

AAV11

AAV11 DIAGNOSTICS
CNAARAO

AH-T464A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



Microscopic image showing a grid of approximately 15 columns and 15 rows of small, dark, rectangular structures, likely representing individual cells or components of a diagnostic kit. The structures are arranged in a regular pattern and appear to be mounted on a light-colored background.

IDENTIFICATION

B 1

JEQ 0001

PRODUCT CODE: AC-T463A-MC
PRODUCT NAME: CNAAAA0 AAV11 DIAGNOSTICS
PRODUCT DATE: DECEMBER, 1982
MAINTAINER: DIAGNOSTIC SERVICES/ISS
AUTHOR: R.SHOOP

COPYRIGHT (C) 1982,1983
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	AAV11 BUS ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	MULTIPLE AAV11 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	EXTENDED UNITS
9.6	TESTER MODE
10.0	REVISION HISTORY
11.0	LISTING

1.0 ABSTRACT

 THE AAV11 DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11/21 COMPUTER OR LSI-11 PROCESSOR
2. DLV11 WITH I/O TYPE TERMINAL
3. AAV11 DAC OPTION

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE

-
1. ASSURE THAT THE LSI-11 IS IN THE ODT MICROCODE STATE.
 2. LOAD THE LOW OR HIGH SPEED READER WITH THE ABSOLUTE LOADER TAPE.
 3. TYPE THE READER'S CSR ADDRESS (177560-LOW OR 177550-HIGH) AND CHARACTER 'L'.
 4. AFTER TAPE IS LOADED, LOAD THE AAV11 BINARY TAPE INTO THE READER AND TYPE THE CHARACTER 'P'.
 5. IF THE ABSOLUTE LOADER HAS ALREADY BEEN LOADED (STEPS 2 & 3), THEN ONLY THE STARTING ADDRESS OF THE ABSOLUTE LOADER AND THE CHARACTER 'G' NEED BE TYPED (WITH THE AAV11 BINARY TAPE IN THE APPROPRIATE READER).

4.0 STARTING PROCEDURE

-
1. MAKE SURE THE DEVICE BUS ADDRESS AGREE WITH THE DEFAULT VALUES DEFINED IN SECTION 7.1. IF NOT, CHANGE LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' ODT COMMAND.
 2. INSURE THAT THE HALT SWITCH IS DISABLED (IF ANY).
 4. TYPE THE STARTING ADDRESS OF 200 AND THE CHARACTER G.
 5. THE PROGRAM WILL RESPOND BY TYPING THE PROGRAM TITLE.

4.1 PROGRAM START

200	STARTING ADDRESS OF THE LOGIC TEST <WITH UP TO FOUR AAV11'S>
204	STARTING ADDRESS OF THE RAMP LOOP
210	STARTING ADDRESS OF THE STATIC CALIBRATION
214	STARTING ADDRESS OF THE DYNAMIC CALIBRATION
230	STARTING ADDRESS OF THE LOGIC TEST <WITH UP TO SIXTEEN AAV11'S>
240	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
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 SOFTWARE SWITCH REGISTER 'SWREG' (LOC. 176) CAN BE CHANGED BY USING THE ODT FACILITIES.
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' 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

*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*BUSADR	AAV11 BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

*ALWAYS REPORTED

7.0 MISCELLANEOUS

7.1 AAV11 BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' (LOC. 1250) IF BASE BUS ADDRESS IS NOT 170440.

*NOTE: USE THE LSI-11 ODT FACILITIES TO MODIFY THIS LOCATIONS
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 AAV11 INTERFACE TESTING

THIS PROGRAM DOES "AUTO-SIZE" THE NUMBER OF AAV11'S CONNECTED.
THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 4 AAV11 INTERFACES,
WHEN STARTED AT 200 AND 16. WHEN STARTED AT ADDRESS 230,
WITH CONTIGUOUS BUS ADDRESSES. THE "AUTO-SIZE" CAN BE INHIBITED
BY THE OPERATOR SETTING BIT 15 OF LOCATION '\$ENV (LOC. 1214) AND
LOADING LOCATION '\$BASE' WITH THE ADDRESS OF THE ONE UNIT TO BE TESTED.

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 AAV11 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 LOGIC TESTS (SA 200)

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE AAV11 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. WHEN STARTED AT LOCATION 200, THE PROGRAM WILL AUTO-SIZE UP TO 4 AAV11'S TO BE TESTED.

9.2 RAMP LOOP (SA 204)

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'S ARE INTERCONNECTED.

9.3 STATIC CALIBRATION LOOP (SA 210)

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 DYNAMIC CALIBRATION LOOP (SA 214)

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 EXTENDED UNITS (SA 230)

SAME FUNCTION AS LOGIC TEST BUT ON 16. AAV11'S

9.6 TESTER SUPPORT (SA 240)

INITIALLY PERFORMS THE LOGIC TESTS AND THEN EMPLOYS A KNOWN GOOD A TO D CONVERTER TO AID IN ADJUSTING THE POT'S ON THE AAV11 BOARD. THE OPERATOR IS INFORMED AS TO WHICH POT TO ADJUST AND WHICH D TO A CONVERTER IS TESTED.

10.0 REVISION HISTORY

DVAAA-A DIAGNOSTIC WAS MODIFIED TO RUN ON 11/21 PROCESSOR BY LOWERING PRIORITY 7 TO 6 AND ALSO ADDING A MACRO CALL .INIT TO CNMAC2.SML LIBRARY AND THE NAME WAS CHANGED TO CNAAA-A0.

11.0 LISTING

```
5687 .TITLE MAINDEC-11-CNAAA-A AAV11 DIAGNOSTIC
(1) .*COPYRIGHT (C) 1982
(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) .*
5688 .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) .***** THE FOLLOWING ODT START ADDRESS FOR SBC 11/21 IS ADDED
(1) 170000 ODTST= 170000
(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

;*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)	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) ;***** THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED
(1) 000100 LKVEC= 100 ::LINE CLOCK VECTOR
(1) 000140 BRKVEC= 140 ::BREAK VECTOR
(1) 000240 PIRQVEC=240 ::PROGRAM INTERRUPT REQUEST VECTOR
5689
5690 174440 ABASE=174440 ;REV:0
```

5692
 5693
 5694
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 5695
 (1)
 (1) 000000
 (1)
 (1)
 (1)
 (1) 000174
 (1) 000174 000000
 (1) 000176 000000
 (1)
 (1) 000200 000137 001450
 5696 000204 000137 006570
 5697 000210 000137 006656
 5698 000214 000137 006716
 5699
 5700
 5701 000230 000137 001440
 5702
 5703
 5704 000240 000137 001432
 5705
 5706
 5707 000100 000104 000200 000002

.SBTTL OPERATIONAL SWITCH SETTINGS

```

: *
: * SWITCH USE
: * -----
: * 15 HALT ON ERROR
: * 14 LOOP ON TEST
: * 13 INHIBIT ERROR TYPEOUTS
: * 11 INHIBIT ITERATIONS
: * 10 BELL ON ERROR
: * 9 LOOP ON ERROR
: * 8 LOOP ON TEST IN SWR<7:0>

```

.SBTTL TRAP CATCHER

```

.=0
: *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
: *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
: *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
.=174
DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
JMP FULRMP ;JUMP TO FULL RAMP LOOP
JMP STATIC ;JUMP TO STATIC DAC CALIBRATION
JMP DYNCAL ;JUMP TO DYNAMIC DAC CALIBRATION
.=230
JMP ADDOK ;JUMP AND ENABLE EXTENDED UJITS <16.>
.=240
JMP TESTER ;JUMP TO TESTER SA.
.=100
i04,200,2 ;B EVENT SAFE GUARD

