

MINC-11

SIZER PROGRAM
CVMNFA0

AH-B101A-MC
COPYRIGHT © 1978
FICHE 1 OF 1

DEC 1978
digital
MADE IN USA

The microfiche card contains 40 frames of data arranged in a 10x4 grid. The frames contain various types of data, including text, tables, and bar charts. The text is small and difficult to read, but appears to be technical data related to the SIZER PROGRAM. The bar charts show varying levels of activity or data points across different categories. The overall appearance is that of a standard microfiche card used for data storage and retrieval.

IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-B100A-MC
PRODUCT NAME: CVMNFA0 MINC-11 SIZER PROG
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	CONTROL
6.0	ERROR REPORTING
7.0	MISCELLANEOUS
7.1	MINC-11 BUS ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	MULTIPLE MINC-11 OPTION INTERFACE
7.5	RESTRICTIONS
8.0	EXECUTION TIME
9.0	EXAMPLE REPORT
10.0	LISTING

1.0 ABSTRACT

THE PURPOSE OF THE MINC-11 SIZER PROGRAM IS TO VERIFY THE CONFIGURATION OF THE MINC-11 HARDWARE. THE PROGRAM IS TO BE USED TO VERIFY THE CORRECT POSITION OF ANALOG CHANNELS, DEVICE ADDRESS AND VECTOR SWITCHES. THE PROGRAM SIZES THE NUMBER OF MINC-11 OPTIONS AND REPORTS THEIR EXISTANCE, BUS AND VECTOR ADDRESSES. TWO ADDITIONAL REPORTS ARE MADE IF REQUESTED. THE FIRST IS THE MODE OF THE MNCAD (A/D) CHANNELS. THE SECOND IS THE ADDRESS USAGE MAP WHICH IS USEFUL IN DETECTING A INCORRECT ADDRESS SWITCH SETTING. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL VIA THE PROVISIONS OF SECTION 5.

2.0 REQUIREMENTS

2.1 EQUIPMENT

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

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 VMNF??' 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 IF THE OPERATOR WANTS THE REPORT OF THE MODES OF THE MNCAD (A/D) CHANNELS.
 3. THE PROGRAM WILL NOW ASK IF THE OPERATOR WANTS THE REPORT OF THE ADDRESS USAGE MAP.
 4. THE PROGRAM WILL THEN SIZE THE MINC-11 SYSTEM AND REPORT THE DEVICE BUS AND VECTOR ADDRESSES AND SELECTED REPORTS.

4.1 PROGRAM START

200

STARTING ADDRESS OF THE PROGRAM

5.0 SOFTWARE SWITCH REGISTER

NONE

5.1 PROGRAM CONTROL

THE SIZING MAYBE STOPPED BY TYPING THE 'CONTROL & C'
KEYS. THIS OPERATION WILL STOP THE PROGRAM AND ENABLE
THE OPERATOR TO SELECT DIFFERENT PROGRAM REPORT'S.

6.0 ERROR REPORTING

NONE, NO ERROR MESSAGES ARE REPORTED.

7.0 MISCELLANEOUS

7.1 MINC-11 BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' (LOC. 1240) IF BASE BUS ADDRESS IS NOT 171000.

7.2 XXDP/APT NOTES

THIS PROGRAM IS CHAINABLE UNDER XYDP (REQUIRES 8K OR MORE).
WHEN RUN IN 'CHAIN' MODE, THE PROGRAM WILL REQUIRE THE OPERATOR
TO PERFORM INITIAL SETUP OPERATIONS. THE SIZER PROGRAM SHOULD
NOT BE INCLUDED IN A 'CHAIN' THAT LOOPS TO THE STARTING FILE.
THIS PROGRAM 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 MINC-11 OPTION INTERFACE TESTING

THIS PROGRAM DOES 'AUTO-SIZE' THE NUMBER OF MINC-11 OPTIONS CONNECTED.
THE PROGRAM WILL SIZE SEQUENTIALLY UP TO:

4	MNCAD (A/D)'S
8	MNCKW (CLOCK)'S
8	MNCAA (D/A)'S
8	MNCDO (DIGITAL OUT)'S
8	MNCDI (DIGITAL IN)'S

EACH MINC-11 OPTION MUST BE CONFIGURED WITH CONTIGUOUS BUS ADDRESSES.

7.5 RESTRICTIONS

ALL USER CONNECTIONS MUST BE REMOVED.

8.0 EXECUTION TIME

DEPENDANT UPON TERMINAL SPEED AND REPORTS SELECTED.

9.0 EXAMPLE REPORT

A 'TYPICAL' SIZER REPORT FOR A SYSTEM WITH THE FOLLOWING:

171000	MNCAD	171020	MNCKW
171060	MNCAA	171160	MNCDI
171260	MNCDO	171420	IBV-11
173000-173776	BOOT-STRAP PROM		
176500-176526	SLU 0, 1, AND 2		
177170	RX01 DISK SUB-SYSTEM		
177560-177566	CONSOLE TERMINAL		

DO YOU WANT THE MNCAD (A/D) CHANNEL MODE REPORT ?

TYPE 'Y' FOR YES = Y

DO YOU WANT THE MEMORY USAGE MAP REPORT ?

TYPE 'Y' FOR YES = Y

```

;MNCAD AT ADDRESS = 171000    VECTOR = 400
  0 - 7 SINGLE ENDED
 10 - 13 DIFFERENTIAL
 14 - 17 DIFFERENTIAL
 20 - 23 DIFFERENTIAL

```

```

;MNCKW AT ADDRESS = 171020    VECTOR = 400
;MNCAA AT ADDRESS = 171060    VECTOR = ** DOES NOT EXIST **
;MNCDI AT ADDRESS = 171160    VECTOR = 130
;MNCDO AT ADDRESS = 171260    VECTOR = 340

```

OF ;MNCAD 1 ;MNCKW 1 ;MNCAA 1 ;MNCDI 1 ;MNCDO 1

MNC11 MEMORY USAGE MAP (EACH BIT = 2 ADDRESSES)

ADDR	000/400	100/500	200/600	300/700
170000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
170400	0000000000000000	0000000000000000	0000000000000000	0000000000000000
171000	10001000000001100	00000000000001100	00000000000001000	00000000000000000
171400	00001100000000000	00000000000000000	00000000000000000	00000000000000000
172000	00000000000000000	00000000000000000	00000000000000000	00000000000000000
172400	00000000000000000	00000000000000000	00000000000000000	00000000000000000
173000	11111111111111111	11111111111111111	11111111111111111	11111111111111111
173400	11111111111111111	11111111111111111	11111111111111111	11111111111111111
174000	00000000000000000	00000000000000000	00000000000000000	00000000000000000
174400	00000000000000000	00000000000000000	00000000000000000	00000000000000000
175000	00000000000000000	00000000000000000	00000000000000000	00000000000000000
175400	00000000000000000	00000000000000000	00000000000000000	00000000000000000
176000	00000000000000000	00000000000000000	00000000000000000	00000000000000000
176400	00000000000000000	11111100000000000	00000000000000000	00000000000000000
177000	00000000000000000	00000000000000010	00000000000000000	00000000000000000
177400	00000000000000000	00000000000001100	00000000000000000	00000000000000000

