYPEOUT ROUTINE
(3) 020060 015640          $TYPOC   ;;CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 020062 015436          $TYPOS   ;;CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 020064 015412          $TYPON   ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 020066 015452          $TYPDS   ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(3) 020070 014054          $TYPBN   ;;CALL=TYPBN     TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
(1)
(3) 020074 016546          $GTSWR   ;;CALL=GTSWR   TRAP+7(104407) GET SOFT-SWR SETTING
(1)
(3) 020076 016456          $CKSWR   ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
(3) 020100 017020          $RDCHR   ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
(3) 020102 017110          $RDLIN   ;;CALL=RDLIN     TRAP+12(104412) TTY TYPEIN STRING ROUTINE
(3) 020104 017452          $RDOCT   ;;CALL=RDOCT     TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
1067
1068          000001          .END

```



ABASE = 171060	20#	34	110	111	112	113	924
ACDW1 = 000000	34						
ACDW2 = 000000	34						
ACPUOP= 000000	34						
ADBR 001464	118#	812					
ADCS 001462	117#	806*	809*	810			
ADDW0 = 000000	34						
ADDW1 = 000000	34						
ADDW10= 000000	34						
ADDW11= 000000	34						
ADDW12= 000000	34						
ADDW13= 000000	34						
ADDW14= 000000	34						
ADDW15= 000000	34						
ADDW2 = 000000	34						
ADDW3 = 000000	34						
ADDW4 = 000000	34						
ADDW5 = 000000	34						
ADDW6 = 000000	34						
ADDW7 = 000000	34						
ADDW8 = 000000	34						
ADDW9 = 000000	34						
ADEVCT= 000000	34						
ADEVN = 000000	34						
ADJR18 012305	606	676	976#				
ADJR19 012266	606	648	975#				
ADJR20 012343	608	978#					
ADJR21 012324	608	977#					
ADJR41 012401	610	980#					
ADJR42 012362	610	979#					
ADJR43 012437	612	982#					
ADJR44 012420	612	981#					
AENV = 000000	34						
AENVN = 000000	34						
AFATAL= 000000	34						
AMADR1= 000000	34						
AMADR2= 000000	34						
AMADR3= 000000	34						
AMADR4= 000000	34						
AMAMS1= 000000	34						
AMAMS2= 000000	34						
AMAMS3= 000000	34						
AMAMS4= 000000	34						
AMSGAD= 000000	34						
AMSGLG= 000000	34						
AMSGTY= 000000	34						
AMTYP1= 000000	34						
AMTYP2= 000000	34						
AMTYP3= 000000	34						
AMTYP4= 000000	34						
APASS = 000000	34						
APRIOR= 000000	34						
APTCSU 000040	1059	1063#					
APTENV= 000001	1050	1059	1063#				





IOTVEC- 000020	18#	157*													
LAMPS 007630	221	786#													
LDSPAC 011601	827	960#													
LDTRAP 002044	171#	181	202	245	291	787	858	895	907						
LF = 000012	18#	1059													
LOGIC 002600	193	218	249#	628											
MASKNM 010406	250*	292*	519*	622	926#	1050									
MESGD 012031	624	964#													
MSGSW 011647	187	961#													
MTEST 002310	192	196#	215												
MTEST1 002314	183	197#	231	233	246	884	1060								
N51200 013003	645	998#													
OFFDAC 006642	606	608	610	612	632#										
PC -%000007	18#	167*	179*	181*	189*	202*	245*	291*	593*	602*	614*	627*	649*		
	654*	677*	682*	705*	725*	731*	747*	753*	770*	776*	782*	787*	798*		
	830*	846*	849*	858*	860*	862*	864*	866*	867*	873*	888*	895*	896*		
	907*	908*	910*	912*	922*	1037*	1050*	1051*	1052	1059*	1060*	1063*			
PIRQ = 177772	18#														
PIRQVE- 000240	18#														
PRIMBA 013606	237	1020#													
PRIMBB 013643	240	1022#													
PRIMEO 013137	196	1002#													
PRO = 000000	18#														