```

5709
5710
(1)
(2)
(1)
(1) 000106
(1) 000046
(1) 000046 005542
(1) 000052 000052
(1) 000052 000000
(1) 000106
5711 001000
5712
(1)
(2)
(1)
(2)
(1) 001000
(1) 000024 000024
(1) 000024 000200
(1) 000044 000044
(1) 000044 001000
(1) 001000
(1) 001000 000000
(1) 001002 001174
(1) 001004 000030
(1) 001006 000010
(1) 001010 000030
(1) 001012 000031

.SBTTL ACT11 HOOKS

```
::*****  
:HOOKS REQUIRED BY ACT11  
$SVPC=      ;SAVE PC  
  =46  
$ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP  
  =52  
.WORD 0     ;;2)SET LOC.52 TO ZERO  
=$SVPC      ;; RESTORE PC  
=1000
```

.SBTTL APT PARAMETER BLOCK

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

```
$APTHD:  
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.  
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)  
$TSTM:  .WORD 30     ;;RUN TIM OF LONGEST TEST  
$PASTM: .WORD 10     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)  
$UNITM: .WORD 30     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT  
SETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

5713

.SBTTL COMMON TAGS

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

(1)		001100		.=1100		::START OF COMMON TAGS
(1)	001100	000000		\$CMTAG: .WORD	0	
(1)	001102	000		\$TSTNM: .BYTE	0	::CONTAINS THE TEST NUMBER
(1)	001103	000		\$ERFLG: .BYTE	0	::CONTAINS ERROR FLAG
(1)	001104	000000		\$ICNT: .WORD	0	::CONTAINS SUBTEST ITERATION COUNT
(1)	001106	000000		\$LPADR: .WORD	0	::CONTAINS SCOPE LOOP ADDRESS
(1)	001110	000000		\$LPERR: .WORD	0	::CONTAINS SCOPE RETURN FOR ERRORS
(1)	001112	000000		\$ERTTL: .WORD	0	::CONTAINS TOTAL ERRORS DETECTED
(1)	001114	000		\$ITEMB: .BYTE	0	::CONTAINS ITEM CONTROL BYTE
(1)	001115	001		\$ERMAX: .BYTE	1	::CONTAINS MAX. ERRORS PER TEST
(1)	001116	000000		\$ERRPC: .WORD	0	::CONTAINS PC OF LAST ERROR INSTRUCTION
(1)	001120	000000		\$GDADR: .WORD	0	::CONTAINS ADDRESS OF 'GOOD' DATA
(1)	001122	000000		\$BDADR: .WORD	0	::CONTAINS ADDRESS OF 'BAD' DATA
(1)	001124	000000		\$GDDAT: .WORD	0	::CONTAINS 'GOOD' DATA
(1)	001126	000000		\$BDDAT: .WORD	0	::CONTAINS 'BAD' DATA
(1)	001130	000000		.WORD	0	::RESERVED--NOT TO BE USED
(1)	001132	000000		.WORD	0	
(1)	001134	000		\$AUTOB: .BYTE	0	::AUTOMATIC MODE INDICATOR
(1)	001135	000		\$INTAG: .BYTE	0	::INTERRUPT MODE INDICATOR
(1)	001136	00000		.WORD	0	
(1)	001140	177560		\$SWR: .WORD	DSWR	::ADDRESS OF SWITCH REGISTER
(1)	001142	177560		\$DISPLAY: .WORD	DDISP	::ADDRESS OF DISPLAY REGISTER
(1)	001144	177560		\$TKS: 177560		::TTY KBD STATUS
(1)	001146	177562		\$TKB: 177562		::TTY KBD BUFFER
(1)	001150	177564		\$TPS: 177564		::TTY PRINTER STATUS REG. ADDRESS
(1)	001152	177566		\$TPB: 177566		::TTY PRINTER BUFFER REG. ADDRESS
(1)	001154	000		\$NULL: .BYTE	0	::CONTAINS NULL CHARACTER FOR FILLS
(1)	001155	002		\$FILLS: .BYTE	2	
(1)						::CONTAINS # OF FILLER CHARACTERS REQUIRED
(1)	001156	012		\$FILLC: .BYTE	12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
(1)	001157	000		\$TPFLG: .BYTE	0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(1)	001160	000000		\$TIMES: 0		::MAX. NUMBER OF ITERATIONS
(1)	001162	000000		\$ESCAPE: 0		::ESCAPE ON ERROR ADDRESS
(1)	001164	177607	000377	\$BELL: .ASCIZ	<207><377><377>	::CODE FOR BELL
(1)	001170	077		\$QUES: .ASCII	/?/	::QUESTION MARK
(1)	001171	015		\$CRLF: .ASCII	<15>	::CARRIAGE RETURN
(1)	001172	000012		\$LF: .ASCIZ	<12>	::LINE FEED

.SBTTL APT MAILBOX-ETABLE

.EVEN
\$MAIL: ::APT MAILBOX
\$MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ::TEST NUMBER
\$PASS: .WORD APASS ::PASS COUNT
\$DEVCT: .WORD ADEVCT ::DEVICE COUNT
\$UNIT: .WORD AUNIT ::I/O UNIT NUMBER

(2)	001210	000000	\$MSGAD: .WORD	AMSGAD	::MESSAGE ADDRESS
(2)	001212	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
(2)	001214		\$ETABLE:		::APT ENVIRONMENT TABLE
(2)	001214	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
(2)	001215	000	\$ENVM: .BYTE	AENVM	
(2)			::ENVIRONMENT	MODE BITS	
(2)	001216	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
(2)	001220	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
(2)	001222	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)	001224	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001225	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)	001226	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			::*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2)	001230	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001231	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
(2)	001232	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001234	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001235	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
(2)	001236	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001240	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001241	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
(2)	001242	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001244	000000	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001246	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001250	174440	\$BASE: .WORD	ABASE	
(2)			::BASE	ADDRESS OF EQUIPMENT UNDER TEST	
(2)	001252	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
(2)	001254	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001256		\$ETEND:		
(2)			.MEXIT		

(1)			.SBTTL	ERROR POINTER TABLE	
(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)			;	* 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)			\$ERRTB:		
(1)	001256		;	ITEM 1	
5714				EM1	:BUT TIME-OUT WHEN REF. A DAC ADDRESS
5716				DH2	:ERRPC BUSADR
5717				DT1	:SERRPC \$BDDAT
5718	001256	007122		DF0	
5719	001260	010322	;	ITEM 2	
5720	001262	011112		EM2	:DAC #0 REGISTER IN ERROR
5721	001264	011204		DH1	:ERRPC BUSADR GOOD BAD
5722				DT2	:SERRPC DAC0 \$GDDAT \$BDDAT
5723				DF0	
5724	001266	007176	;	ITEM 3	
5725	001270	010267		EM3	:DAC #1 REGISTER IN ERROR
5726	001272	011120		DH1	:ERRPC BUSADR GOOD BAD
5727	001274	011204		DT3	:SERRPC DAC1 \$GDDAT \$BDDAT
5728				DF0	
5729			;	ITEM 4	
5730	001276	007225		EM4	:DAC #2 REGISTER IN ERROR
5731	001300	010267		DH1	:ERRPC BUSADR GOOD BAD
5732	001302	011132		DT4	:SERRPC DAC2 \$GDDAT \$BDDAT
5733	001304	011204		DF0	
5734			;	ITEM 5	
5735				EM5	:DAC #3 REGISTER IN ERROR
5736	001306	007254		DH1	:ERRPC BUSADR GOOD BAD
5737	001310	010267		DT5	:SERRPC DAC3 \$GDDAT \$BDDAT
5738	001312	011144		DF0	
5739	001314	011204	;	ITEM 6	
5740				EM6	:SELECTED DAC OFFSET POT IS NOT ADJUSTED CORRECTLY
5741				DH6	:ERRPC BUSADR EXPECT WAS SPREAD
5742	001316	007303		DT6	:SERRPC DACBAD \$GDDAT \$BDDAT SPREAD
5743	001320	010267		DF0	
5744	001322	011156			
5745	001324	011204			
5746					
5747					
5748	001326	007332			
5749	001330	010337			
5750	001332	011170			
5751	001334	011204			
5752					

5754					
5755			:ITEM	7	
5756	001336	007413		EM7	:SELECTED DAC GAIN POT IS NOT ADJUSTED CORRECTLY
5757	001340	010337		DH6	:ERRPC BUSADR EXPECT WAS SPREAD
5758	001342	011170		DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
5759	001344	011204		DF0	
5760					
5761			:ITEM	10	
5762	001346	007472		EM10	:SELECTED DAC HAS A LINEARITY PROBLEM
5763	001350	010337		DH6	:ERRPC BUSADR EXPECT WAS SPREAD
5764	001352	011170		DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
5765	001354	011204		DF0	
5766					
5767			:ITEM	11	
5768	001356	007537		EM11	:+15 VOLT SUPPLY IS INCORRECT
5769	001360	010337		DH6	:ERRPC BUSADR EXPECT WAS SPREAD
5770	001362	011170		DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
5771	001364	011204		DF0	
5772					
5773			:ITEM	12	
5774	001366	007574		EM12	:-15 VOLT SUPPLY IS INCORRECT
5775	001370	010337		DH6	:ERRPC BUSADR EXPECT WAS SPREAD
5776	001372	011170		DT6	:\$ERRPC DACBAD \$GDDAT \$BDDAT SPREAD
5777	001374	011204		DF0	
5778					
5779			:ITEM	13	
5780	001376	007631		EM13	:DAC #3 DIGITAL OUTPUT BITS IN ERROR
5781	001400	010267		DH1	:ERRPC BUSADR GOOD BAD
5782	001402	011156		DT5	:\$ERRPC DAC3 \$GDDAT \$BDDAT
5783	001404	011204		DF0	
5784					
5785			:ITEM	14	
5786	001406	007675		EM14	:WAKE UP OPERATOR AND ADJUST THE POT
5787	001410	000000		0	
5788	001412	000000		0	
5789	001414	000000		0	
5790					
5791	001416	000010	VADDR:	10	:OFFSET TO NEXT AAV11 ADDRESS
5792	001420	000000	EVER:	0	
5793	001422	174440	DACO:	ABASE	
5794	001424	174442	DAC1:	ABASE+2	
5795	001426	174444	DAC2:	ABASE+4	
5796	001430	174446	DAC3:	ABASE+6	