END PASS # 1

10. LISTING

ATTACHED

13	BASIC DEFINITIONS
18	TRAP CATCHER
39	ACT11 HOOKS
41	APT PARAMETER BLOCK
42	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
79	INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE
81	INITIALIZE THE COMMON TAGS
102	TYPE PROGRAM NAME
254	END OF PASS ROUTINE
519	TTY INPUT ROUTINE
520	TYPE ROUTINE
522	BINARY TO OCTAL (ASCII) AND TYPE
523	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
524	BINARY TO ASCII AND TYPE ROUTINE
525	APT COMMUNICATIONS ROUTINE
527	TRAP DECODER
(3)	TRAP TABLE

```

12      .TITLE MAINDEC-11-DVMNF-A      MINC-11 OPTION SIZER PROGRAM
(1)    .*COPYRIGHT (C) 1978
(1)    .*DIGITAL EQUIPMENT CORP.
(1)    .*MAYNARD, MASS. 01754
(1)    .*
(1)    .*PROGRAM BY R SHOOP
(1)    .*
(1)    .*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)    .*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)    .*
13      .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  ;;PROFESSOR 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
14 171000 ABASE= 171000
15 000100 .=100
16 000100 000104 000200 000002 .WORD 104,200,2 ;B EVENT VECTOR
17
18 .SBTTL TRAP CATCHER
19
20 000000 .=0
21 ;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A '+2'
22 ;*AND 'JSR PC,RO' SEQUENCE TO CATCH ILLEGAL INTERRUPTS.
23 ;*AND INTERRUPTS TO THE WRONG VECTOR.
24 ;*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
25 ;*VECTORS.
35
36 000200 .=200
37 000200 000137 001324 JMP BEGIN ;START ADDRESS
```

```
39          .SBTTL ACT11 HOOKS
(1)
(2)          ;*****
(1)          ;HOOKS REQUIRED BY ACT11
(1)          000204          $SVPC=.          ;SAVE PC
(1)          000046          .=46
(1) 000046 002644          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)          000052          .=52
(1) 000052 000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)          000204          .= $SVPC          ;; RESTORE PC
(1)          40          001000
(1)          41          .SBTTL APT PARAMETER BLOCK
(2)          ;*****
(1)          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)          ;*****
(1)          001000          .$X=.          ;;SAVE CURRENT LOCATION
(1)          000024          -.24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200          200          ;;FOR APT START UP
(1)          000044          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000          $APTHDR ;;POINT TO APT HEADER BLOCK
(1)          001000          .=.$X          ;;RESET LOCATION COUNTER
(2)          ;*****
(1)          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)          ;INTERFACE SPEC.
(1)          001000          $APTHD:
(1) 001000 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001164          $MBADR: .WORD $MAIL          ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 002260          $TSTM: .WORD 1200.          ;;RUN TIM OF LONGEST TEST
(1) 001006 000764          $PASTM: .WORD 500.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 003244          $UNITM: .WORD 1700.          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000031          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

42

(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)

(2)

(2)

(2)

(3)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

.SBTTL COMMON TAGS

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

.=1100

\$CMTAG: .WORD 0 ;:START OF COMMON TAGS

\$STNM: .BYTE 0 ;:CONTAINS THE TEST NUMBER

\$ERFLG: .BYTE 0 ;:CONTAINS ERROR FLAG

\$ICNT: .WORD 0 ;:CONTAINS SUBTEST ITERATION COUNT

\$LPADR: .WORD 0 ;:CONTAINS SCOPE LOOP ADDRESS

\$LPERR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS

\$ERTTL: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED

\$ITEMB: .BYTE 0 ;:CONTAINS ITEM CONTROL BYTE

\$ERMAX: .BYTE 1 ;:CONTAINS MAX. ERRORS PER TEST

\$ERRPC: .WORD 0 ;:CONTAINS PC OF LAST ERROR INSTRUCTION

\$GDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'GOOD' DATA

\$BDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'BAD' DATA

\$GDDAT: .WORD 0 ;:CONTAINS 'GOOD' DATA

\$BDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA

.WORD 0 ;:RESERVED--NOT TO BE USED

.WORD 0 ;:RESERVED--NOT TO BE USED

\$AUTOB: .BYTE 0 ;:AUTOMATIC MODE INDICATOR

\$INTAG: .BYTE 0 ;:INTERRUPT MODE INDICATOR

.WORD 0 ;:RESERVED--NOT TO BE USED

SWR: .WORD DSWR ;:ADDRESS OF SWITCH REGISTER

DISPLAY: .WORD DDISP ;:ADDRESS OF DISPLAY REGISTER

\$TKS: 177560 ;:TTY KBD STATUS

\$TKB: 177562 ;:TTY KBD BUFFER

\$TPS: 177564 ;:TTY PRINTER STATUS REG. ADDRESS

\$TPB: 177566 ;:TTY PRINTER BUFFER REG. ADDRESS

\$NULL: .BYTE 0 ;:CONTAINS NULL CHARACTER FOR FILLS

\$FILLS: .BYTE 2 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED

\$FILLC: .BYTE 12 ;:INSERT FILL CHARS. AFTER A 'LINE FEED'

\$TPFLG: .BYTE 0 ;: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)

\$QUES: .ASCII /?/ ;:QUESTION MARK

\$CRLF: .ASCII <15> ;:CARRIAGE RETURN

\$LF: .ASCIZ <12> ;:LINE FEED

::*****

.SBTTL APT MAILBOX-ETABLE

::*****

.EVEN ;:APT MAILBOX

\$MAIL: .WORD ;:MESSAGE TYPE CODE

\$MSGTY: .WORD AMSGTY ;:FATAL ERROR NUMBER

\$FATAL: .WORD AFATAL ;:TEST NUMBER

\$TESTN: .WORD ATESTN ;:PASS COUNT

\$PASS: .WORD APASS ;:DEVICE COUNT

\$DEVCT: .WORD ADEVCT ;:I/O UNIT NUMBER

\$UNIT: .WORD AUNIT ;:MESSAGE ADDRESS

\$MSGAD: .WORD AMSGAD ;:MESSAGE LENGTH

\$MSGLG: .WORD AMSGLG ;:MESSAGE LENGTH

(2)	001204		\$ETABLE:	::APT ENVIRONMENT TABLE
(2)	001204	000	\$ENV: .BYTE	AENV ::ENVIRONMENT BYTE
(2)	001205	000	\$ENVM: .BYTE	AENVM ::ENVIRONMENT MODE BITS
(2)	001206	000000	\$SWREG: .WORD	ASWREG ::APT SWITCH REGISTER
(2)	001210	000000	\$USWR: .WORD	AUSWR ::USER SWITCHES
(2)	001212	000000	\$CPUOP: .WORD	ACPUOP ::CPJ 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)	001214	000	\$MAMS1: .BYTE	AMAMS1 ::HIGH ADDRESS,M.S. BYTE
(2)	001215	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)	001216	000000	\$MADR1: .WORD	AMADR1 ::HIGH ADDRESS,BLK#1
(2)			.*	MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001220	000	\$MAMS2: .BYTE	AMAMS2 ::HIGH ADDRESS,M.S. BYTE
(2)	001221	000	\$MTYP2: .BYTE	AMTYP2 ::MEM. TYPE,BLK#2
(2)	001222	000000	\$MADR2: .WORD	AMADR2 ::MEM.LAST ADDRESS,BLK#2
(2)	001224	000	\$MAMS3: .BYTE	AMAMS3 ::HIGH ADDRESS,M.S.BYTE
(2)	001225	000	\$MTYP3: .BYTE	AMTYP3 ::MEM. TYPE,BLK#3
(2)	001226	000000	\$MADR3: .WORD	AMADR3 ::MEM.LAST ADDRESS,BLK#3
(2)	001230	000	\$MAMS4: .BYTE	AMAMS4 ::HIGH ADDRESS,M.S.BYTE
(2)	001231	000	\$MTYP4: .BYTE	AMTYP4 ::MEM. TYPE,BLK#4
(2)	001232	000000	\$MADR4: .WORD	AMADR4 ::MEM.LAST ADDRESS,BLK#4
(2)	001234	000000	\$VECT1: .WORD	AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001236	000000	\$VECT2: .WORD	AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001240	171000	\$BASE: .WORD	ABASE ::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001242	000000	\$DEVN: .WORD	ADEVN ::DEVICE MAP
(2)	001244	000000	\$CDW1: .WORD	ACDW1 ::CONTROLLER DESCRIPTION WORD#1
(2)	001246		\$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) $ERRTB:
(1) 001246 ;NO ERRORS ARE REPORTED
43
44
45
46 001246 000020 KWOFF: 20 ;OFFSET FROM $BASE VALUE TO KW ADDRESS
47 001250 000060 AAOFF: 60 ;OFFSET FROM $BASE VALUE TO AA ADDRESS
48 001252 000160 DIOFF: 160 ;OFFSET FROM $BASE VALUE TO DI ADDRESS
49 001254 000260 DOOFF: 260 ;OFFSET FROM $BASE VALUE TO DO ADDRESS
50
51 ;PROGRAM CREATES THESE ADDRESSES FROM THE VALUE OF $BASE PLUS THE OFFSET FOR THAT DEVICE
52
53 001256 171000 ADBASE: 171000
54 001260 171020 KWBASE: 171020
55 001262 171060 AABASE: 171060
56 001264 171160 DIBASE: 171160
57 001266 171260 DOBASE: 171260
58
59 ;COUNT OF EACH UNIT FOUND
60
61 001270 000000 ADCNT: 0
62 001272 000000 KWCNT: 0
63 001274 000000 AACNT: 0
64 001276 000000 DICNT: 0
65 001300 000000 DOCNT: 0
66
67 ;MISC. TEMP LOCATIONS
68 001302 000000 VADR: 0
69 001304 000000 VECT: 0
70 001306 000000 TEMP: 0
71 001310 000000 NBEXT: 0
72 001312 010114 OUTADR: ADRPOK ;ADDRESS VALUE FOR MEMORY USAGE
73 001314 000000 FLAGMP: 0 ;NON-ZERO INDICATES MEMORY USAGE MAP IS WANTED
74 001316 000000 FLAGAD: 0 ;NON-ZERO INDICATES THE A/D CHANNEL MODES BE REPORTED
75 001320 000000 SWREG: 0 ;HERE BECAUSE OF SYSMAC
76 001322 000000 DISPRE: 0 ;HERE BECAUSE OF SYSMAC

```

```

79          .SBTTL          INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
80 001324 000005 BEGIN: RESET
81          .SBTTL          INITIALIZE THE COMMON TAGS
(1)          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001326 012706 001100 MOV    #SCMTAG,R6          ;;FIRST LOCATION TO BE CLEARED
(1) 001332 005026 CLR    (R6)+              ;;CLEAR MEMORY LOCATION
(1) 001334 022706 001140 CMP    #SWR,R6 ;;DONE?
(1) 001340 001374 BNE    -6                ;;LOOP BACK IF NO
(1) 001342 012706 001100 MOV    #STACK,SP          ;;SETUP THE STACK POINTER
(1)          ;;INITIALIZE A FEW VECTORS
(1) 001346 012737 010036 000034 MOV    #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001354 012737 000340 000036 MOV    #340,@TRAPVEC+2;LEVEL 7
(1) 001362 013737 002612 002604 MOV    SENDCT,$EOPCT    ;;SETUP END-OF-PROGRAM COUNTER
(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) 001370 013746 000004 MOV    @ERRVEC,-(SP)    ;;SAVE ERROR VECTOR
(2) 001374 012737 001430 000004 MOV    #64$,@ERRVEC    ;;SET UP ERRVEC VECTOR
(2) 001402 012737 177570 001140 MOV    #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001410 012737 177570 001142 MOV    #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
(2) 001416 022777 177777 177514 CMP    #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
(2) 001424 001012 BNE    66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)          ;;AND THE HARDWARE SWR IS NOT - -1
(2) 001426 000403 BR     65$             ;;BRANCH IF NO TIMEOUT
(2) 001430 012716 001436 64$: MOV    #65$,(SP)          ;;SET UP FOR TRAP RETURN
(2) 001434 000002 RTI
(2) 001436 012737 001320 001140 65$: MOV    #SWREG,SWR        ;;POINT TO SOFTWARE SWR
(2) 001444 012737 001322 001142 MOV    #DISPREG,D.SPLAY
(2) 001452 012637 000004 66$: MOV    (SP)+,@ERRVEC    ;;RESTORE ERROR VECTOR
(1)
(2) 001456 005037 001172 CLR    $PASS           ;;CLEAR PASS COUNT
(2) 001462 132737 000200 001205 BITB  #APTSIZE,$ENVM    ;;TEST USER SIZE UNDER APT
(2) 001470 001403 BEQ    67$           ;;YES,USE NON-APT SWITCH
(2) 001472 012737 001206 001140 MOV    #SSWREG,SWR    ;;NO,USE APT SWITCH REGISTER
(2) 001500 67$:
82 001500 012737 005046 006560 MOV    #5046,$TYPE    ;
83 001506 012737 012746 006562 MOV    #12746,$TYPE+2 ;
84 001514 012737 006572 006564 MOV    #STYPE+12,$TYPE+4
85 001522 012737 000002 006566 MOV    #RTI,$TYPE+6
86 001530 004737 005656 JSR    PC,$TKINT      ;;INIT THE TKB
87 001534 013737 001240 001256 MOV    $BASE,ADBASE   ;;GET INITIAL BASE
88 001542 013737 001256 001260 MOV    ADBASE,KWBASE  ;;GET KW BASE
89 001550 063737 001246 001260 ADD    KWOFF,KWBASE  ;;UPDATE KW ADDRESS
90 001556 013737 001256 001262 MOV    ADBASE,AABASE  ;;GET AA BASE
91 001564 063737 001250 001262 ADD    AAOFF,AABASE  ;;UPDATE AA ADDRESS
92 001572 013737 001256 001264 MOV    ADBASE,DIBASE ;;GET DI BASE
93 001600 063737 001252 001264 ADD    DIOFF,DIBASE  ;;UPDATE DI ADDRESS
94 001606 013737 001256 001266 MOV    ADBASE,DOBASE ;;GET DO BASE
95 001614 063737 001254 001266 ADD    DOOFF,DOBASE  ;;UPDATE DO ADDRESS
96 001622 005037 001270 CLR    ADCNT          ;;CLEAR THE UNIT COUNTERS
97 001626 005037 001272 CLR    KWCNT
98 001632 005037 001274 CLR    AACNT
99 001636 005037 001276 CLR    DICNT
100 001642 005037 001300 CLR    DOCNT