PR1 = 000040	18#														
PR2 = 000100	18#														
PR3 = 000140	18#														
PR4 = 000200	18#														
PR5 = 000240	18#														
PR6 = 000300	18#														
PR7 = 000340	18#														
PS = 177776	18#														
PSW = 177776	18#														
PWRMSG 015344	1052	1053#													
PWRVEC= 000024	18#	157*	1052*												
P51175 013061	673	1000#													
RDCHR 104411	1060	1066#													
RDLIN = 104412	204	828	1062	1066#											
RDOCT 104413	241	1066#													
REMAIN 005334	320	468	509#												
RESTRT 001466	25	148#													
RESVEC= 000010	18#														
RSTRT 001516	149	155#													
RUNIT 002534	205*	206*	207	210	213	216	219	222	225	228	234#				
RO %000000	18#	171*	172	173	174	175	242*	244	614*	789*	794*	834	836*		
	838*	840*	841	844*	848*	859*	861*	863*	865*	869*	870*	871	897*		
	898	899	900	901	909*	911*	915	916	917	918	919*	920*	1035*		
	1051*	1052*	1059*	1062*	1063*	1066*									
R1 =%000001	18#	790*	792*	835	837*	838	843*	847*	1034*	1035*	1051*	1052*	1062*		
	1063*														
R2 %000002	18#	788*	791	796*	1035*	1052*	1062*								
R3 =%000003	18#	1035*	1052*	1056*	1060*										
R4 =%000004	18#	1052*	1056*												
R5 %000005	18#	590*	599*	606*	608*	610*	612*	632	635	636	637	651*	660*		
	665	668	669	679*	688*	693	696	701*	711*	715	718	719	728*		



TRYAGN	012735	658	686	997#
TSTR2	001456	115#	302#	
TST1	003034	299#	522	
TST10	003514	350#		
TST11	003550	355	357#	
TST12	003606	362	365#	
TST13	003660	374#		
TST14	003752	375#		
TST15	004046	377#		
TST16	004102	382	385#	
TST17	004140	390	393#	
TST2	003104	301	307	310#
TST20	004212	403#		
TST21	004304	405#		
TST22	004400	407#		
TST23	004434	412	415#	
TST24	004472	420	423#	
TST25	004544	434#		
TST26	004636	435#		
TST27	004732	437#		
TST3	003162	322#		
TST30	005114	463	466#	
TST31	005166	475	478#	
TST32	005232	485	489#	
TST33	005270	495	498#	
TST34	005334	505	510#	
TST35	005434	513	557#	
TST36	005456	559	563#	
TST37	005640	588#		
TST4	003216	327	330#	
TST40	005702	594	597#	
TST41	005744	603	606#	
TST42	005762	606#		
TST43	005776	606#		
TST44	006016	606#		
TST45	006032	606#		
TST46	006046	606#		
TST47	006062	608#		
TST5	003254	335	338#	
TST50	006100	608#		
TST51	006114	608#		
TST52	006134	608#		
TST53	006150	608#		
TST54	006164	608#		
TST55	006200	610#		
TST56	006216	610#		
TST57	006232	610#		
TST6	003326	348#		
TST60	006252	610#		
TST61	006266	610#		
TST62	006302	610#		
TST63	006316	612#		
TST64	006334	612#		
TST65	006350	612#		











CVMND-A MNCAA CVMNDA.P11		DIAGNOSTIC CROSS REFERENCE TABLE										MACY11 27(654) 19-SEP-78 08:58 N 6 PAGE 31-11				SEQ 0078
ADJR	972#	975	976	977	978	979	980	981	982							
COMEN	18#															
DYNIC	134#	348	374	403	434											
ENDCOM	18#															
ERROR	18#	268	288	308	318	328	336	344	348	349	356	363	371	374	375	
	383	391	399	403	405	413	421	429	434	435	446	452	458	464	476	
	486	496	506	573	581	595	604	656	684	707	733	755	778			
ESCAPE	18#															
FIXUNT	1038#	1050														
GETPRI	18#															
GETSWR	18#	190#														
MULT	18#															
NEWTST	18#	299	310	322	330	338	348	349	350	357	365	374	375	377	385	
	393	403	405	407	415	423	434	435	437	466	478	489	498	510	557	
	563	588	597	606	608	610	612									
POP	18#	1035	1052	1062	1063											
PUSH	18#	1035	1052	1062	1063											
REPORT	18#															
SCOPE	18#	299	310	322	330	338	348	349	350	357	365	374	375	377	385	