```

5798
5827 001432 005237 007020      TESTER: INC      WFTST           ;INDICATE TESTER MODE
5828 001436 000411              BR      BEGIN1
5829 001440 012737 000021 007006 ADDOK: MOV      #17,NUMBOK      ;LOAD 16 MAX UNITS
5830 001446 000403              BR      BEGINA
5831 001450 012737 000005 007006 BEGIN: MOV      #5,NUMBOK      ;LOAD 4 MAX UNITS
5832 001456 005037 007020      BEGINA: CLR     WFTST
5833 001462 005037 007012      BEGIN1: CLR    TEMP
5834 001466 005037 001420              CLR    EVER
5835 001472 000005              RESET
5837
(1)      .SBTTL INITIALIZE THE COMMON TAGS
(1)      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001474 012706 001100      MOV      #CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 001500 005026              CLR      (R6)+          ;;CLEAR MEMORY LOCATION
(1) 001502 022706 001140      CMP      #SWR,R6      ;;DONE?
(1) 001506 001374              BNE     #-6            ;;LOOP BACK IF NO
(1) 001510 012706 001100      MOV      #STACK,SP     ;;SETUP THE STACK POINTER
(1)      ;;INITIALIZE A FEW VECTORS
(1) 001514 012737 011524 000020      MOV      #SSCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001522 012737 000300 000022      MOV      #PR6,@#IOTVEC+2 ;;LEVEL 6
(1) 001530 012737 012006 000030      MOV      #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001536 012737 000300 000032      MOV      #PR6,@#EMTVEC+2 ;;LEVEL 6
(1)      ;;BIT02
(1) 001544 012737 014412 000034      MOV      #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001552 012737 000300 000036      MOV      #PR6,@#TRAPVEC+2;LEVEL 6
(1) 001560 012737 012336 000024      MOV      #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 001566 012737 000300 000026      MOV      #PR6,@#PWRVEC+2 ;;LEVEL 6
(1) 001574 005037 001160      CLR     $TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001600 005037 001162      CLR     $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001604 112737 000001 001115      MOV     #1,$ERMAX      ;;ALLOW ONE ERROR PER TEST
(1) 001612 012737 001612 001106      MOV     #,$SLPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001620 012737 001620 001110      MOV     #,$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) 001626 013746 000004              MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
(2) 001632 012737 001666 000004      MOV     #64$,@#ERRVEC  ;;SET UP ERROR VECTOR
(2) 001640 012737 177570 001140      MOV     #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001646 012737 177570 001142      MOV     #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001654 022777 177777 177256      CMP     #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
(2) 001662 001012              BNE     66$           ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)      ;;AND THE HARDWARE SWR IS NOT = -1
(2) 001664 000403              BR      65$           ;;BRANCH IF NO TIMEOUT
(2) 001666 012716 001674      64$: MOV     #65$,(SP)    ;;SET UP FOR TRAP RETURN
(2) 001672 000002              RTI
(2) 001674 012737 000176 001140      65$: MOV     #SWREG,SWR   ;;POINT TO SOFTWARE SWR
(2) 001702 012737 000174 001142      MOV     #DISPREG,DISPLAY
(2) 001710 012637 000004      66$: MOV     (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 001714 005037 001202              CLR     $PASS          ;;CLEAR PASS COUNT
(2) 001720 132737 000200 001215      BITB   #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
(2) 001726 001403              BEQ    67$            ;;YES,USE NON-APT SWITCH
(2) 001730 012737 001216 001140      MOV     #SSWREG,SWR   ;;NO,USE APT SWITCH REGISTER
(2) 001736
5838 001736 005037 007004      67$: CLR     BADUNT      ;RESET BAD INDICATOR
5839 001742 000137 002044      JMP    INIT1

```

```
5841  
5842  
5843  
5844 001746 012702 000252  
5845 001752 012701 000250  
5846 001756 010221  
5847 001760 005021  
5848 001762 010102  
5849 001764 005722  
5850 001766 020227 001002  
5851 001772 001371  
5852  
5853  
5854  
5855 001774 013700 001250  
5856 002000 010037 001422  
5857 002004 010037 001424  
5858 002010 010037 001426  
5859 002014 010037 001430  
5860 002020 062737 000002 001424  
5861 002026 062737 000004 001426  
5862 002034 062737 000006 001430  
5863 002042 000207  
5870  
5871 002044 004737 001746  
5872 002050 005737 007012  
5873 002054 001012  
5874 002056 005737 000042  
5875 002062 001007  
5876 002064 005737 007020  
5877 002070 001402  
5878 002072 104401  
5879 002074 010011  
5880 002076 104401  
5881 002100 007040
```