```

```
102 .SBTTL TYPE PROGRAM NAME
(1) ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 001646 005227 177777 INC #-1 ;;FIRST TIME?
(1) 001652 001036 BNE 68$ ;;BRANCH IF NO
(1) 001654 022737 002644 000042 CMP #SENDAD,@#42 ;;ACT-11?
(1) 001662 001432 BEQ 68$ ;;BRANCH IF YES
(1) 001664 104401 001672 TYPE ,69$ ;;TYPE ASCIZ STRING
(1) 001670 000427 BR 68$ ;;GET OVER THE ASCIZ
(1) ;;69$: .ASCIZ <CRLF>#MD-11-DVMNF-A MINC-11 OPTION SIZER PROGRAM<CRLF>
68$:
103 001750 105737 001205 TSTB $ENVM ;;CHECK APT MODE?
104 001754 001042 BNE LOOP ;;BR IF YES
105 001756 005737 000042 TST CHAIN ;;CHECK XXDP/ACT MODE?
106 001762 001405 BEQ 1$ ;;BR IF NOT
107 001764 104401 004664 TYPE ,SETMNC ;;TELL OPER. ABOUT SETUP STUFF
108 001770 104407 RDCHR ;;GET A CHAR
109 001772 012600 MOV (SP)+,R0 ;;READ WHATEVER CHARACTER
110 001774 000432 BR LOOP ;;RUN TEST WHEN OPER IS READY
111 001776 104401 004262 1$: TYPE ,QUESAD ;;ASK OPERATOR ABOUT A/D MODE
112 002002 104407 RDCHR ;;GET OPER INPUT
113 002004 012600 MOV (SP)+,R0
114 002006 042700 177600 BIC #177600,R0 ;;MASK OFF JUNK
115 002012 005037 001316 CLR FLAGAD ;;INIT THE EXPANDED A/D REPORT
116 002016 122700 000131 CMPB #'Y,R0 ;;CHECK IF 'Y'
117 002022 001002 BNE 2$ ;;BR IF NOT
118 002024 005237 001316 INC FLAGAD ;;SET EXPANDED A/D REPORT
119 002030 104401 004373 2$: TYPE ,QUESMP ;;ASK OPERATOR ABOUT MEMORY USAGE MAP
120 002034 104407 RDCHR ;;GET OPER INPUT
121 002036 012600 MOV (SP)+,R0
122 002040 042700 177600 BIC #177600,R0 ;;MASK OFF JUNK
123 002044 005037 001314 CLR FLAGMP ;;INIT THE MEMORY USAGE REPORT
124 002050 122700 000131 CMPB #'Y,R0 ;;CHECK IF 'Y'
125 002054 001002 BNE LOOP ;;BR IF NOT
126 002056 005237 001314 INC FLAGMP ;;SET MEMORY USAGE FLAG
127
128 ;NOW VERIFY THE MNCAD MODULES
LOOP: JSR R5,TEST ;;CHECK THE MNCAD'S
129 002062 004537 002700 ADGO ;;ADDRESS OF INTR. STARTUP
130 002066 003456 ADBASE ;;STARTING A/D ADDRESS
131 002070 001256 4 ;;OFFSET TO NEXT UNIT
132 002072 000004 ADPRI ;;A/D MESSAGE POINTER
133 002074 004030 4 ;;MAX. # OF A/D'S ON A SYSTEM
134 002076 000004 ADCNT ;;# OF A/D'S FOUND ON SYSTEM
135 002100 001270
136 ;NOW VERIFY THE MNCKW MODULES
137 002102 004537 002700 JSR R5,TEST ;;CHECK THE MNCKW'S
138 002106 003474 KWGO ;;ADDRESS OF INTR. STARTUP
139 002110 001260 KWBASE ;;STARTING K/W ADDRESS
140 002112 000004 4 ;;OFFSET TO NEXT UNIT
141 002114 004042 KWPRI ;;KW MESSAGE POINTER
142 002116 000010 8. ;;MAX. # OF KW ON A SYSTEM
143 002120 001272 KWCNT ;;# OF KW FOUND ON SYSTEM
```



```

145      ;NOW VERIFY THE MNCAA MODULES
146 002122 004537 002700      JSR      R5,TEST      ;CHECK THE MNCAA
147 002126 000000              0      ;NO INTR. STARTUP
148 002130 001262              AABASE ;STARTING AA ADDRESS
149 002132 000010              10     ;OFFSET TO NEXT UNIT
150 002134 004054              AAPRI  ;AA MESSAGE POINTER
151 002136 000010              8.     ;MAX. # OF AA ON A SYSTEM
152 002140 001274              AACNT  ;# OF AA FOUND ON SYSTEM
153      ;NOW VERIFY THE MNCDI MODULES
154 002142 004537 002700      JSR      R5,TEST      ;CHECK THE MNCDI
155 002146 003534              DIGO   ;INTR. STARTUP ADDRESS
156 002150 001264              DIBASE ;STARTING DI ADDRESS
157 002152 000010              10     ;OFFSET TO NEXT UNIT
158 002154 004066              DIPRI  ;DI MESSAGE POINTER
159 002156 000010              8.     ;MAX. # OF DI ON A SYSTEM
160 002160 001276              DICNT  ;# OF DI FOUND ON SYSTEM
161      ;NOW VERIFY THE MNCDO MODULES
162 002162 004537 002700      JSR      R5,TEST      ;CHECK THE MNCDO
163 002166 003560              DOGO   ;INTR. STARTUP ADDRESS
164 002170 001266              DOBASE ;STARTING DO ADDRESS
165 002172 000004              4      ;OFFSET TO NEXT UNIT
166 002174 004100              DOPRI ;DO MESSAGE POINTER
167 002176 000010              8.     ;MAX. # OF DO ON A SYSTEM
168 002200 001300              DOCNT  ;# OF DO FOUND ON SYSTEM
169      ;NOW REPORT THE TOTAL COUNT OF EACH MINC-11 MODULE
170
171 002202 104401 001161      TYPE    ,SCLRF      ;FRESH LINE
172 002206 104401 004141      TYPE    ,NIPRI     ;TELL THE #
173 002212 104401 004030      TYPE    ,ADPRI     ;A/D
174 002216 013746 001270      MOV     ADCNT,-(SP)
175 002222 104403              TYPOS
176 002224 001 000          .BYTE  1,0
177 002226 104401 004042      TYPE    ,KWPRI     ;KW
178 002232 013746 001272      MOV     KWCNT,-(SP)
179 002236 104403              TYPOS
180 002240 002 000          .BYTE  2,0
181 002242 104401 004054      TYPE    ,AAPRI     ;AA
182 002246 013746 001274      MOV     AACNT,-(SP)
183 002252 104403              TYPOS
184 002254 002 000          .BYTE  2,0
185 002256 104401 004066      TYPE    ,DIPRI     ;DI
186 002262 013746 001276      MOV     DICNT,-(SP)
187 002266 104403              TYPOS
188 002270 002 000          .BYTE  2,0
189 002272 104401 004100      TYPE    ,DOPRI     ;DO
190 002276 013746 001300      MOV     DOCNT,-(SP)
191 002302 104403              TYPOS
192 002304 002 000          .BYTE  2,0
193 002306 000240              NOP
194 002310 000240              NOP
195 002312 000240              NOP
196 002314 000240              NOP

```

```

198 ; DETERMINE IF THE MEMORY USAGE MAP SHOULD BE REPORTED
199 002316 105737 001205 TSTB $ENVM ; TEST IF APT
200 002322 001006 BNE 1$ ; BR IF YES
201 002324 005737 000042 TST @#42 ; TEST IF XXDP
202 002330 001003 BNE 1$ ; BR IF YES
203 002332 005737 001314 TST FLAGMP ; TEST IF OPER. ASKED FOR MAP
204 002336 001002 BNE 2$ ; BR IF YES
205 002340 000137 002566 1$: JMP $EOP ; BYPASS REPORT
206 002344 005037 001314 2$: CLR FLAGMP ; ENSURE ONLY THE 1ST PASS IT'S REPORTED
207 002350 104401 004474 TYPE ,MNCMAP ; INFORM THE OPER. THE HEADER
208 002354 013746 000004 MOV ERRVEC,-(SP) ; SAVE CURRENT BUS TRAP VALUE
209 002360 012700 170000 MOV #170000,R0 ; GET 1ST ADDRESS TO MAP
210 002364 010037 001312 MOV R0,OUTADR ; AND SAVE FOR TYPEOUT
211 002370 012701 100000 MOV #BIT15,R1 ; PRIME THE ROTATING POINTER
212 002374 005002 CLR R2 ; CLEAR TEMP MASK
213 002376 012703 010114 MOV #ADRPOK,R3 ; LOAD MAP BLOCK POINTER
214 002402 012737 002546 000004 MOV #4$,ERRVEC ; LOAD ADDRESS TIMEOUT RETURN
215 002410 005710 3$: TST (R0) ; TEST THE ADDRESS
216 002412 050102 BIS R1,R2 ; IF NO TIMEOUT-MARK A 1
217 002414 062700 000002 ADD #2,R0 ; UPDATE ADDRESS
218 002420 005710 TST (R0) ; TEST THE ADDRESS
219 002422 050102 BIS R1,R2 ; IF NO TIMEOUT-MARK A 1
220 002424 062700 000002 ADD #2,R0 ; UPDATE ADDRESS
221 002430 000257 CCC ; ENSURE CLEAR CARRY
222 002432 006001 ROR R1 ; MOVE RIGHT
223 002434 103365 BCC 3$ ; BR IF MORE BITS LEFT
224 002436 010223 MOV R2,(R3)+ ; SAVE IN MAP BLOCKS
225 002440 005002 CLR R2 ; CLEAR MASK
226 002442 012701 100000 MOV #BIT15,R1 ; LOAD ROTATING BIT
227 002446 022703 010314 CMP #ADRTOP,R3 ; CHECK IF MAP BLOCKS FILLED
228 002452 001356 BNE 3$ ; BR IF NOT
229 ; NOW REPORT THE MEMORY MAP
230 002454 012700 010114 MOV #ADRPOK,R0 ; LOAD MEMORY MAP POINTER
231 002460 104401 001161 5$: TYPE ,SCLF
232 002464 013746 001312 MOV OUTADR,-(SP) ; REPORT ADDRESS
233 002470 104402 TYPOC
234 002472 104401 004205 TYPE ,SPACE1
235 002476 004737 002554 JSR PC,OUTBIN ; OUTPUT BINARY BITS
236 002502 004737 002554 JSR PC,OUTBIN
237 002506 004737 002554 JSR PC,OUTBIN
238 002512 004737 002554 JSR PC,OUTBIN
239 002516 062737 000400 001312 ADD #400,OUTADR ; UPDATE ADDRESS VALUE
240 002524 022700 010314 CMP #ADRTOP,R0 ; TEST IF FINISHED
241 002530 001353 BNE 5$ ; BR IF NOT
242 002532 012637 000004 MOV (SP)+,ERRVEC ; RESTORE BUS TRAP VECTOR
243 002536 104401 001161 TYPE ,SCLF
244 002542 000137 002566 JMP $EOP
245 ; RETURN TO HERE UPON MEMORY TIME-OUT
246 002546 062716 000002 4$: ADD #2,(SP) ; ADJUST RETURN ADDRESS
247 002552 000002 RTI
248 ; SUBROUTINE TO REPORT I/O FORMAT MESSAGES
249 002554 012046 OUTBIN: MOV (R0)+,-(SP)
250 002556 104406 TYPBN
251 002560 104401 004205 TYPE ,SPACE1 ; INSERT A SPACE

```

252 002564 000207 RTS PC

```
254          .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 LOOP
(1)
(1) 002566    $EOP:
(2) 002566    000240      NOP
(1) 002570    005237    001172      INC      $PASS      ;;INCREMENT THE PASS NUMBER
(1) 002574    042737    100000    001172      BIC      #100000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
(1) 002602    005327      DEC      (PC)+      ;;LOOP?
(1) 002604    000001    $EOPCT: .WORD      1
(1) 002606    003022      BGT      $DOAGN      ;;YES
(1) 002610    012737      MOV      (PC)+,@(PC)+  ;;RESTORE COUNTER
(1) 002612    000001    $ENDCT: .WORD      1
(1) 002614    002604      $EOPCT
(1) 002616    104401    002663      TYPE     ,SENDMG      ;;TYPE 'END PASS #'
(2) 002622    013746    001172      MOV      $PASS,-(SP)  ;;SAVE $PASS FOR TYPEOUT
(2) 002626    104405      TYPDS   ,SENDMG      ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 002630    104401    002660      TYPE     ,SENULL     ;;TYPE A NULL CHARACTER
(1) 002634    013700    000042    $GET42: MOV      @#42,R0  ;;GET MONITOR ADDRESS
(1) 002640    001405      BEQ     $DOAGN      ;;BRANCH IF NO MONITOR
(1) 002642    000005      RESET  ;;CLEAR THE WORLD
(1) 002644    004710    $ENDAD: JSR     PC,(R0)  ;;GO TO MONITOR
(1) 002646    000240      NOP     ;;SAVE ROOM
(1) 002650    000240      NOP     ;;FOR
(1) 002652    000240      NOP     ;;ACT11
(1) 002654
(1) 002654    000137    $DOAGN: JMP      @(PC)+      ;;RETURN
(1) 002656    002062    $RTNAD: .WORD     LOOP
(1) 002660     377      377      000    $ENULL: .BYTE    -1,-1,0  ;;NULL CHARACTER STRING
(1) 002663     015    042412  042116    $ENDMG: .ASCIZ   <15><12>/END PASS #/
(1) 002670    050040    051501  020123
(1) 002676    000043
```

```

256 ;SUBROUTINE TO VERIFY THE BASE AND VECTOR ADDRESSES
257
258 002700 012537 003222 TEST:  MOV (R5)+,67$ ;GET INTR. STARTUP ADDRESS
259 002704 013537 001126      MOV @ (R5)+,$BDDAT ;GET ADDRESS
260 002710 012537 001302      MOV (R5)+,VADR ;GET ADDRESS OFFSET
261 002714 012537 003036      MOV (R5)+,70$ ;GET NAME POINTER
262 002720 012537 001306      MOV (R5)+,TEMP ;GET MAX # OF UNITS
263 002724 013746 000004      MOV @ERRVEC,-(SP) ;SAVE ERRVEC
264 002730 005037 001310      CLR NBEXT ;CLEAR COUNTER
265 ;ADDRESS THE DEVICE TO VERIFY IT EXISTS
266 002734 012737 003140 000004 1$:  MOV #2$,ERRVEC ;SET UP TIME-OUT RETURN
267 002742 005777 176160      TST @SBDDAT ;ADDRESS THE UNIT
268 ;NOW CAUSE AN INTR. AND FIND THE VECTOR
269 002746 005037 001304 60$:  CLR VECT ;CLEAR VECTOR FLAG
270 002752 012737 003224 000004      MOV #IOTRD,ERRVEC ;SET UP INTELL. TRAP RETURN
271 002760 005737 003222      TST 67$ ;TEST IF NON-INTR DEVICE
272 002764 001414      BEQ 65$ ;BR IF NOT INTR. DEVICE
273 002766 004737 003306      JSR PC,FIXVCT ;LOAD VECTOR TRAP
274 002772 004777 000224      JSR PC,@67$ ;GO PRIME THE DEVICE
275 002776 012700 000000      MOV #0,R0 ;PRIME A COUNTER
276 003002 005046      CLR -(SP) ;LOWER
277 003004 012746 003012      MOV #66$,-(SP) ;PS
278 003010 000002      RTI
279 003012 005300      66$:  DEC R0 ;DELAY
280 003014 001376      BNE 66$ ;SOME TIME
281 003016 005077 176104      65$:  CLR @SBDDAT ;LONG ENOUGH DELAY
282 ;REPORT THE ADDRESS AND VECTOR VALUES ON FIRST PASS
283 003022 005737 001172      TST $PASS ;CHECK IF FIRST
284 003026 001033      BNE 73$ ;BR IF NOT
285 003030 104401 001161      TYPE ,$CRLF ;ENSURE FRESH LINE
286 003034 104401      TYPE ;REPORT THE NAME
287 003036 000000      70$:  0 ;ADDRESS OF MESSAGE POINTER
288 003040 104401 004147      TYPE ,ATPRI ;TYPE ADDRESS STARTUP
289 003044 013746 001126      MOV $BDDAT,-(SP) ;GET ADDRESS
290 003050 104403      TYPOS ;REPORT IT
291 003052 006 001      .BYTE 6,1
292 003054 104401 004165      TYPE ,VTPRI ;TYPE VECTOR STATUP
293 003060 005737 001304      TST VECT ;CHECK IF INTR OCCURRED
294 003064 001003      BNE 71$ ;BR IF YES
295 003066 104401 004112      TYPE ,NOPRI ;TELL OPER. NONE
296 003072 000404      BR 72$
297 003074 013746 001304      71$:  MOV VECT,-(SP) ;GET VECTOR VALUE
298 003100 104403      TYPOS ;REPORT IT
299 003102 003 001      .BYTE 3,1
300 ;SENSE IF A/D IF SO CHECK OPERATOR INPUT ABOUT CHANNEL MODE REPORT
301 003104 005737 001316      72$:  TST FLAGAD ;IS CHANNEL MODE REPORT ENABLED?
302 003110 001402      BEQ 73$ ;BR IF NOT
303 003112 004737 003620      JSR PC,TCHANL ;REPORT THE CONFIGURATION