	393	403	405	407	415	423	434	435	437	466	478	489	498	510	557	
	563	588	597	606	608	610	612	614								
SELD	990#	993	994	995	996											
SETPRI	18#	1060														
SETTRA	1066#															
SETUP	18#	157														
SKIP	18#	262	285	287	301	307	316	327	335	343	348	349	355	362	370	
	374	375	382	390	398	403	405	412	420	428	434	435	445	451	457	
	463	475	485	495	505	513	559	572	580	594	603	655	683	706	710	
SLASH	18#															
SPACE	18#															
STARS	18#	31	33	34	299	310	322	330	338	348	349	350	357	365	374	
	375	377	385	393	403	405	407	415	423	434	435	437	466	478	489	
	498	510	557	563	588	597	606	608	610	612	614	1034	1035	1037	1050	
	1051	1052	1056	1059	1060	1062	1063	1066								
SUBTST	120#	349	375	405	435											
SUPER	525#	606	608	610	612											
SWRSU	18#	157#														
TRMTRP	1066#															
TYPBIN	18#	625														
TYPDEC	18#	614	1051													
TYPNAM	18#	190														
TYPNUM	18#															
TYPOCS	18#															
TYPOCT	18#	1051	1060													
TYPTXT	18#															
SSCMRE	34#															
SSCMTM	34#															
SSESCA	18#															
SSNEWT	18#	299	310	322	330	338	348	349	350	357	365	374	375	377	385	
	393	403	405	407	415	423	434	435	437	466	478	489	498	510	557	
	563	588	597	606	608	610	612									
SSSET	1066#															
SSSETM	157#															
SSSKIP	18#	301	307	327	335	355	362	382	390	412	420	463	475	485	495	

	505	513	559	594	603
.EQUAT	11#	18			
.HEADE	11#	17			
.SETUP	12#	156			
.SWRHI	13#	23			
.SWRLO	23#				
.SACT1	15#	31			
.SAPT8	15#	34#			
.SAPTH	15#	33			
.SAPTY	15#	1063			
.SCATC	11#	24			
.SCMTA	11#	34			
.SEOP	11#	614			
.SERRO	11#	1050			
.SERR1	13#	1051			
.SPARM	12#				
.SPOWE	12#	1052			
.SRDOC	14#	1062			
.SREAD	12#	1060			
.SSAVE	12#				
.SSCOP	12#	1037			
.SSPAC	12#				
.SSWDO	12#				
.STRAP	12#	1066			
.STYPB	13#	1034			
.STYPD	13#	1035			
.STYPE	11#	12#	1059		
.STYPO	11#	1056			

CVMND-A MNCAA CVMNDA.P11		DIAGNOSTIC CROSS REFERENCE TABLE													
		MACY11 27(654)				19-SEP-78 08:58				C 7 PAGE 31-13				SEQ 0080	
ADCB	1034														
ADD	176	177	178	254	515	516	517	518	812	845	870	1035	1051	1056	1059
	1060	1062	1063												
ASL	519	1051	1060	1062	1066										
ASLB	1035														
ASR	345	348	372	374	400	403	430	434	584	585	796	813	817	818	1050
	1063														
BCC	1035														
BEQ	157	185	190	192	243	287	301	307	327	335	343	348	349	355	362
	370	374	375	382	390	398	403	405	412	420	428	434	435	445	451
	457	463	475	485	495	505	513	572	580	614	618	623	1034	1037	1050
	1051	1056	1059	1060	1062	1063									
BGE	1037														
BGT	614	1035	1056	1060											
BHI	1037														
BIC	305	569	575	578	614	805	880	1056	1060	1062					
BICB	206														
BIS	198	284	472	482	493	503	616	1035	1050	1056	1060				
BISB	1051														
BIT	258	1037	1050												
BITB	157	1059	1063												
BLE	842														
BLO	1060														
BLOS	1060														
BLT	1035	1056	1059	1060											
BMI	257	271	1035												
BNE	157	183	190	208	211	214	217	220	223	226	229	259	261	265	267
	273	281	346	348	349	373	374	375	401	403	405	431	434	435	468
	559	586	639	671	710	721	743	766	793	795	797	814	872	882	886
	1035	1037	1050	1051	1052	1056	1059	1060	1063						
BPL	811	839	878	921	1035	1050	1056	1059	1060						
BR	149	151	157	190	231	262	285	316	594	603	655	659	683	687	706
	732	754	777	799	868	902	913	1034	1035	1037	1050	1051	1052	1056	1059
	1060	1062	1063												