```
      ;SUBROUTINE TO LOAD A TRAP CATCHER  
LDTRAP: MOV      #252,R2          ;LOAD R2  
        MOV      #250,R1          ;LOAD R1  
5$:     MOV      R2,(R1)+         ;LOAD .+2  
        CLR      (R1)+           ;LOAD HALT  
        MOV      R1,R2           ;LOAD R2  
        TST      (R2)+           ;BUMP R2  
        CMP      R2,#1002        ;TEST FOR LAST  
        BNE      5$              ;BR UNTIL DONE  
  
      ;AND LOAD DEVICE ADDRESSES LOCATIONS  
        MOV      $BASE,R0        ;GET BASE ADDRESS  
        MOV      R0,DAC0         ;LOAD X ADDRESS  
        MOV      R0,DAC1         ;LOAD Y ADDRESS  
        MOV      R0,DAC2         ;LOAD DAC #2  
        MOV      R0,DAC3         ;LOAD DAC #3  
        ADD      #2,DAC1  
        ADD      #4,DAC2  
        ADD      #6,DAC3  
        RTS      PC              ;EXIT  
  
INIT1: JSR      PC,LDTRAP  
        TST      TEMP            ;TEST IF START OR RESTART  
        BNE      MTEST          ;RESTART  
        TST      @#42           ;TEST IF MONITOR  
        BNE      MTEST          ;BR IF NOT  
        TST      WFTST          ;TEST IF ON TESTER  
        BEQ      1$             ;BR IF NOT  
        TYPE     MSGSW          ;TELL OPERATOR ABOUT TESTER SWITCHES  
1$:     TYPE     TYPE           ;CALL MESSAGE PRINTER VIA 'EMT'  
        TITLE    TITLE         ;TYPE PROGRAM HEADER.
```

```

5883
5884
5885
5886 002102 013737 001250 001126 MTEST: MOV $BASE,$BDDAT ;GET THE BASE ADDRESS
5887 002110 005037 007010 CLR MASKNM
5888 002114 005037 001206 CLR $UNIT ;CLEAR UNIT #
5889 002120 012737 002164 000004 MOV #2$,ERRVEC ;LOAD TRAP RETURN
5890 002126 005777 176774 1$: TST @BDDAT ;TEST IF ADDR EXISTS
5891 002132 063737 001416 001126 ADD VADDR,$BDDAT ;UPDATE THE BUS ADDRESS
5892 002140 005237 001206 INC $UNIT ;UPDATE UNIT COUNT
5893 002144 005737 001214 TST $ENV ;TEST IF "DO NOT SIZE"
5894 002150 100413 BMI 3$ ;BR IF NO SIZEING
5895 002152 023737 007006 001206 CMP NUMBOK,$UNIT ;TEST IF MAX. NUMBER
5896 002160 001362 BNE 1$ ;BR IF NOT
5897 002162 000406 BR 3$ ;:BR IF MAX.
5898 002164 022626 2$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
5899 002166 005737 001206 TST $UNIT ;TEST IF ANY EXIST
5900 002172 001002 BNE 3$ ;BR IF SOME ARE THERE
5901 002174 104001 ERROR 1 ;BASE ADDRESS CAUSED AN BUS TRAP
5902 002176 000443 BR TST1 ;:IS $BASE CORRECT??
5903 002200 005737 001420 3$: TST EVER ;TEST IF # HAS BEEN REPORTED
5904 002204 100422 BMI 4$ ;BR IF IT HAS
5905 002206 005737 007020 TST WFTST ;TEST IF TESTER MODE
5906 002212 001010 BNE 6$ ;BR IF TESTER
5907 002214 104401 TYPE
5908 002216 010224 FOUND1 ;TELL OPERATOR THE # OF AAV11'S
5909 002220 013746 001206 MOV $UNIT,-(SP)
5910 002224 104403 TYPOS
5911 002226 002 .BYTE 2
5912 002227 000 .BYTE 0
5913 002230 104401 TYPE
5914 002232 010250 FOUND2
5915 002234 013737 001206 001420 6$: MOV $UNIT,EVER ;SAVE THE # OF AAV11'S FOR LATER
5916 002242 052737 100000 001420 BIS #BIT15,EVER ;SET "REPORTED # FLAG"
5917 002250 000405 BR 5$ ;:
5918 002252 123737 001420 001206 4$: CMPB EVER,$UNIT ;TEST IF ANY HAVE GONE AWAY
5919 002260 001401 BEQ 5$ ;:BR IF ALL ARE STILL HERE
5920 002262 104013 ERROR 13 ;EXISTING UNIT FAILED TO RESPOND NOW
5921 002264 005037 001206 5$: CLR $UNIT ;RESET UNIT POINTER
5922 002270 005037 001204 CLR $DEVCT ;MAKE APT HAPPY
5923 002274 004737 001746 JSR PC,LDTRAP ;LOAD TRAP CATCHER AND BUS ADDRESSES
5924 002300 012737 000001 007010 MOV #BIT0,MASKNM ;LOAD MASK NUMBER IF ERROR
  
```

```
5926
5927
(3)
(3)
(2) 002306 000004
(2)
5928 002310 012737 002340 000004      MOV    #1$,ERRVEC      ;LOAD BUS TRAP RETURN
5929 002316 005777 177100                TST    @DAC0           ;TEST DAC #0
5930 002322 005777 177076                TST    @DAC1           ;TEST DAC #1
5931 002326 005777 177074                TST    @DAC2           ;TEST DAC #2
5932 002332 005777 177072                TST    @DAC3           ;TEST DAC #3
5933 002336 000407                        BR     2$              ;BR AND RESTORE LOC. 4
5934 002340 022626      1$:    CMP    (SP)+,(SP)+  ;CLEAN THE STACK
5935 002342 104001      ERROR  1              ;ERROR, BUS TIMEOUT WHEN ADDRESSING THE AAV11
5936 002344 012737 000006 000004      MOV    #6,ERRVEC      ;LOAD LOC 4
5937 002352 000137 004530                JMP    REMAIN          ;TEST IF ANY OTHER'S
5938 002356 012737 000006 000004      2$:    MOV    #6,ERRVEC      ;LOAD RETURN
5939
(3)
(3)
(2) 002364 000004
(2)
5940 002366 005037 001124                CLR    $GDDAT          ;LOAD EXPECTED
5941 002372 013777 001124 177022      MOV    $GDDAT,@DAC0   ;LOAD REG
5942 002400 017737 177016 001126      MOV    @DAC0,$BDDAT   ;READ REG
5943 002406 023737 001124 001126      CMP    $GDDAT,$BDDAT ;COMPARE
5944 002414 001401      BEQ    TST3           ;;BR IF EQUAL
5945 002416 104002      ERROR  2              ;ERROR, DACO REGISTER NOT = 0
5946
5947
(3)
(3)
(2) 002420 000004
(2)
5948 002422 012737 007777 001124      MOV    #7777,$GDDAT   ;LOAD EXPECTED
5949 002430 013777 001124 176764      MOV    $GDDAT,@DAC0   ;LOAD REG
5950 002436 017737 176760 001126      MOV    @DAC0,$BDDAT   ;READ REG
5951 002444 023737 001124 001126      CMP    $GDDAT,$BDDAT ;COMPARE
5952 002452 001401      BEQ    TST4           ;;BR IF EQUAL
5953 002454 104002      ERROR  2              ;ERROR, DACO REGISTER NOT = 7777
5954
5955
(3)
(3)
(2) 002456 000004
(2)
(1) 002460 012737 000100 001160      MOV    #100,$TIMES    ;;DO 100 ITERATIONS
5956 002466 012737 004000 001124      MOV    #BIT11,$GDDAT  ;LOAD EXPECTED
5957 002474 013777 001124 176720      1$:    MOV    $GDDAT,@DAC0 ;LOAD DACO REGISTER
5958 002502 017737 176714 001126      MOV    @DAC0,$BDDAT   ;READ THE REGISTER
5959 002510 023737 001124 001126      CMP    $GDDAT,$BDDAT ;COMPARE THE DATA
5960 002516 001401      BEQ    2$              ;;BR IF SAME
5961 002520 104002      ERROR  2              ;ERROR, DACO REGISTER FAILED TO HOLD A FLOATING
5962 002522 006237 001124      2$:    ASR    $GDDAT      ;CHANGE THE DATA
5963 002526 001362      BNE    1$              ;BR AND TEST MORE DATA
```