```

```

305          ;BUMP THE COUNTER AND TEST IF FINISHED?
306 003116 005237 001310 73$: INC NBEXT ;INCREMENT UNIT COUNTER
307 003122 005337 001306      DEC TEMP ;REACHED MAX?
308 003126 001424      BEQ 4$ ;:REACHED MAX NO OF UNIT'S
309 003130 063737 001302 001126  ADD VADR,$BDDAT ;GET NEXT UNIT
310 003136 000676      BR 1$ ;:TRY NEXT UNIT
311          ;COME HERE UPON ADDRESS TIME-OUT
312 003140 022626 2$: CMP (SP)+,(SP)+ ;POP 2 WORDS OFF STACK
313 003142 005737 001310      TST NBEXT ;CHECK IF ANY UNITS
314 003146 001014      BNE 4$ ;BR IF SOME
315 003150 005737 001172      TST $PASS ;TEST IF FIRST PASS
316 003154 001011      BNE 4$ ;BR IF NOT FIRST
317 003156 013737 003036 003172  MOV 70$,76$ ;GET DEVICE MESSAGE POINTER
318 003164 104401 001161      TYPE ;$CRLF
319 003170 104401      TYPE ;TELL OPER. IT'S NOT THERE
320 003172 000000 76$: 0
321 003174 104401 004112      TYPE ,NOPRI ;TELL OPER. NONE FOUND
322 003200 000240 4$: NOP
323 003202 000240      NOP
324 003204 013735 001310      MOV NBEXT,@(R5)+ ;SAVE THE # OF UNITS FOUND
325 003210 012637 000004      MOV (SP)+,ERRVEC ;RESTORE ERRVEC
326 003214 005037 001316      CLR FLAGAD ;CLEAR EXPANDED A/D REPORT FLAG
327 003220 000205      RTS R5
328 003222 003456 67$: ADGO ;ADDRESS TO DEVICE PRIMER
329
330          ;INTELLIGENT INTERRUPT HANDLER
331 003224 011637 003304 IOTRD: MOV (SP),TRTO ;GET ADDRESS
332 003230 162737 000004 003304  SUB #4,TRTO ;ADJUST VALUE
333 003236 023727 003304 001000  CMP TRTO,#1000 ;CHECK IF VECTOR OR FATAL TRAP
334 003244 003402      BLE 2$ ;BR IF VECTOR
335 003246 000000 1$: HALT ;FATAL BUS TRAP IN PROGRAM
336 003250 000776      BR 1$
337 003252 013737 003304 001304 2$: MOV TRTO,VECT ;LOAD VECTOR
338 003260 005077 175642      CLR @BDDAT ;CLEAR CURRENT UNIT STATUS
339 003264 022626      CMP (SP)+,(SP)+
340 003266 000240      NOP
341 003270 000240      NOP
342 003272 000002      RTI ;EXIT
343 003274 000240      NOP
344 003276 000240      NOP
345 003300 000240      NOP
346 003302 000240      NOP
347 003304 000000      TRTO: 0

```

```

349          ;SUBROUTINE TO LOAD INTELLIGENT TRAP CATCHER
350 003306 012700 000020      FIXVCT: MOV #20,R0      ;LOAD STARTING ADDRESS
351 003312 012701 000022      MOV #22,R1      ;LOAD STARTING POINTER
352 003316 012702 004700      MOV #4700,R2    ;LOAD 'BAD INSTR'
353 003322 005037 000000      CLR @0        ;CLEAR LOC. 0
354 003326 005037 000002      CLR @2        ;CLEAR LOC. 2
355 003332 005037 000010      CLR @10       ;CLEAR LOC. 10
356 003336 005037 000012      CLR @12       ;CLEAR LOC. 12
357 003342 022700 000060      1$:  CMP #60,R0    ;TEST IF TKB VECTOR
358 003346 001432             BEQ 3$        ;BR IF YES
359 003350 022700 000100      CMP #100,R0  ;TEST IF LINE INTR. <B EVENT>
360 003354 001425             BEQ 2$        ;BR IF YES
361 003356 022700 000040      CMP #40,R0   ;TEST FOR XXDP FLAG VECTOR
362 003362 001422             BEQ 2$        ;BR IF YES
363 003364 022700 000200      CMP #200,R0  ;TEST IF STARTING ADDRESS
364 003370 001421             BEQ 3$        ;BR IF YES
365 003372 022700 000034      CMP #34,R0   ;TEST IF TRAP VECTOR
366 003376 001414             BEQ 2$        ;BR IF YES
367 003400 000240             NOP
368 003402 000240             NOP
369 003404 000240             NOP
370 003406 000240             NOP
371 003410 000240             NOP
372 003412 010120             MOV R1,(R0)+  ;LOAD POINTER
373 003414 010220             MOV R2,(R0)+  ;LOAD 'BAD'
374 003416 022121             CMP (R1)+,(R1)+ ;BUMP POINTER
375 003420 020027 001000      CMP R0,#1000 ;DONE ?
376 003424 001346             BNE 1$        ;BR IF NOT
377 003426 000207             RTS PC         ;RETURN
378 003430 022020             2$:  CMP (R0)+,(R0)+ ;BUMP
379 003432 022121             CMP (R1)+,(R1)+
380 003434 022020             3$:  CMP (R0)+,(R0)+
381 003436 022121             CMP (R1)+,(R1)+
382 003440 000740             BR 1$
383 003442 000240             NOP
384 003444 000240             NOP
385 003446 000240             NOP
386 003450 000240             NOP
387 003452 000240             NOP
388 003454 000240             NOP

```

```

390
391 003456 012777 000101 175442 ;SUBROUTINE TO PRIME THE MNCAD
ADGO: MOV #101,@$BDDAT ;ENABLE INTR. AND START CONVERSION
392 003464 000240 NOP
393 003466 000240 NOP
394 003470 000240 NOP
395 003472 000207 RTS PC ;EXIT
396
397 003474 013700 001126 ;SUBROUTINE TO PRIME THE MNCKW
KWGO: MOV $BDDAT,R0 ;GET ADDR.
398 003500 005200 INC R0
399 003502 005200 INC R0 ;MAKE ADDRESS OF CLOCK PRESET BUFFER
400 003504 012710 177777 MOV #-1,(R0) ;LOAD CLOCK PRESET
401 003510 012777 000161 175410 MOV #161,@$BDDAT ;LOAD RATE, INTR ENABLE AND GO
402 003516 052777 000400 175402 BIS #BIT8,@$BDDAT ;LOAD MAINT CLOCK
403 003524 000207 RTS PC ;EXIT
404 003526 000240 NOP
405 003530 000240 NOP
406 003532 000240 NOP
407
408 003534 012777 000102 175364 ;SUBROUTINE TO PRIME THE MNCDI
DIGO: MOV #102,@$BDDAT ;LOAD INTR. ENABLE AND MODE
409 003542 052777 004200 175356 BIS #4200,@$BDDAT ;LOAD MAINT. STROBE
410 003550 000207 RTS PC ;EXIT
411 003552 000240 NOP
412 003554 000240 NOP
413 003556 000240 NOP
414
415 003560 013700 001126 ;SUBROUTINE TO PRIME THE MNCDO
DOGO: MOV $BDDAT,R0 ;GET ADDRESS
416 003564 062700 000003 ADD #3,R0 ;MAKE OUTPUT DATA POINTER
417 003570 105010 CLRB (R0) ;ENABLE MAINT. STROBE
418 003572 162700 000002 SUB #2,R0 ;MAKE HIGH BYTE STATUS ADDRESS
419 003576 112710 000001 MOVB #BIT0,(R0) ;MAINT. STROBE
420 003602 052777 000100 175316 BIS #BIT6,@$BDDAT ;ENABLE INTR.
421 003610 000207 RTS PC ;EXIT
422 003612 000240 NOP
423 003614 000240 NOP
424 003616 000240 NOP