CCC	582	815													
CLC	1034														
CLR	152	153	154	157	158	168	199	200	201	250	251	289	290	302	323
	351	378	408	470	480	491	500	520	521	614	790	807	869	911	1035
	1037	1051	1052	1056	1060	1062									
CLRB	1035	1037	1059	1060	1063										
CMP	157	190	260	263	306	317	326	334	342	348	349	354	361	369	374
	375	381	389	397	403	405	411	419	427	434	435	444	450	456	462
	474	484	571	579	622	841	881	885	1035	1037	1050	1060			
CMPB	190	207	210	213	216	219	222	225	228	286	512	1037	1050	1059	1060
	1063														
COM	577														
DEC	614	792	794	920	1051	1060									
DECB	1056	1059													
EMT	18														
HALT	24	1050	1052	1059											
INC	148	150	190	255	511	514	614	1035	1037	1050	1052	1056	1060	1063	
INCB	809	1037	1050	1059											
IOT	18														
JMP	24	25	26	169	193	209	212	215	218	221	224	227	233	246	269

JSR	282	320	560	614	628	884	1060										
	167	181	189	202	245	291	590	593	599	602	606	608	610	612	614	614	614
	627	649	651	654	677	679	682	701	705	725	728	731	747	750	753	753	753
	770	773	776	782	787	798	858	860	862	864	866	867	895	896	907	907	907
MOV	908	910	912	1037	1050	1059	1060	1063									
	157	159	163	164	165	166	171	172	173	174	175	205	238	242	244	244	244
	249	252	276	283	292	303	304	311	319	321	324	325	331	332	333	333	333
	338	339	340	341	348	349	352	353	358	359	360	365	366	367	368	368	368
	374	375	379	380	386	387	388	393	394	395	396	403	405	409	410	410	410
	416	417	418	423	424	425	426	434	435	438	439	440	441	442	443	443	443
	448	449	454	455	460	461	469	473	479	483	490	494	499	501	504	504	504
	510	522	557	563	564	565	566	567	568	570	574	576	588	589	592	592	592
	597	598	601	606	608	610	612	614	620	625	632	633	634	635	636	636	636
	637	646	650	653	665	666	667	668	669	674	678	681	693	694	695	695	695
	696	698	699	704	715	716	717	718	719	726	727	730	737	738	739	739	739
	740	741	748	749	752	759	760	761	762	763	764	771	772	775	786	786	786
	788	789	791	803	806	808	819	834	835	836	837	843	844	847	848	848	848
	857	859	861	863	865	879	894	897	898	899	900	901	906	909	915	915	915
	916	917	918	919	1034	1035	1037	1050	1051	1052	1056	1059	1060	1062	1063	1063	1063
MOV B	1066																
	157	190	1034	1035	1037	1050	1056	1059	1060	1062	1063	1066					
NEG	840	1035	1056														
NOP	614	1050															
RESET	155	197	471	481	492	502	614										
ROL	1034	1056	1062														
ROR	583	816															
RTI	157	160	166	523	1034	1035	1037	1050	1052	1056	1059	1060	1062	1066	1051		
RTS	179	660	688	711	734	756	783	820	830	846	849	873	888	922	1051		
	1059	1060	1063	1066													
SEC	1034																
SUB	349	375	405	435	708	709	838	1035	1050	1063							
SWAB	804																
TRAP	1066																
TST	182	184	190	191	253	256	264	266	270	272	280	300	312	313	314	314	314
	315	349	375	405	435	467	558	617	638	670	720	742	765	829	871	871	871
	883	1035	1037	1050	1051	1056	1059	1060	1062	1063	1066						
TST B	810	877	1035	1037	1051	1059	1060	1063									
.ASCII	34	1003	1005	1007	1009	1011	1013	1015									
.ASCII Z	34	190	614	945	946	947	948	949	950	951	952	953	954	955	956	956	956
	957	958	959	960	961	962	963	964	966	967	968	969	970	971	975	975	975
	976	977	978	979	980	981	982	984	985	986	987	988	993	994	995	995	995
	996	997	999	1001	1017	1019	1021	1022	1051	1053	1060						
.BLKB	1060																
.BLKW	1035																
.BYTE	34	614	965	983	998	1000	1002	1004	1006	1008	1010	1012	1014	1016	1018	1018	1018