```

5965
5966      ::*****
(4)      : *TEST 5      TEST THAT DACO CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
(4)      : *****
(3) 002530 000004      TST5:  SCOPE
(3)
(2) 002532 012737 000100 001160      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
(1) 002540 012737 004000 001124      MOV      $BIT11,$GDDAT      :LOAD EXPECTED
(1) 002546 013777 001124 176646      MOV      $GDDAT,@DACO      :LOAD DACO
(1) 002554 017737 176642 001126      1$:  MOV      @DACO,$BDDAT      :READ THE REGISTER
(1) 002562 023737 001124 001126      CMP      $GDDAT,$BDDAT      :COMPARE THE GOOD TO DACO
(2) 002570 001407      BEQ      2$      ;;BR IF THE SAME
(1) 002572 017737 176624 001126      MOV      @DACO,$BDDAT      :SAVE FOR TYPEOUT
(1) 002600 104002      ERROR    2      :DACO FAILED TO HOLD A FLOATING 1 PATTERN
(1) 002602 013777 001124 176612      MOV      $GDDAT,@DACO      :LOAD DACO AGAIN
(1) 002610 006277 176606      2$:  ASR      @DACO      :CHANGE THE DATA
(1) 002614 006237 001124      ASR      $GDDAT      :CHANGE THE EXPECTED
(1) 002620 001355      BNE      1$      :BR IF MORE DATA
5967      : *****
(4)      : *TEST 6      TEST THE 'SUB' INSTRUCTION WORKS ON DACO
(4)      : *****
(3) 002622 000004      TST6:  SCOPE
(3)
(2) 002624 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
(1) 002632 012737 007777 001124      MOV      #7777,$GDDAT      :LOAD EXPECTED
(1) 002640 013777 001124 176554      MOV      $GDDAT,@DACO      :LOAD DACO
(1) 002646 162737 000001 001124      1$:  SUB      #1,$GDDAT      :SUB A VALUE
(1) 002654 162777 000001 176540      SUB      #1,@DACO      :FROM EXPECTED AND DACO
(1) 002662 017737 176534 001126      MOV      @DACO,$BDDAT      :READ THE REGISTER
(1) 002670 023737 001124 001126      CMP      $GDDAT,$BDDAT      :COMPARE
(2) 002676 001404      BEQ      2$      ;;BR IF SAME
(1) 002700 104002      ERROR    2      :THE SUB INSTRUCTION FAILED ON DACO
(1) 002702 013777 001124 176512      MOV      $GDDAT,@DACO      :LOAD THE REGISTER AGAIN
(1) 002710 005737 001124      2$:  TST      $GDDAT      :TEST FOR MORE DATA
(2) 002714 001354      BNE      1$      ;;BR IF MORE DATA
5968      : *****
(3)      : *TEST 7      TEST THAT DAC1 REGISTER CAN BE CLEARED
(3)      : *****
(2) 002716 000004      TST7:  SCOPE
(2)
5969 002720 005037 001124      CLR      $GDDAT      :LOAD EXPECTED
5970 002724 013777 001124 176472      MOV      $GDDAT,@DAC1      :LOAD DAC1
5971 002732 017737 176466 001126      MOV      @DAC1,$BDDAT      :READ REG
5972 002740 023737 001124 001126      CMP      $GDDAT,$BDDAT      :COMPARE
5973 002746 001401      BEQ      TST10      ;;BR IF EQUAL
5974 002750 104003      ERROR    3      :ERROR, DAC1 REGISTER NOT = 0
5975      : *****
(3)      : *TEST 10     TEST THAT DAC #1 REGISTER CAN BE LOADED WITH #7777
(3)      : *****
(2) 002752 000004      TST10: SCOPE
(2)
5976 002754 012737 007777 001124      MOV      #7777,$GDDAT      :LOAD EXPECTED
5977 002762 013777 001124 176434      MOV      $GDDAT,@DAC1      :LOAD DAC #1
5978 002770 017737 176430 001126      MOV      @DAC1,$BDDAT      :READ REG
5979 002776 023737 001124 001126      CMP      $GDDAT,$BDDAT      :COMPARE
5980 003004 001401      BEQ      TST11      ;;BR IF EQUAL

```

MAINDEC-11-CNAAA-A AAV11
CNAAA.P11 27-DEC-82 14:25

DIAGNOSTIC
T10

MACY11 30(1046) 27-DEC-82 14:27 PAGE 65-1
TEST THAT DAC #1 REGISTER CAN BE LOADED WITH #7777

SEQ 0021

5981 003006 104003

ERROR 3

;ERROR, DAC1 REGISTER NOT = 7777

```
5983
5984
(3)
(3)
(2) 003010 000004
(2)
(1) 003012 012737 000100 001160
5985 003020 012737 004000 001124
5986 003026 013777 001124 176370
5987 003034 017737 176364 001126
5988 003042 023737 001124 001126
5989 003050 001401
5990 003052 104003
5991 003054 006237 001124
5992 003060 001362
5993
(4)
(4)
(3) 003062 000004
(3)
(2) 003064 012737 000100 001160
(1) 003072 012737 004000 001124
(1) 003100 013777 001124 176316
(1) 003106 017737 176312 001126
(1) 003114 023737 001124 001126
(2) 003122 001407
(1) 003124 017737 176274 001126
(1) 003132 104003
(1) 003134 013777 001124 176262
(1) 003142 006277 176256
(1) 003146 006237 001124
(1) 003152 001355
5994
(4)
(4)
(3) 003154 000004
(3)
(2) 003156 012737 000010 001160
(1) 003164 012737 007777 001124
(1) 003172 013777 001124 176224
(1) 003200 162737 000001 001124
(1) 003206 162777 000001 176210
(1) 003214 017737 176204 001126
(1) 003222 023737 001124 001126
(2) 003230 001404
(1) 003232 104003
(1) 003234 013777 001124 176162
(1) 003242 005737 001124
(2) 003246 001354