```



```

426 ;SUB-ROUTINE TO TYPE OUT A/D CONFIGURATION
427
428 003620 104401 001161 TCHANL: TYPE , $CRLF
429 003624 000240 NOP
430 003626 000240 NOP
431 003630 000240 NOP
432 003632 013702 001126 MOV $BDDAT,R2
433 003636 062702 000002 ADD #2,R2 ;MAKE CONVERTED VALUE ADDRESS
434 003642 011201 MOV (R2),R1 ;ENSURE A/D FLAG IS CLEARED
435 003644 005001 CLR R1
436 003646 000240 NOP
437 003650 005277 175252 2$: INC @SBDDAT ;START A CONVERSION
438 003654 105777 175246 3$: TSTB @SBDDAT ;WAIT FOR CONVERSION TO FINISH
439 003660 100375 BPL 3$
440 003662 011200 MOV (R2),R0 ;GET RESULTS
441 003664 104401 004202 TYPE ,SSPACE ;MOVE IT OVER SOME
442 003670 010146 MOV R1,-(SP) ;;SAVE R1 FOR TYPEOUT
(1) 003672 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 003674 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
(1) 003675 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
443 003676 104401 004207 TYPE ,MDASH ;TYPE A DASH
444 003702 062701 000003 ADD #3,R1 ;ADD 3 TO CHANNEL FOR DIFFERENTIAL
445 003706 042700 007777 BIC #7777,R0 ;IS CHANNEL SINGLE ENDED
446 003712 001002 BNE 5$ ;CHANNEL IS NOT SINGLE ENDED
447 003714 062701 000004 ADD #4,R1 ;ADD 4 CHANNELS FOR SINGLE ENDED
448 003720 022701 000100 5$: CMP #100,R1 ;IS CHANNEL > LAST POSSIBLE CHANNEL
449 003724 101002 BHI 6$ ;:NO
450 003726 012701 000077 6$: MOV #77,R1 ;YES, SET TO LAST CHANNEL
(1) 003732 010146 MOV R1,-(SP) ;;SAVE R1 FOR TYPEOUT
(1) 003734 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 003736 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
(1) 003737 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
452 003740 005700 TST R0 ;IS CHANNEL SINGLE ENDED?
453 003742 001003 BNE 7$ ;:NO
454 003744 104401 004213 TYPE ,MSE ;TYPE SINGLE ENDED MESSAGE
455 003750 000410 BR 9$ ;GO TEST MORE CHANNELS
456 003752 022700 020000 7$: CMP #BIT13,R0 ;IS CHANNEL DIFFERENTIAL?
457 003756 001003 BNE 8$ ;:NO, MUST HAVE PREAMP
458 003760 104401 004232 TYPE ,MDIF ;TYPE DIFFERENTIAL MESSAGE
459 003764 000402 BR 9$ ;GO TEST MORE CHANNELS
460 003766 104401 004251 8$: TYPE ,MPRMP ;TYPE PREAMP MESSAGE
461 003772 005201 9$: INC R1 ;SET CHANNEL TO NEXT SET OF CHANNELS
462 003774 022701 000100 CMP #100,R1 ;DONE?
463 004000 001412 BEQ 10$ ;:YES
464 004002 010100 MOV R1,R0 ;GET CHANNEL
465 004004 000300 SWAB R0 ;PUT CHANNEL NUMBER IN HIGH BYTE
466 004006 052700 000010 BIS #BIT3,R0 ;SET STATUS ENABLE BIT
467 004012 010077 175110 MOV R0,@SBDDAT ;LOAD INTO A/D STATUS REGISTER
468 004016 032777 000002 175102 BIT #BIT1,@SBDDAT ;IS NON-EXISTENT CHANNEL BIT SET?
469 004024 001711 BEQ 2$ ;:NO
470 004026 000207 10$: RTS PC

```

```

472                                     ;ASCII MESSAGES
473 004030 020040 046473 041516 ADPRI: .ASCIZ / ;MNCAD /
      004036 042101 000040
474 004042 020040 046473 041516 KWPRI: .ASCIZ / ;MNCKW /
      004050 053513 000040
475 004054 020040 046473 041516 AAPRI: .ASCIZ / ;MNCAA /
      004062 040501 000040
476 004066 020040 046473 041516 DIPRI: .ASCIZ / ;MNCDI /
      004074 044504 000040
477 004100 020040 046473 041516 DOPRI: .ASCIZ / ;MNCDO /
      004106 047504 000040
478 004112 025040 020052 042040 NOPRI: .ASCIZ / ** DOES NOT EXIST **/
      004120 042517 020123 047516
      004126 020124 054105 051511
      004134 020124 025052 000
479 004141 043 047440 020106 NIPRI: .ASCIZ /# OF /
      004146 000
480 004147 101 020124 042101 ATPRI: .ASCIZ /AT ADDRESS = /
      004154 051104 051505 020123
      004162 020075 000
481 004165 040 020040 042526 VTPRI: .ASCIZ / VECTOR = /
      004172 052103 051117 036440
      004200 000040
482 004202 004411 000 SSPACE: .ASCIZ / /
483 004205 040 000 SPACE1: .ASCIZ / /
484 004207 040 020055 000 MDASH: .ASCIZ / - /
485 004213 040 044523 043516 MSE: .ASCIZ / SINGLE ENDED/<200>
      004220 042514 042440 042116
      004226 042105 000200
486 004232 042040 043111 042506 MDIF: .ASCIZ / DIFFERENTIAL/<200>
      004240 042522 052116 040511
      004246 100114 000
487 004251 040 051120 040505 MPRMP: .ASCIZ / PREAMP/<200>
      004256 050115 000200
488 004262 042200 020117 047531 QUESAD: .ASCII <200>\DO YOU WANT THE MNCAD (A/D) CHANNEL MODE REPORT ?\
      004270 020125 040527 052116
      004276 052040 042510 046440
      004304 041516 042101 024040
      004312 027501 024504 041440
      004320 040510 047116 046105
      004326 046440 042117 020105
      004334 042522 047520 052122
      004342 037440
489 004344 004600 052011 050131 .ASCIZ <200>\ TYPE 'Y' FOR YES \
      004352 020105 054442 020042
      004360 047506 020122 042531
      004366 020123 020075 000
490 004373 200 047504 054440 QUESMP: .ASCII <200>\DO YOU WANT THE MEMORY USAGE MAP REPORT ?\
      004400 052517 053440 047101
      004406 020124 044124 020105
      004414 042515 047515 054522
      004422 052440 040523 042507
      004430 046440 050101 051040
      004436 050105 051117 020124

```

491	004444	077						
	004445	200	004411	054524	.ASCII	<200>\	TYPE 'Y' FOR YES - \	
	004452	042520	021040	021131				
	004460	043040	051117	054440				
	004466	051505	036440	000040				
492	004474	046600	047111	026503	MNCMAP: .ASCII	<200>\MINC-11 MEMORY USAGE MAP (EACH BIT - 2 ADDRESSES)\<200>		
	004502	030461	046440	046505				
	004510	051117	020131	051525				
	004516	043501	020105	040515				
	004524	004520	042450	041501				
	004532	020110	044502	020124				
	004540	020075	020062	042101				
	004546	051104	051505	042523				
	004554	024523	200					
493	004557	040	042101	051104	.ASCII	\ ADDR	000/400	100/500\
	004564	020011	020040	030040				
	004572	030060	032057	030060				
	004600	020040	020040	020040				
	004606	020040	020040	030061				
	004614	027460	030065	060				
494	004621	040	020040	020040	.ASCII	\	200/600	300/700\
	004626	020040	020040	031040				
	004634	030060	033057	030060				
	004642	020040	020040	020040				
	004650	020040	020040	030063				
	004656	027460	030067	000060				
495	004664	015	012	012	SETMNC: .BYTE	15,12,12		
496	004667	117	042520	040522	.ASCII	/OPERATOR	-	PLEASE DO THE FOLLOWING: /
	004674	047524	004522	004455				
	004702	046120	040505	042523				
	004710	042040	020117	044124				
	004716	020105	047506	046114				
	004724	053517	047111	035107				
497	004732	015	012	012	.BYTE	15,12,12		
498	004735	122	046505	053117	.ASCII	/REMOVE THE CUSTOMER CONNECTIONS FROM EACH 'MINC-11' OPTION/		
	004742	020105	044124	020105				
	004750	052503	052123	046517				
	004756	051105	041440	047117				
	004764	042516	052103	047511				
	004772	051516	043040	047522				
	005000	020115	040505	044103				
	005006	021040	044515	041516				
	005014	030455	021061	047440				
	005022	052120	047511	116				
499	005027	015	012		.BYTE	15,12		
500	005031	115	041516	042101	.ASCII	\MNCAD (A/D)		SET FRONT PANEL SWITCHES TO THE 'TEST' POSITION\
	005036	024040	027501	024504				
	005044	004411	042523	020124				
	005052	051106	047117	020124				
	005060	040520	042516	020114				
	005066	053523	052111	044103				
	005074	051505	052040	020117				
	005102	044124	020105	052042				
	005110	051505	021124	050040				

	005116	051517	052111	047511		
	005124	116				
501	005125	015	012		.BYTE	15,12
502	005127	115	041516	053513	.ASCII	\MNCKW (CLOCK)
	005134	024040	046103	041517		PULL OUT 'ST1' AND 'ST2' SWITCHES\
	005142	024513	004411	052520		
	005150	046114	047440	052125		
	005156	021040	052123	021061		
	005164	040440	042116	021040		
	005172	052123	021062	051440		
	005200	044527	041524	042510		
	005206	123				
503	005207	015	012		.BYTE	15,12
504	005211	011	004411	047101	.ASCII	\
	005216	020104	047522	040524		AND ROTATE FULLY CLOCKWISE\
	005224	042524	043040	046125		
	005232	054514	041440	047514		
	005240	045503	044527	042523		
505	005246	015	012		.BYTE	15,12
506	005250	047115	042103	020111	.ASCII	\MNCDI (DIGITAL IN)
	005256	042050	043511	052111		SET 'DATA' SWITCH TO THE '-' POSITION\
	005264	046101	044440	024516		
	005272	051411	052105	021040		
	005300	040504	040524	020042		
	005306	053523	052111	044103		
	005314	052040	020117	044124		
	005322	020105	026442	020042		
	005330	047520	044523	044524		
	005336	047117				
507	005340	015	012		.BYTE	15,12
508	005342	046123	020125	004460	.ASCII	\SLU 0
	005350	004411	047111	052123		INSTALL 'SLU TEST CONNECTOR'\
	005356	046101	020114	051442		
	005364	052514	052040	051505		
	005372	020124	047503	047116		
	005400	041505	047524	021122		
509	005406	015	012		.BYTE	15,12
510	005410	046123	020125	004461	.ASCII	\SLU 1
	005416	004411	047111	052123		INSTALL 'SLU TEST CONNECTOR'\
	005424	046101	020114	051442		
	005432	052514	052040	051505		
	005440	020124	047503	047116		
	005446	041505	047524	021122		
511	005454	015	012		.BYTE	15,12
512	005456	046123	020125	004462	.ASCII	\SLU 2
	005464	004411	047111	052123		INSTALL 'SLU TEST CONNECTOR'\
	005472	046101	020114	051442		
	005500	052514	052040	051505		
	005506	020124	047503	047116		
	005514	041505	047524	021122		
513	005522	015	012	012	.BYTE	15,12,12
514	005525	104	050105	042522	.ASCII	\DEPRESS A KEY ON THE CONSOLE TERMINAL WHEN READY. \
	005532	051523	040440	045440		
	005540	054505	047440	020116		

005546	044124	020105	047503
005554	051516	046117	020105
005562	042524	046522	047111
005570	046101	053440	042510
005576	020116	042522	042101
005604	027131	000040	

515
516
517

.EVEN

```

519      .SBTTL  TTY INPUT ROUTINE
(1)
(2)      ::*****
(1)      .ENABL  LSB
(1) 005610 000000 $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
(1) 005612 000000 $TKQIN: .WORD 0      ;;INPUT POINTER
(1) 005614 000000 $TKQOUT: .WORD 0     ;;OUTPUT POINTER
(1) 005616 000040 $TKQSRT: .BLKB 32.   ;;TTY KEYBOARD QUEUE
(1)      $TKQEND=.
(1)
(1)      ;*TK INITIALIZE ROUTINE
(1)      ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
(1)      ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
(1)      ;
(1)      ;*CALL:
(1)      ;*      JSR      PC,$TKINT
(1)      ;*      RETURN
(1)      ;
(1) 005656 005037 005610 $TKINT: CLR  $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
(1) 005662 012737 005616 005612 MOV  # $TKQSRT,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
(1) 005670 013737 005612 005614 MOV  $TKQIN,$TKQOUT    ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1) 005676 012737 005726 000060 MOV  # $TKSRV,@TKVEC   ;;INITIALIZE THE KEYBOARD VECTOR
(1) 005704 012737 000200 000062 MOV  #200,@TKVEC+2    ;;'BR' LEVEL 4
(1) 005712 005777 173230 TST  @TKB             ;;CLEAR DONE FLAG
(1) 005716 012777 000100 173220 MOV  #100,@TKS        ;;ENABLE TTY KEYBOARD INTERRUPT
(1) 005724 000207      RTS      PC      ;;RETURN TO CALLER
(1)
(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 (BEGIN)
(1)      ;
(1) 005726 117746 173214 $TKSRV: MOVB  @TKB,-(SP)  ;;PICKUP THE CHARACTER
(1) 005732 042716 177600 BIC  #^C177,(SP)      ;;STRIP THE JUNK
(1) 005736 021627 000003 CMP  (SP),#3          ;;IS IT A CONTROL C?
(1) 005742 001007 BNE  1$              ;;BRANCH IF NO
(1) 005744 104401 006516 TYPE  ,SCNTLC          ;;TYPE A CONTROL-C (^C)
(1) 005750 004737 005656 JSR  PC,$TKINT        ;;INIT THE KEYBOARD
(1) 005754 005726 TST  (SP)+          ;;CLEAN UP STACK
(1) 005756 000137 001324 JMP  BEGIN           ;;CONTROL C RESTART
(1)
(1) 005762      1$:
(1) 005762 022737 000040 005610 CMP  #32,$TKCNT      ;;IS THE QUEUE FULL?
(1) 005770 001004 BNE  3$              ;;BRANCH IF NO
(1) 005772 104401 006512 TYPE  ,SBELL          ;;RING THE TTY BELL
(1) 005776 005726 TST  (SP)+          ;;CLEAN CHARACTER OFF OF STACK
(1) 006000 000451 BR   5$              ;;EXIT
(1) 006002 021627 000023 3$: CMP  (SP),#23          ;;IS IT A CONTROL-S?
(1) 006006 001021 BNE  32$            ;;BRANCH IF NO
(1) 006010 005077 173130 CLR  @TKS            ;;DISABLE TTY KEYBOARD INTERRUPTS
(1) 006014 005726 TST  (SP)+          ;;CLEAN CHAR OFF STACK
(1) 006016 105777 173122 31$: TSTB @TKS            ;;WAIT FOR A CHAR