	1020	1031	1034	1050	1056	1060	1063										
.DSABL	1060																
.ENABL	8	1060															
.END	1068																
.ENDC	17	18	23	24	31	33	34	156	157	190	262	285	287	299	301	301	301
	307	310	316	322	327	330	335	338	343	348	349	350	355	357	362	362	362
	365	370	374	375	377	382	385	390	393	398	403	405	407	412	415	415	415
	420	423	428	434	435	437	445	451	457	463	466	475	478	485	489	489	489
	495	498	505	510	513	557	559	563	572	580	588	594	597	603	606	606	606

CVMND-A MNCAA CVMNDA.P11		DIAGNOSTIC CROSS REFERENCE TABLE														MACY11 27(654)	19-SEP-78	08:58	E 7	PAGE 31-15	SEQ 0082
	608	610	612	614	655	683	706	710	1034	1035	1037	1050	1051	1052	1056						
	1059	1060	1062	1063	1066																
.EQUIV	18																				
.EVEN	34	190	1023	1051	1054	1060	1063														
.IF	17	18	23	24	31	33	34	156	157	190	262	285	287	299	301						
	307	310	316	322	327	330	335	338	343	348	349	350	355	357	362						
	365	370	374	375	377	382	385	390	393	398	403	405	407	412	415						
	420	423	428	434	435	437	445	451	457	463	466	475	478	485	489						
	495	498	505	510	513	557	559	563	572	580	588	594	597	603	606						
	608	610	612	614	655	683	706	710	1034	1035	1037	1050	1051	1052	1056						
	1059	1060	1062	1063	1066																
.IFF	18	23	31	33	34	157	190	262	285	287	299	301	307	310	316						
	322	327	330	335	338	343	348	349	350	355	357	362	365	370	374						
	375	377	382	385	390	393	398	403	405	407	412	415	420	423	428						
	434	435	437	445	451	457	463	466	475	478	485	489	495	498	505						
	510	513	557	559	563	572	580	588	594	597	603	606	608	610	612						
	614	655	683	706	710	1034	1035	1037	1050	1051	1052	1056	1059	1060	1062						
	1063	1066																			
.IFT	190	1037	1050	1060	1062																
.IFTF	170	1037	1050	1060	1062																
.IIF	17	23	24	34	157	190	614	625	1037	1050	1051	1059	1060	1066							
.IRP	156	299	310	322	330	338	348	349	350	357	365	374	375	377	385						
	393	403	405	407	415	423	434	435	437	466	478	489	498	510	557						
	563	588	597	606	608	610	612	1035	1037	1050	1052	1062	1063	1063							
.LIST	6	16	18	23	24	34	156	157	190	298	299	310	322	330	338						
	348	349	350	357	365	374	375	377	385	393	403	405	407	415	423						
	434	435	437	466	478	489	498	510	557	563	588	597	606	608	610						
	612	614	854	939	1037	1050	1060	1066													
.MACRO	23	34	120	134	157	525	972	990	1038	1066											
.MCALL	11	12	13	14	15	18	34	157	190												
.MEXIT	34																				
.NLIST	1	7	18	23	24	34	156	157	190	293	299	310	322	330	338						
	348	349	350	357	365	374	375	377	385	393	403	405	407	415	423						
	434	435	437	466	478	489	498	510	557	563	588	597	606	608	610						
	612	614	851	937	1037	1050	1060	1066													
.PAGE	34																				
.REPT	24																				
.SBTTL	18	23	24	31	33	34	157	190	236	248	294	295	296	297	299						
	310	322	330	338	348	349	350	357	365	374	375	377	385	393	403						
	405	407	415	423	434	435	437	466	478	489	498	510	557	563	588						
	597	606	608	610	612	614	630	663	691	714	736	758	785	801	825						
	832	852	853	855	875	892	904	938	943	1034	1035	1037	1050	1051	1052						
	1056	1059	1060	1062	1063	1066															
.TITLE	17																				
.WORD	24	31	33	34	614	1051	1052	1056	1059	1060	1062	1063	1066								

ERRORS DETECTED: 0

CVMND-A MNCAA DIAGNOSTIC  
CVMNDA.P11

MACY11 27(654) 19-SEP-78 08:58 F 7 PAGE 31-16

SEQ 0083

\*CVMNDA,CVMNDA/CRF CVMNDA  
RUN-TIME: 31 14 2 SECONDS  
CORE USED: 26K