*****
*TEST 11 TEST THAT DAC #1 REGISTER CAN HOLD A FLOATING 1 PATTERN
*****
TST11: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #BIT11,$GDDAT ;LOAD EXPECTED
1$: MOV $GDDAT,@DAC1 ;LOAD THE REGISTER
MOV @DAC1,$BDDAT ;READ THE REGISTER
CMP $GDDAT,$BDDAT ;COMPARE THE DATA
BEQ 2$ ;;BR IF DATA IS SAME
ERROR 3 ;ERROR, DAC #1 REGISTER FAILED TO HOLD A FLOATIN
2$: ASR $GDDAT ;CHANGE THE DATA
BNE 1$ ;BR AND TEST MORE DATA
*****
*TEST 12 TEST THAT DAC1 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
*****
TST12: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #BIT11,$GDDAT ;LOAD EXPECTED
1$: MOV $GDDAT,@DAC1 ;LOAD DAC1
MOV @DAC1,$BDDAT ;READ THE REGISTER
CMP $GDDAT,$BDDAT ;COMPARE THE GOOD TO DAC1
BEQ 2$ ;;BR IF THE SAME
MOV @DAC1,$BDDAT ;SAVE FOR TYPEOUT
ERROR 3 ;DAC1 FAILED TO HOLD A FLOATING 1 PATTERN
2$: MOV $GDDAT,@DAC1 ;LOAD DAC1 AGAIN
ASR @DAC1 ;CHANGE THE DATA
ASR $GDDAT ;CHANGE THE EXPECTED
BNE 1$ ;BR IF MORE DATA
*****
*TEST 13 TEST THE "SUB" INSTRUCTION WORKS ON DAC1
*****
TST13: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #7777,$GDDAT ;LOAD EXPECTED
1$: MOV $GDDAT,@DAC1 ;LOAD DAC1
SUB #1,$GDDAT ;SUB A VALUE
SUB #1,@DAC1 ;FROM EXPECTED AND DAC1
MOV @DAC1,$BDDAT ;READ THE REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ;;BR IF SAME
ERROR 3 ;THE SUB INSTRUCTION FAILED ON DAC1
2$: MOV $GDDAT,@DAC1 ;LOAD THE REGISTER AGAIN
TST $GDDAT ;TEST FOR MORE DATA
BNE 1$ ;BR IF MORE DATA
```

```
5996
5997
(3)
(3)
(2) 003250 000004
(2)
5998 003252 005037 001124
5999 003256 013777 001124 176142
6000 003264 017737 176136 001126
6001 003272 023737 001124 001126
6002 003300 001401
6003 003302 104004
6004
6005
(3)
(3)
(2) 003304 000004
(2)
6006 003306 012737 007777 001124
6007 003314 013777 001124 176104
6008 003322 017737 176100 001126
6009 003330 023737 001124 001126
6010 003336 001401
6011 003340 104004
6012
6013
(3)
(3)
(2) 003342 000004
(2)
(1) 003344 012737 000100 001160
6014 003352 012737 004000 001124
6015 003360 013777 001124 176040 1$:
6016 003366 017737 176034 001126
6017 003374 023737 001124 001126
6018 003402 001401
6019 003404 104004
6020 003406 006237 001124 2$:
6021 003412 001362
6022
6023
(4)
(4)
(3) 003414 000004
(3)
(2) 003416 012737 000100 001160
(1) 003424 012737 004000 001124
(1) 003432 013777 001124 175766
(1) 003440 017737 175762 001126 1$:
(1) 003446 023737 001124 001126
(2) 003454 001407
(1) 003456 017737 175744 001126
(1) 003464 104004
(1) 003466 013777 001124 175732 2$:
(1) 003474 006277 175726
(1) 003500 006237 001124
```

```
*****
*TEST 14 TEST THAT THE DAC #2 REGISTER CAN BE CLEARED
*****
TST14: SCOPE
CLR $GDDAT ;LOAD EXPECTED
MOV $GDDAT,@DAC2 ;LOAD REG
MOV @DAC2,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST15 ;;BR IF EQUAL
ERROR 4 ;ERROR, DAC #2 REGISTER NOT = 0

*****
*TEST 15 TEST THAT THE DAC #2 REGISTER CAN BE LOADED WITH #7777
*****
TST15: SCOPE
MOV #7777,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@DAC2 ;LOAD REG
MOV @DAC2,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST16 ;;BR IF EQUAL
ERROR 4 ;ERROR, DAC #2 REGISTER NOT = 7777

*****
*TEST 16 TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN
*****
TST16: 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 17 TEST THAT DAC2 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
*****
TST17: 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,$BDDAT ;SAVE FOR TYPEOUT
ERROR 4 ;DAC2 FAILED TO HOLD A FLOATING 1 PATTERN
2$: MOV $GDDAT,@DAC2 ;LOAD DAC2 AGAIN
ASR @DAC2 ;CHANGE THE DATA
ASR $GDDAT ;CHANGE THE EXPECTED
```


MAINDEC-11-CNAAA-A AAV11
CNAAA.P11 27-DEC-82 14:25

DIAGNOSTIC
T17

MACY11 30(1046) ^{L 2} 27-DEC-82 14:27 PAGE 67-1
TEST THAT DAC2 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)

SEQ 0024

(1) 003504 001355

BNE 1\$

;BR IF MORE DATA

6025

6026

(4)

(4)

(3)

(3)

(2)

(1)

(1)

(1)

(1)

(1)

(1)

(2)

(1)

(1)

(1)

(2)

6027

6028

(3)

(3)

(2)

(2)

6029

6030

6031

6032

6033

6034

6035

6036

(3)

(3)

(2)

(2)

6037

6038

6039

6040

6041

6042

6043

6044

(3)

(3)

(2)

(2)

6045

6046

6047

6048

6049

6050

6051

*TEST 20 TEST THE "SUB" INSTRUCTION WORKS ON DAC2

TST20: SCOPE

MOV #10,\$TIMES ;:DO 10 ITERATIONS

MOV #7777,\$GDDAT ;:LOAD EXPECTED

MOV \$GDDAT,@DAC2 ;:LOAD DAC2

1\$: SUB #1,\$GDDAT ;:SUB A VALUE

SUB #1,@DAC2 ;:FROM EXPECTED AND DAC2

MOV @DAC2,\$BDDAT ;:READ THE REGISTER

CMP \$GDDAT,\$BDDAT1 ;:COMPARE

BEQ 2\$;:BR IF SAME

ERROR 4 ;:THE SUB INSTRUCTION FAILED ON DAC2

MOV \$GDDAT,@DAC2 ;:LOAD THE REGISTER AGAIN

2\$: TST \$GDDAT ;:TEST FOR MORE DATA

BNE 1\$;:BR IF MORE DATA

*TEST 21 TEST THAT THE DAC #3 REGISTER CAN BE CLEARED

TST21: SCOPE

CLR \$GDDAT ;:LOAD EXPECTED

MOV \$GDDAT,@DAC3 ;:LOAD REG

MOV @DAC3,\$BDDAT ;:READ REG

CMP \$GDDAT,\$BDDAT ;:COMPARE

BEQ TST22 ;:BR IF EQUAL

ERROR 5 ;:ERROR, DAC #3 REGISTER NOT = 0

*TEST 22 TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777

TST22: SCOPE

MOV #7777,\$GDDAT ;:LOAD EXPECTED

MOV \$GDDAT,@DAC3 ;:LOAD REG

MOV @DAC3,\$BDDAT ;:READ REG

CMP \$GDDAT,\$BDDAT ;:COMPARE

BEQ TST23 ;:BR IF EQUAL

ERROR 5 ;:ERROR, DAC #3 REGISTER NOT = 7777

*TEST 23 TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN

TST23: SCOPE

MOV #100,\$TIMES ;:DO 100 ITERATIONS

MOV #BIT11,\$GDDAT ;:LOAD EXPECTED

1\$: MOV \$GDDAT,@DAC3 ;:LOAD DAC #3 REGISTER

MOV @DAC3,\$BDDAT ;:READ THE REGISTER

CMP \$GDDAT,\$BDDAT ;:COMPARE THE DATA

BEQ 2\$;:BR IF SAME

ERROR 5 ;:ERROR, DAC #3 REGISTER FAILED TO HOLD A FLOATIN

2\$: ASR \$GDDAT ;:CHANGE THE DATA

MAINDEC-11-CNAAA-A AAV11
CNAAA.P11 27-DEC-82 14:25

DIAGNOSTIC
T23

MACY11 30(1046) 27-DEC-82 14:27 PAGE 68-1
TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN

SEQ 0026

6052 003744 001362
6053

BNE 1\$

;BR AND TEST MORE DATA

6055

6056

(4)

(4)

(3)

(3)

(2)

(1)

(1)

(1)

(1)

(1)

(2)

(1)

(1)

(1)

(1)

(1)

6057

(4)

(4)

(3)

(3)

(2)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(2)

003746 000004

003750 012737 000100 001160

003756 012737 004000 001124

003764 013777 001124 175436

003772 017737 175432 001126

004000 023737 001124 001126

004006 001407

004010 017737 175414 001126

004016 104005

004020 013777 001124 175402

004026 006277 175376

004032 006237 001124

004036 001355

004040 000004

004042 012737 000010 001160

004050 012737 007777 001124

004056 013777 001124 175344

004064 162737 000001 001124

004072 162777 000001 175330

004100 017737 175324 001126

004106 023737 001124 001126

004114 001404

004116 104005

004120 013777 001124 175302

004126 005737 001124

004132 001354

*TEST 24 TEST THAT DAC3 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)

TST24: 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 25 TEST THE "SUB" INSTRUCTION WORKS ON DAC3

TST25: 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 $GDDAT,@DAC3 ;LOAD THE REGISTER AGAIN
2$: TST $GDDAT ;TEST FOR MORE DATA
BNE 1$ ;BR IF MORE DATA
```