```

```
(1) 006022 100375           BPL      31$           ;; LOOP UNTIL ITS THERE
(1) 006024 117746 173116   MOVB     @STKB, -(SP)   ;; GET THE CHARACTER
(1) 006030 042716 177600   BIC     #^C177, (SP)  ;; MAKE IT 7-BIT ASCII
(1) 006034 022627 000021   CMP     (SP)+, #21    ;; IS IT A CONTROL-Q?
(1) 006040 001366           BNE     31$           ;; BRANCH IF NO
(1) 006042 012777 000100 173074   MOV     #100, @STKS   ;; REENABLE TTY KEYBOARD INTERRUPTS
(1) 006050 000002           RTI                     ;; RETURN
(1) 006052 005237 005610   32$:    INC     $TKCNT    ;; COUNT THIS CHARACTER
(1) 006056 021627 000140   CMP     (SP), #140    ;; IS IT UPPER CASE?
(1) 006062 002405           BLT     4$            ;; BRANCH IF YES
(1) 006064 021627 000175   CMP     (SP), #175    ;; IS IT A SPECIAL CHAR?
(1) 006070 003002           BGT     4$            ;; BRANCH IF YES
(1) 006072 042716 000040   BIC     #40, (SP)     ;; MAKE IT UPPER CASE
(1) 006076 112677 177510   4$:    MOVB     (SP)+, @STKQIN ;; AND PUT IT IN QUEUE
(1) 006102 005237 005612   INC     $TKQIN        ;; UPDATE THE POINTER
(1) 006106 023727 005612 005656   CMP     $TKQIN, #STKQEND ;; GO OFF THE END?
(1) 006114 001003           BNE     5$            ;; BRANCH IF NO
(1) 006116 012737 005616 005612   MOV     #STKQSRT, $TKQIN ;; RESET THE POINTER
(1) 006124 000002   5$:    RTI                     ;; RETURN
(1)
(1) .DSABL  LSB
(1)
(1)
(2)
(1) ;*****
(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) ;RDCHR: MOV      (SP), -(SP)  ;; PUSH DOWN THE PC AND
(1) 006126 011646           MOV     4(SP), 2(SP)   ;; THE PS
(1) 006130 016666 000004 000002   CLR     4(SP)         ;; GET READY FOR A CHARACTER
(1) 006136 005066 000004           CLR     -(SP)         ;; PUT NEW PS ON STACK
(2) 006142 005046           MOV     #64$, -(SP)   ;; PUT NEW PC ON STACK
(2) 006144 012746 006152           RTI                     ;; POP NEW PC AND PS
(2) 006150 000002   64$:
(1) 006152 005737 005610   1$:    TST     $TKCNT        ;; WAIT ON A CHARACTER
(1) 006156 001775           BEQ     1$            ;;
(1) 006160 005337 005610   DEC     $TKCNT        ;; DECREMENT THE COUNTER
(1) 006164 117766 177424 000004   MOVB     @STKQOUT, 4(SP) ;; GET ONE CHARACTER
(1) 006172 005237 005614   INC     $TKQOUT        ;; UPDATE THE POINTER
(1) 006176 023727 005614 005656   CMP     $TKQOUT, #STKQEND ;; DID IT GO OFF OF THE END?
(1) 006204 001003           BNE     2$            ;; BRANCH IF NO
(1) 006206 012737 005616 005614   MOV     #STKQSRT, $TKQOUT ;; RESET THE POINTER
(1) 006214 000002   2$:    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) 006216 010346          $RDLIN: MOV    R3,-(SP)          ;;SAVE R3
(1) 006220 005046          CLR    -(SP)          ;;CLEAR THE RUBOUT KEY
(1) 006222 012703 006452  1$:    MOV    #$TTYIN,R3      ;;GET ADDRESS
(1) 006226 022703 006512  2$:    CMP    #$TTYIN+32.,R3    ;;BUFFER FULL?
(1) 006232 101456          BLOS   4$             ;;BR IF YES
(1) 006234 104407          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
(1) 006236 112613          MOVB   (SP)+,(R3)     ;;GET CHARACTER
(1) 006240 122713 000177  10$:   CMPB   #177,(R3)     ;;IS IT A RUBOUT
(1) 006244 001022          BNE   5$             ;;BR IF NO
(1) 006246 005716          TST   (SP)          ;;IS THIS THE FIRST RUBOUT?
(1) 006250 001007          BNE   6$             ;;BR IF NO
(1) 006252 112737 000134 006450  MOVB   #'\',9$        ;;TYPE A BACK SLASH
(1) 006260 104401 006450          TYPE  ,9$
(1) 006264 012716 177777          MOV   #-1,(SP)      ;;SET THE RUBOUT KEY
(1) 006270 005303          6$:    DEC    R3          ;;BACKUP BY ONE
(1) 006272 020327 006452          CMP    R3,$$TTYIN   ;;STACK EMPTY?
(1) 006276 103434          BLO   4$             ;;BR IF YES
(1) 006300 111337 006450          MOVB   (R3),9$      ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1) 006304 104401 006450          TYPE  ,9$          ;;GO TYPE
(1) 006310 000746          BR    2$             ;;GO READ ANOTHER CHAR.
(1) 006312 005716          5$:    TST   (SP)          ;;RUBOUT KEY SET?
(1) 006314 001406          BEQ   7$             ;;BR IF NO
(1) 006316 112737 000134 006450  MOVB   #'\',9$        ;;TYPE A BACK SLASH
(1) 006324 104401 006450          TYPE  ,9$
(1) 006330 005016          CLR   (SP)          ;;CLEAR THE RUBOUT KEY
(1) 006332 122713 000025  7$:    CMPB   #25,(R3)     ;;IS CHARACTER A CTRL U?
(1) 006336 001003          BNE   8$             ;;BR IF NO
(1) 006340 104401 006523          TYPE  ,SCNTLU      ;;TYPE A CONTROL 'U'
(1) 006344 000726          BR    1$             ;;GO START OVER
(1) 006346 122713 000022  8$:    CMPB   #22,(R3)     ;;IS CHARACTER A '^R'?
(1) 006352 001011          BNE   3$             ;;BRANCH IF NO
(1) 006354 105013          CLRB  (R3)          ;;CLEAR THE CHARACTER
(1) 006356 104401 001161          TYPE  ,SCRLF      ;;TYPE A 'CR' & 'LF'
(1) 006362 104401 006452          TYPE  ,TTYIN      ;;TYPE THE INPUT STRING
(1) 006366 000717          BR    2$             ;;GO PICKUP ANOTHER CHACTER
(1) 006370 104401 001160  4$:    TYPE  ,QUES        ;;TYPE A '?'
(1) 006374 000712          BR    1$             ;;CLEAR THE BUFFER AND LOOP
(1) 006376 111337 006450  3$:    MOVB   (R3),9$     ;;ECHO THE CHARACTER
(1) 006402 104401 006450          TYPE  ,9$
(1) 006406 122723 000015          CMPB   #15,(R3)+    ;;CHECK FOR RETURN
(1) 006412 001305          BNE   2$             ;;LOOP IF NOT RETURN
(1) 006414 105063 177777          CLRB  -1(R3)       ;;CLEAR RETURN (THE 15)
(1) 006420 104401 001162          TYPE  ,LF          ;;TYPE A LINE FEED
(1) 006424 005726          TST   (SP)+        ;;CLEAN RUBOUT KEY FROM THE STACK
(1) 006426 012603          MOV   (SP)+,R3     ;;RESTORE R3
(1) 006430 011646          MOV   (SP),-(SP)   ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 006432 016666 000004 000002  MOV   4(SP),2(SP)  ;;FIRST ASCII CHARACTER ON IT
(1) 006440 012766 006452 000004  MOV   #$TTYIN,4(SP)
(1) 006446 000002          RTI                ;;RETURN
(1) 006450 000          9$:    .BYTE  0          ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 006451 000          .BYTE  0          ;;TERMINATOR
(1) 006452 000040          $TTYIN: .BLKB  32.  ;;RESERVE 32. BYTES FOR TTY INPUT
(1) 006512 177607 000377          $BELL:  .ASCIZ <207><377><377> ;;CODE FOR BELL
(1) 006516 041536 005015 000          $CNTLC: .ASCIZ /^C/<15><12>    ;;CONTROL 'C'

```



```

(1) 006523      136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
(1) 006530    043536  005015      000  $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
(1) 006535      015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1) 006542    036440  000040
(1) 006546    020040  042516  020127  $MNEW:  .ASCIZ  / NEW = /
(1) 006554    020075      000
(1)           006560
520           .EVEN
(1)           .SBTTL  TYPE ROUTINE
(2)
(1)           ;;*****
(1)           ;;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)           ;;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)           ;;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)           ;;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)           ;;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)           ;;*
(1)           ;;*CALL:
(1)           ;;*1) USING A TRAP INSTRUCTION
(1)           ;;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)           ;;*OR
(1)           ;;*      TYPE
(1)           ;;*      MESADR
(1)           ;;*
(1)           $)TYPE:  TSTB      $)PFLG      ;;IS THERE A TERMINAL?
(1)           006564    100002      BPL      1$      ;;BR IF YES
(1)           006566    000000      HALT     ;;HALT HERE IF NO TERMINAL
(1)           006570    000430      BR      3$      ;;LEAVE
(1)           006572    010046      1$:  MOV    RO,-(SP)      ;;SAVE RO
(1)           006574    017600    000002      MOV    @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
(1)           006600    122737    000001  001204      CMPB   #APTENV,$ENV   ;;RUNNING IN APT MODE
(1)           006606    001011      BNE     62$      ;;NO,GO CHECK FOR APT CONSOLE
(1)           006610    132737    000100  001205      BITB   #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1)           006616    001405      BEQ     62$      ;;NO,GO CHECK FOR CONSOLE
(1)           006620    010037    006630      MOV    RO,61$      ;;SETUP MESSAGE ADDRESS FOR APT
(1)           006624    004737    007576      JSR    PC,$ATY3    ;;SPOOL MESSAGE TO APT
(1)           006630    000000      61$:  .WORD   0      ;;MESSAGE ADDRESS
(1)           006632    132737    000040  001205      62$:  BITB   #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1)           006640    001003      BNE     60$      ;;YES,SKIP TYPE OUT
(1)           006642    112046      2$:  MOVB   (RO)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1)           006644    001005      BNE     4$      ;;BR IF IT ISN'T THE TERMINATOR
(1)           006646    005726      TST    (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
(1)           006650    012600      60$:  MOV    (SP)+,RO    ;;RESTORE RO
(1)           006652    062716    000002      3$:  ADD    #2,(SP)     ;;ADJUST RETURN PC
(1)           006656    000002      RTI    ;;RETURN
(1)           006660    122716    000011      4$:  CMPB   #HT,(SP)    ;;BRANCH IF <HT>
(1)           006664    001430      BEQ     8$
(1)           006666    122716    000200      CMPB   #CRLF,(SP)  ;;BRANCH IF NOT <CRLF>
(1)           006672    001006      BNE     5$
(1)           006674    005726      TST    (SP)+      ;;POP <CR><LF> EQUIV
(1)           006676    104401      TYPE   ;;TYPE A CR AND LF
(1)           006700    001161      $CRLF
(1)           006702    105037    007036      CLRB   $CHARCNT    ;;CLEAR CHARACTER COUNT
(1)           006706    000755      BR     2$      ;;GET NEXT CHARACTER