6059
6060
(3)
(3)
(2)
(2)
6061
6062
6063
6064
6065
6066
6067
6068
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087

004134 000004
004136 012777
004144 012777
004152 012777
004160 012777
004166 012737
004174 017737
004202 023737
004210 001401
004212 104002
004214 012737
004222 017737
004230 023737
004236 001401
004240 104003
004242 012737
004250 017737
004256 023737
004264 001401
004266 104004
004270 012737
004276 017737
004304 023737
004312 001401
004314 104005

001111 175256
002222 175252
004444 175246
007777 175242
001111 001124
175222 001126
001124 001126
002222 001124
175176 001126
001124 001126
004444 001124
175152 001126
001124 001126
007777 001124
175126 001126
001124 001126

```
*****  
:*TEST 26 TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA  
*****  
TST26: SCOPE  
MOV #1111,@DAC0 ;LOAD DAC #0  
MOV #2222,@DAC1 ;LOAD DAC #1  
MOV #4444,@DAC2 ;LOAD DAC #2  
MOV #7777,@DAC3 ;LOAD DAC #3  
MOV #1111,$GDDAT ;LOAD EXPECTED  
MOV @DAC0,$BDDAT ;READ REG  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ 1$ ;;BR IF EQUAL  
ERROR 2 ;ERROR, SELECTED DAC #0 IN ERROR  
1$: MOV #2222,$GDDAT ;LOAD EXPECTED  
MOV @DAC1,$BDDAT ;READ REG  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ 2$ ;;BR IF EQUAL  
ERROR 3 ;ERROR, SELECTED DAC #1 IN ERROR  
2$: MOV #4444,$GDDAT ;LOAD EXPECTED  
MOV @DAC2,$BDDAT ;READ REG  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ 3$ ;;BR IF SAME  
ERROR 4 ;ERROR, SELECTED DAC #2 IN ERROR  
3$: MOV #7777,$GDDAT ;LOAD EXPECTED  
MOV @DAC3,$BDDAT ;READ REG  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST27 ;;BR IF SAME  
ERROR 5 ;ERROR, SELECTED DAC #3 IN ERROR
```

```
6089
6090      ;*****
(3)      ;*TEST 27      TEST THAT RESET CLEARS DAC #0 REGISTER
(3)      ;*****
(2) 004316 000004      TST27: SCOPE
(2)
(1) 004320 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
6091 004326 012777 177777 175066      MOV      #-1,@DAC0      ;LOAD THE REGISTER
6092 004334 005037 001124      CLR      $GDDAT      ;CLEAR EXPECTED
6093 004340 000005      RESET
6094 004342 017737 175054 001126      MOV      @DAC0,$BDDAT      ;READ REG
6095 004350 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
6096 004356 001401      BEQ      TST30      ;;BR IF EQUAL
6097      ;      ERROR      2      ;ERROR, RESET FAILED TO CLEAR DAC #0
6098 004360 000240      NOP      ;PATCH 1 OF CVA-AAA
6099
6100      ;*****
(3)      ;*TEST 30      TEST THAT RESET CLEARS DAC #1 REGISTER
(3)      ;*****
(2) 004362 000004      TST30: SCOPE
(2)
(1) 004364 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
6101 004372 012777 177777 175024      MOV      #-1,@DAC1      ;LOAD THE REGISTER
6102 004400 005037 001124      CLR      $GDDAT      ;CLEAR EXPECTED
6103 004404 000005      RESET
6104 004406 017737 175012 001126      MOV      @DAC1,$BDDAT      ;READ REG
6105 004414 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
6106 004422 001401      BEQ      TST31      ;;BR IF EQUAL
6107      ;      ERROR      3      ;ERROR, RESET FAILED TO CLEAR DAC #1
6108 004424 000240      NOP      ;PATCH 1 OF CVA-AAA
6109
6110      ;*****
(3)      ;*TEST 31      TEST THAT RESET CLEARS DAC #2 REGISTER
(3)      ;*****
(2) 004426 000004      TST31: SCOPE
(2)
(1) 004430 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
6112 004436 012777 177777 174762      MOV      #-1,@DAC2      ;LOAD THE REGISTER
6113 004444 005037 001124      CLR      $GDDAT      ;CLEAR EXPECTED
6114 004450 000005      RESET
6115 004452 017737 174750 001126      MOV      @DAC2,$BDDAT      ;READ THE REGISTER
6116 004460 001401      BEQ      TST32      ;;BR IF CLEARED
6117      ;      ERROR      4      ;ERROR, RESET FAILED TO CLEAR DAC #2
6118 004462 000240      NOP      ;PATCH 1 OD CVA-AAA
6119
6120      ;*****
(3)      ;*TEST 32      TEST THAT RESET CLEARS DAC #3 REGISTER
(3)      ;*****
(2) 004464 000004      TST32: SCOPE
(2)
(1) 004466 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
6121 004474 012777 177777 174726      MOV      #-1,@DAC3      ;LOAD THE REGISTER
6122 004502 005037 001124      CLR      $GDDAT      ;CLEAR THE EXPECTED
6123 004506 012737 000001 007012      MOV      #1,TEMP
6124 004514 000005      RESET
```

MAINDEC-11-CNAAB-A
CNAAAA.P11 27-DEC-82 14:25

AAV11 DIAGNOSTIC
14:25 T32

E 3
MACY11 30(1046) 27-DEC-82 14:27 PAGE 71-1
TEST THAT RESET CLEARS DAC #3 REGISTER

SEQ 0030

6125 004516 017737 174706 001126
6126 004524 001401
6127 :
6128 004526 000240
6129