```

```

(1) 006710 004737 006772      5$: JSR PC,$TYPEC      ;;GO TYPE THIS CHARACTER
(1) 006714 123726 001156      6$: CMPB $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
(1) 006720 001350              BNE 2$                ;;IF NO GO GET NEXT CHAR.
(1) 006722 013746 001154      MOV $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
(1)                                ;;AND THE NULL CHAR.
(1) 006726 105366 000001      7$: DECB 1(SP)        ;;DOES A NULL NEED TO BE TYPED?
(1) 006732 002770              BLT 6$                ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1) 006734 004737 006772      JSR PC,$TYPEC      ;;GO TYPE A NULL
(1) 006740 105337 007036      DECB $CHARCNT      ;;DO NOT COUNT AS A COUNT
(1) 006744 000770              BR 7$                 ;;LOOP

(1)                                ;HORIZONTAL TAB PROCESSOR
(1)
(1) 006746 112716 000040      8$: MOVB #' ,(SP)    ;;REPLACE TAB WITH SPACE
(1) 006752 004737 006772      9$: JSR PC,$TYPEC    ;;TYPE A SPACE
(1) 006756 132737 000007 007036 BITB #7,$CHARCNT    ;;BRANCH IF NOT AT
(1) 006764 001372              BNE 9$                ;;TAB STOP
(1) 006766 005726              TST (SP)+            ;;POP SPACE OFF STACK
(1) 006770 000724              BR 2$                 ;;GET NEXT CHARACTER
(1) 006772 105777 172152      $TYPEC: TSTB @$TPS   ;;WAIT UNTIL PRINTER IS READY
(1) 006776 100375              BPL $TYPEC
(1) 007000 116677 000002 172144 MOVB 2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 007006 122766 000015 000002 CMPB #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
(1) 007014 001003              BNE 1$                ;;BRANCH IF NO
(1) 007016 105037 007036      CLRB $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
(1) 007022 000406              BR $TYPEX             ;;EXIT
(1) 007024 122766 000012 000002 1$: CMPB #LF,2(SP)     ;;IS CHARACTER A LINE FEED?
(1) 007032 001402              BEQ $TYPEX           ;;BRANCH IF YES
(1) 007034 105227              INCB (PC)+          ;;COUNT THE CHARACTER
(1) 007036 000000      $CHARCNT: .WORD 0  ;;CHARACTER COUNT STORAGE
(1) 007040 000207      $TYPEX: RTS PC
(1)

```

```

522          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)          ;*****
(1)          ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)          ;OCTAL (ASCII) NUMBER AND TYPE IT.
(1)          ;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)          ;CALL:
(1)          ;      MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
(1)          ;      TYPOS      ;;CALL FOR TYPEOUT
(1)          ;      .BYTE  N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)          ;      .BYTE  M      ;;M=1 OR 0
(1)          ;                               ;;1=TYPE LEADING ZEROS
(1)          ;                               ;;0-SUPPRESS LEADING ZEROS
(1)          ;
(1)          ;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)          ;$TYPOS OR $TYPOC
(1)          ;CALL:
(1)          ;      MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
(1)          ;      TYPON      ;;CALL FOR TYPEOUT
(1)          ;
(1)          ;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)          ;CALL:
(1)          ;      MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
(1)          ;      TYPOC      ;;CALL FOR TYPEOUT
(1)
(1) 007042 017646 000000          $TYPOS: MOV     @ (SP),-(SP)      ;;PICKUP THE MODE
(1) 007046 116637 000001 007265  MOV     1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
(1) 007054 112637 007267          MOV     (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
(1) 007060 062716 000002          ADD     #2, (SP)          ;;ADJUST RETURN ADDRESS
(1) 007064 000406          BR     $TYPON
(1) 007066 112737 000001 007265  $TYPOC: MOV     #1, $OFILL      ;;SET THE ZERO FILL SWITCH
(1) 007074 112737 000006 007267  MOV     #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
(1) 007102 112737 000005 007264  $TYPON: MOV     #5, $OCNT      ;;SET THE ITERATION COUNT
(1) 007110 010346          MOV     R3, -(SP)          ;;SAVE R3
(1) 007112 010446          MOV     R4, -(SP)          ;;SAVE R4
(1) 007114 010546          MOV     R5, -(SP)          ;;SAVE R5
(1) 007116 113704 007267          MOV     $OMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 007122 005404          NEG     R4
(1) 007124 062704 000006          ADD     #6, R4          ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 007130 110437 007266          MOV     R4, $OMODE      ;;SAVE IT FOR USE
(1) 007134 113704 007265          MOV     $OFILL, R4      ;;GET THE ZERO FILL SWITCH
(1) 007140 016605 000012          MOV     12(SP), R5      ;;PICKUP THE INPUT NUMBER
(1) 007144 005003          CLR     R3          ;;CLEAR THE OUTPUT WORD
(1) 007146 006105          1$: ROL     R5          ;;ROTATE MSB INTO 'C'
(1) 007150 000404          BR     3$          ;;GO DO MSB
(1) 007152 006105          2$: ROL     R5          ;;FORM THIS DIGIT
(1) 007154 006105          ROL     R5
(1) 007156 006105          ROL     R5
(1) 007160 010503          MOV     R5, R3
(1) 007162 006103          3$: ROL     R3          ;;GET LSB OF THIS DIGIT
(1) 007164 105337 007266          DECB   $OMODE          ;;TYPE THIS DIGIT?
(1) 007170 100016          BPL     7$          ;;BR IF NO
(1) 007172 042703 177770          BIC     #177770, R3      ;;GET RID OF JUNK
(1) 007176 001002          BNE     4$          ;;TEST FOR 0

```

```
(1) 007200 005704          TST      R4          ;;SUPPRESS THIS 0?
(1) 007202 001403          BEQ      5$          ;;BR IF YES
(1) 007204 005204          4$: INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
(1) 007206 052703 000060   BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
(1) 007212 052703 000040   5$: BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
(1) 007216 110337 007262   MOV      R3,8$       ;;SAVE FOR TYPING
(1) 007222 104401 007262   TYPE     .8$         ;;GO TYPE THIS DIGIT
(1) 007226 105337 007264   7$: DECB    $OCNT      ;;COUNT BY 1
(1) 007232 003347          BGT      2$          ;;BR IF MORE TO DO
(1) 007234 002402          BLT      6$          ;;BR IF DONE
(1) 007236 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 007240 000744          BR       2$          ;;GO DO THE LAST DIGIT
(1) 007242 012605          6$: MOV      (SP)+,R5   ;;RESTORE R5
(1) 007244 012604          MOV      (SP)+,R4   ;;RESTORE R4
(1) 007246 012603          MOV      (SP)+,R3   ;;RESTORE R3
(1) 007250 016666 000002 000004  MOV      2(SP),4(SP) ;;SET THE STACK FOR RETURNING
(1) 007256 012616          MOV      (SP)+,(SP)
(1) 007260 000002          RTI              ;;RETURN
(1) 007262 000          8$: .BYTE    0          ;;STORAGE FOR ASCII DIGIT
(1) 007263 000          .BYTE    0          ;;TERMINATOR FOR TYPE ROUTINE
(1) 007264 000          $OCNT: .BYTE 0          ;;OCTAL DIGIT COUNTER
(1) 007265 000          $OFILL: .BYTE 0        ;;ZERO FILL SWITCH
(1) 007266 000000          $OMODE: .WORD 0       ;;NUMBER OF DIGITS TO TYPE
523 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
```

```
(1)
(2)
(1) *****
(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) $TYPDS:
(3) 007270 010046          MOV      R0,-(SP)     ;;PUSH R0 ON STACK
(3) 007272 010146          MOV      R1,-(SP)     ;;PUSH R1 ON STACK
(3) 007274 010246          MOV      R2,-(SP)     ;;PUSH R2 ON STACK
(3) 007276 010346          MOV      R3,-(SP)     ;;PUSH R3 ON STACK
(3) 007300 010546          MOV      R5,-(SP)     ;;PUSH R5 ON STACK
(1) 007302 012746 020200   MOV      #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
(1) 007306 016605 000020   MOV      20(SP),R5    ;;GET THE INPUT NUMBER
(1) 007312 100004          BPL      1$          ;;BR IF INPUT IS POS.
(1) 007314 005405          NEG      R5          ;;MAKE THE BINARY NUMBER POS.
(1) 007316 112766 000055 000001  MOV      #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
(1) 007324 005000          1$: CLR      R0          ;;ZERO THE CONSTANTS INDEX
(1) 007326 012703 007504   MOV      #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
(1) 007332 112723 000040   MOV      #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
(1) 007336 005002          2$: CLR      R2          ;;CLEAR THE BCD NUMBER
(1) 007340 016001 007474   MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
(1) 007344 160105          3$: SUB      R1,R5     ;;FORM THIS BCD DIGIT
(1) 007346 002402          BLT      4$          ;;BR IF DONE
(1) 007350 005202          INC      R2          ;;INCREASE THE BCD DIGIT BY 1
```

```

(1) 007352 000774          BR           3$
(1) 007354 060105          4$: ADD       R1,R5           ;;ADD BACK THE CONSTANT
(1) 007356 005702          TST        R2             ;;CHECK IF BCD DIGIT=0
(1) 007360 001002          BNE        5$             ;;FALL THROUGH IF 0
(1) 007362 105716          TSTB       (SP)           ;;STILL DOING LEADING 0'S?
(1) 007364 100407          BMI        7$             ;;BR IF YES
(1) 007366 106316          5$: ASLB       (SP)           ;;MSD?
(1) 007370 103003          BCC        6$             ;;BR IF NO
(1) 007372 116663 000001 177777 MOVB       1(SP),-1(R3)    ;;YES--SET THE SIGN
(1) 007400 052702 000060          6$: BIS        #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1) 007404 052702 000040          7$: BIS        #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 007410 110223          MOVB       R2,(R3)+       ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 007412 005720          TST        (R0)+         ;;JUST INCREMENTING
(1) 007414 020027 000010          CMP        R0,#10        ;;CHECK THE TABLE INDEX
(1) 007420 002746          BLT        2$             ;;GO DO THE NEXT DIGIT
(1) 007422 003002          BGT        8$             ;;GO TO EXIT
(1) 007424 010502          MOV        R5,R2         ;;GET THE LSD
(1) 007426 000764          BR         6$             ;;GO CHANGE TO ASCII
(1) 007430 105726          8$: TSTB       (SP)+       ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 007432 100003          BPL        9$             ;;BR IF NO
(1) 007434 116663 177777 177776 MOVB       -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
(1) 007442 105013          9$: CLRB       (R3)         ;;SET THE TERMINATOR
(3) 007444 012605          MOV        (SP)+,R5      ;;POP STACK INTO R5
(3) 007446 012603          MOV        (SP)+,R3      ;;POP STACK INTO R3
(3) 007450 012602          MOV        (SP)+,R2      ;;POP STACK INTO R2
(3) 007452 012601          MOV        (SP)+,R1      ;;POP STACK INTO R1
(3) 007454 012600          MOV        (SP)+,R0      ;;POP STACK INTO R0
(1) 007456 104401 007504          TYPE      ,SDBLK         ;;NOW TYPE THE NUMBER
(1) 007462 016666 000002 000004 MOV        2(SP),4(SP)    ;;ADJUST THE STACK
(1) 007470 012616          MOV        (SP)+,(SP)
(1) 007472 000002          RTI                          ;;RETURN TO USER
(1) 007474 023420          $DTBL:  10000.
(1) 007476 001750          1000.
(1) 007500 000144          100.
(1) 007502 000012          10.
(1) 007504 000004          $DBLK:  .BLKW 4
524 .SBTTL BINARY TO ASCII AND TYPE ROUTINE

```

```

(1)
(2)
(1) ;;*****
(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) $TYPBN: MOV   R1,-(SP)      ;;SAVE R1 ON THE STACK
(1) 007516 016601 000006 MOV   6(SP),R1        ;;GET THE INPUT NUMBER
(1) 007522 000261 SEC                      ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1) 007524 112737 000060 007566 1$: MOVB #'0,$BIN        ;;SET CHARACTER TO AN ASCII '0'.
(1) 007532 006101 ROL   R1               ;;GET THIS BIT
(1) 007534 001406 BEQ   2$               ;;DONE?
(1) 007536 105537 007566 ADCB  $BIN          ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1) 007542 104401 007566 TYPE  ,SBIN          ;;GO TYPE THIS BIT
(1) 007546 000241 CLC                      ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS

```

```

(1) 007550 000765          BR      1$          ;;GO DO THE NEXT BIT
(1) 007552 012601          MOV     (SP)+,R1      ;;POP THE STACK INTO R1
(1) 007554 016666 000002 000004 2$:    MOV     2(SP),4(SP)   ;;ADJUST THE STACK
(1) 007562 012616          MOV     (SP)+,(SP)
(1) 007564 000002          RTI                    ;;RETURN TO USER
(1) 007566 000          000          $BIN:  .BYTE 0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
525          .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)
(1) 007570 112737 000001 010034 $ATY1: MOVB  #1,$FFLG  ;;TO REPORT FATAL ERROR
(1) 007576 112737 000001 010032 $ATY3: MOVB  #1,$MFLG  ;;TO TYPE A MESSAGE
(1) 007604 000403          BR      $ATYC
(1) 007606 112737 000001 010034 $ATY4: MOVB  #1,$FFLG  ;;TO ONLY REPORT FATAL ERROR
(2) 007614
(3) 007614 C10046          MOV     R0,-(SP)      ;;PUSH R0 ON STACK
(3) 007616 010146          MOV     R1,-(SP)      ;;PUSH R1 ON STACK
(1) 007620 105737 010032          TSTB   $MFLG        ;;SHOULD TYPE A MESSAGE?
(1) 007624 001450          BEQ    5$           ;;IF NOT: BR
(1) 007626 122737 000001 001204 CMPB   #APTENV,$ENV  ;;OPERATING UNDER APT?
(1) 007634 001031          BNE    3$           ;;IF NOT: BR
(1) 007636 132737 000100 001205 BITB   #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 007644 001425          BEQ    3$           ;;IF NOT: BR
(1) 007646 017600 000004          MOV     @4(SP),R0     ;;GET MESSAGE ADDR.
(1) 007652 062766 000002 000004 ADD     #2,4(SP)       ;;BUMP RETURN ADDR.
(1) 007660 005737 001164          1$:    TST     $MSGTYPE  ;;SEE IF DONE W/ LAST XMISSION?
(1) 007664 001375          BNE    1$           ;;IF NOT: WAIT
(1) 007666 010037 001200          MOV     R0,$MSGAD     ;;PUT ADDR IN MAILBOX
(1) 007672 105720          2$:    TSTB   (R0)+      ;;FIND END OF MESSAGE
(1) 007674 001376          BNE    2$
(1) 007676 163700 001200          SUB     $MSGAD,R0     ;;SUB START OF MESSAGE
(1) 007702 005200          ASR    R0            ;;GET MESSAGE LGTH IN WORDS
(1) 007704 010037 001202          MOV     R0,$SMSGLGT   ;;PUT LENGTH IN MAILBOX
(1) 007710 012737 000004 001164 MOV     #4,$MSGTYPE   ;;TELL APT TO TAKE MSG.
(1) 007716 000413          BR      5$
(1) 007720 017637 000004 007744 3$:    MOV     @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
(1) 007726 062766 000002 000004 ADD     #2,4(SP)       ;;BUMP RETURN ADDRESS
(3) 007734 013746 177776          MOV     177776,-(SP) ;;PUSH 177776 ON STACK
(1) 007740 004737 006560          JSR    PC,$TYPE      ;;CALL TYPE MACRO
(1) 007744 000000          4$:    .WORD 0
(1) 007746          5$:
(1) 007746 105737 010034          10$:   TSTB   $FFLG        ;;SHOULD REPORT FATAL ERROR?
(1) 007752 001416          BEQ    12$          ;;IF NOT: BR
(1) 007754 005737 001204          TST     $ENV          ;;RUNNING UNDER APT?
(1) 007760 001413          BEQ    12$          ;;IF NOT: BR
(1) 007762 005737 001164          11$:   TST     $MSGTYPE     ;;FINISHED LAST MESSAGE?
(1) 007766 001375          BNE    11$          ;;IF NOT: WAIT
(1) 007770 017637 000004 001166 MOV     @4(SP),$FATAL ;;GET ERROR #
(1) 007776 062766 000002 000004 ADD     #2,4(SP)       ;;BUMP RETURN ADDR.
(1) 010004 005237 001164          INC     $MSGTYPE     ;;TELL APT TO TAKE ERROR
(1) 010010 105037 010034          12$:   CLRB   $FFLG        ;;CLEAR FATAL FLAG
(1) 010014 105037 010033          CLRB   $LFLG        ;;CLEAR LOG FLAG
(1) 010020 105037 010032          CLRB   $MFLG        ;;CLEAR MESSAGE FLAG
(3) 010024 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
(3) 010026 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0

```

(1)	010030	000207	RTS	PC	::RETURN
(1)	010032	000	\$MFLG: .BYTE	0	::MESSG. FLAG
(1)	010033	000	\$LFLG: .BYTE	0	::LOG FLAG
(1)	010034	000	\$FFLG: .BYTE	0	::FATAL FLAG
(1)		010036	.EVEN		
(1)		000200	APTSIZE=200		
(1)		000001	APTENV=001		
(1)		000100	APTSPOOL=100		
(1)		000040	APTCSUP=040		

```

527          .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) 010036 010046          $TRAP: MOV     RO,-(SP)          ;;SAVE RO
(1) 010040 016600 000002  MOV     2(SP),RO          ;;GET TRAP ADDRESS
(1) 010044 005740          TST     -(RO)            ;;BACKUP BY 2
(1) 010046 111000          MOVB   (RO),RO          ;;GET RIGHT BYTE OF TRAP
(1) 010050 006300          ASL    RO               ;;POSITION FOR INDEXING
(1) 010052 016000 010072  MOV     $TRPAD(RO),RO    ;;INDEX TO TABLE
(1) 010056 000200          RTS    RO               ;;GO TO ROUTINE
(1)
(1)          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1) 010060 011646          $TRAP2: MOV    (SP),-(SP) ;;MOVE THE PC DOWN
(1) 010062 016666 000004 000002  MOV    4(SP),2(SP)      ;;MOVE THE PSW DOWN
(1) 010070 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)          ;
(3)          ; ROUTINE
(3)          ;-----
(3) 010072 010060          $TRPAD: .WORD  $TRAP2
(3) 010074 006560          $TYPE  ;;CALL TYPE     TRAP+1(104401) TTY TYPEOUT ROUTINE
(3) 010076 007066          $TYPOC ;;CALL TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 010100 007042          $TYPOS ;;CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 010102 007102          $TYPON ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 010104 007270          $TYPDS ;;CALL=TYPDS    TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(3) 010106 007514          $TYPBN ;;CALL=TYPBN     TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
(1)
(1)
(3) 010110 006126          $RDCHR ;;CALL=RDCHR    TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
(3) 010112 006216          $RDLIN ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
528
529          ;MEMORY USAGE MAP BLOCK
530 010114 000100          ADRPOK: .BLKW  100
531 010314 000000          ADRTOP: 0                ;LAST LOCATION USED BY PROGRAM
532
533          .END
    
```


AABASE	001262	55#	90*	91*	148			
AACNT	001274	63#	98*	152	182			
AAOFF	001250	47#	91					
AAPRI	004054	150	181	475#				
ABASE =	171000	14#	42					
ACDW1 =	000000	42						
ACDW2 =	000000	42						
ACPUOP=	000000	42						
ADBASE	001256	53#	87*	88	90	92	94	131
ADCNT	001270	61#	96*	135	174			
ADDW0 =	000000	42						
ADDW1 =	000000	42						
ADDW10=	000000	42						
ADDW11=	000000	42						
ADDW12=	000000	42						
ADDW13=	000000	42						
ADDW14=	000000	42						
ADDW15=	000000	42						
ADDW2 =	000000	42						
ADDW3 =	000000	42						
ADDW4 =	000000	42						
ADDW5 =	000000	42						
ADDW6 =	000000	42						
ADDW7 =	000000	42						
ADDW8 =	000000	42						
ADDW9 =	000000	42						
ADEVCT=	000000	42						
ADEVN =	000000	42						
ADGO	003456	130	328	391#				
ADPRI	004030	133	173	473#				
ADRPOK	010114	72	213	230	530#			
ADRTOP	010314	227	240	531#				
AENV =	000000	42						
AENVN =	000000	42						
AFATAL=	000000	42						
AMADR1=	000000	42						
AMADR2=	000000	42						
AMADR3=	000000	42						
AMADR4=	000000	42						
AMAMS1=	000000	42						
AMAMS2=	000000	42						
AMAMS3=	000000	42						
AMAMS4=	000000	42						
AMSGAD=	000000	42						
AMSGLG=	000000	42						
AMSGTY=	000000	42						
AMTYP1=	000000	42						
AMTYP2=	000000	42						
AMTYP3=	000000	42						
AMTYP4=	000000	42						
APASS =	000000	42						
APRIOR=	000000	42						
APTCSU=	000040	520	525#					
APTENV=	000001	520	525#					

APTSIZ=	000200	81	525#		
APTSPO=	000100	520	525#		
ASWREG=	000000	42			
ATESTN=	000000	42			
ATPRI	004147	288	480#		
AUNIT =	000000	42			
AUSWR =	000000	42			
AVECT1=	000000	42			
AVECT2=	000000	42			
BEGIN	001324	37	80#	519	
BIT0 =	000001	13#	419		
BIT00 =	000001	13#			
BIT01 =	000002	13#			
BIT02 =	000004	13#			
BIT03 =	000010	13#			
BIT04 =	000020	13#			
BIT05 =	000040	13#			
BIT06 =	000100	13#			
BIT07 =	000200	13#			
BIT08 =	000400	13#			
BIT09 =	001000	13#			
BIT1 =	000002	13#	468		
BIT10 =	002000	13#			
BIT11 =	004000	13#			
BIT12 =	010000	13#			
BIT13 =	020000	13#	456		
BIT14 =	040000	13#			
BIT15 =	100000	13#	211	226	
BIT2 =	000004	13#			
BIT3 =	000010	13#	466		
BIT4 =	000020	13#			
BIT5 =	000040	13#			
BIT6 =	000100	13#	420		
BIT7 =	000200	13#			
BIT8 =	000400	13#	402		
BIT9 =	001000	13#			
BPTVEC=	000014	13#			
CHAIN	000042	33#	105		
CR =	000015	13#	520		
CRLF =	000200	13#	102	520	
DDISP =	177570	13#	42	81	
DIBASE	001264	56#	92*	93*	156
DICNT	001276	64#	99*	160	186
DIGO	003534	155	408#		
DIOFF	001252	48#	93		
DIPRI	004066	158	185	476#	
DISPLA	001142	42#	81*		
DISPRE	001322	76#	81		
DOBASE	001266	57#	94*	95*	164
DOCNT	001300	65#	100*	168	190
DOGO	003560	163	415#		
DOOFF	001254	49#	95		
DOPRI	004100	166	189	477#	
DSWR =	177570	13#	42	81	