MOV @DAC3,\$BDDAT ;READ THE REGISTER
BEQ TST33 ;:BR IF CLEARED
ERROR 5 ;ERROR, RESET FAILED TO CLEAR DAC #3
NOP ;PATCH 1 OF CVA AAA

```
6131  
6132 004530  
6133  
(3)  
(3)  
(2) 004530 000004  
(2)  
(1) 004532 012737 000001 001160  
6134 004540 005237 001206  
6135 004544 123737 001206 001420  
6136 004552 001424  
6137 004554 005237 001204  
6138 004560 063737 001416 001422  
6139 004566 063737 001416 001424  
6140 004574 063737 001416 001426  
6141 004602 063737 001416 001430  
6142 004610 006337 007010  
6143 004614 005037 001102  
6144 004620 000137 002306  
6145  
6171  
(3)  
(3)  
(2) 004624 000004  
(2)  
(1) 004626 012737 000001 001160  
6172 004634 005737 007020  
6173 004640 001002  
6174 004642 000137 005454
```

REMAIN:

*TEST 33 DETERMINE IF MORE AAV11'S REMAIN TO BE TESTED

TST33: SCOPE

```
MOV #1,$TIMES ;;DO 1 ITERATION  
INC $UNIT ;UPDATE UNIT #  
CMPB $UNIT,EVER ;TEST IF MORE  
BEQ TST34 ;;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 $TSTNM  
JMP TST1 ;TEST THE NEXT UNIT
```

*TEST 34 DETERMINE IF RUNNING ON THE HARDWARE TESTER (IF NOT REPORT END OF PA

TST34: SCOPE

```
MOV #1,$TIMES ;;DO 1 ITERATION  
TST WFST ;TEST IF ON TESTER  
BNE TST35 ;;BR TO TEST  
JMP $EOP
```



```
6176
6177      ::*****
(3)      : *TEST 35      TEST THAT DAC #3 OUTPUT BITS (0-3) FUNCTION
(3)      : *****
(2) 004646 000004      TST35: SCOPE
(2)
(1) 004650 012737 000001 001160      MOV #1,$TIMES      ;;DO 1 ITERATION
6178 004656 012737 000010 007014      MOV #BIT3,$TEMP      ;LOAD DAC PATTERN
6179 004664 012737 004000 001124      MOV #BIT11,$GDDAT      ;LOAD EXPECTED PATTERN
6180
6181 004672 013777 007014 174530 1$: MOV $TEMP,@DAC3      ;LOAD DAC REGISTER
6182 004700 017737 002120 001126      MOV @DRIN,$BDDAT      ;READ THE REGISTER
6183 004706 042737 170377 001126      BIC #170377,$BDDAT      ;MASK OFF OTHER BITS
6184 004714 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE
6185 004722 001401      BEQ 2$      ;;BR IF THE SAME
6186 004724 104013      ERROR 13      ;DAC #3 DIGITAL OUTPUT BITS IN ERROR
6187
6188 004726 006237 001124 2$: ASR $GDDAT      ;ADJUST EXPECTED
6189 004732 006237 007014      ASR $TEMP      ;ADJUST LOADED PATTERN
6190 004736 001355      BNE 1$
6191
6192      ::*****
(3)      : *TEST 36      VERIFY THE AAV11 +15 SUPPLY
(3)      : *****
(2) 004740 000004      TST36: SCOPE
(2)
(1) 004742 012737 000001 001160      MOV #1,$TIMES      ;;DO 1 ITERATION
6193 004750 013737 007032 001124      MOV V5744,$GDDAT      ;LOAD EXPECTED
6194 004756 004537 006302      JSR R5,CONVRT      ;SAMPLE THE CHANNEL
6195 004762 000012      12
6196 004764 013737 007034 007016      MOV V144,SPREAD      ;LOAD TOLERANCE
6197 004772 004737 006520      JSR PC,COMPAR      ;TEST IT
6198 004776 000401      BR TST37      ;;BR
6199 005000 104011      ERROR 11      ;+15 VOLT SUPPLY IS WRONG
6200
6201      ::*****
(3)      : *TEST 37      VERIFY THE AAV11 -15 SUPPLY
(3)      : *****
(2) 005002 000004      TST37: SCOPE
(2)
(1) 005004 012737 000001 001160      MOV #1,$TIMES      ;;DO 1 ITERATION
6202 005012 013737 007036 001124      MOV V2034,$GDDAT      ;LOAD EXPECTED
6203 005020 004537 006302      JSR R5,CONVRT      ;SAMPLE THE CHANNEL
6204 005024 000011      11
6205 005026 013737 007034 007016      MOV V144,SPREAD      ;LOAD TOLERANCE
6206 005034 004737 006520      JSR PC,COMPAR      ;TEST IT
6207 005040 000401      BR TST40      ;;BR
6208 005042 104012      ERROR 12      ;-15 VOLT SUPPLY IS WRONG
6209
```

```
6211
6212
(4)
(4)
(3) 005044 000004
(3)
(2) 005046 012737 000001 001160
(1) 005054 005737 001202
(3) 005060 001006
(1)
(1) 005062 004537 005634
(1) 005066 001422
(1) 005070 010726
(1) 005072 010552
(1) 005074 000013
(1)
(5)
(4)
(4)
(3) 005076 000004
(3)
(2) 005100 012737 000001 001160
(1) 005106 005737 001202
(3) 005112 001005
(1)
(1) 005114 004537 005764
(1) 005120 001422
(1) 005122 010376
(1) 005124 000013
(1)
(5)
(4)
(4)
(3) 005126 000004
(3)
(2) 005130 012737 000001 001160
(1) 005136 004537 006104
(1) 005142 001422
(1) 005144 000013

*****
*TEST 40 DACO OFFSET ADJUSTMENT
*****
TST40: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST $PASS ;TEST IF FIRST PASS
BNE TST41 ;;BR IF NOT
JSR R5,OFFDAC ;LOAD AND EXECUTE DAC OFFSET ADJ.
DACO ;DAC ADDRESS
SELDO ;TYPEOUT ADDRESS
ADJR46 ;RES. TO ADJUST
13 ;RESULT CHANNEL #

*****
*TEST 41 DACO GAIN ADJUSTMENT
*****
TST41: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST $PASS ;TEST IF FIRST PASS
BNE TST42 ;;BR IF NOT
JSR R5,GAIDAC ;LOAD AND EXECUTE DAC GAIN ADJ.
DACO ;DAC ADDRESS
ADJR34 ;RES. TO ADJUST
13 ;CHANNEL # FOR RESULTS

*****
*TEST 42 DACO CALIBRATION
*****
TST42: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
JSR R5,CALDAC ;LOAD AND EXECUTE CALIBRATION
DACO ;DAC ADDRESS
13 ;CHANNEL # FOR RESULTS
```

```
6214
6215
(4)
(4)
(3) 005146 000004
(3)
(2) 005150 012737 000001 001160      MOV    #1,$TIMES      ;;DO 1 ITERATION
(1) 005156 005737 001202                TST    $PASS          ;:TEST IF FIRST PASS
(3) 005162 001006                BNE    TST44          ;:BR IF NOT
(1)
(1) 005164 004537 005634                JSR    R5,OFFDAC      ;LOAD AND EXECUTE DAC OFFSET ADJ.
(1) 005170 001424                DAC1   ;DAC ADDRESS
(1) 005172 010744                SELD1  ;TYPEOUT ADDRESS
(1) 005174 010605                ADJR47 ;RES. TO ADJUST
(1) 005176 000014                14     ;RESULT CHANNEL #
(1)
(5)
(4)
(4)
(3) 005200 000004
(3)
(2) 005202 012737 000001 001160      MOV    #1,$TIMES      ;;DO 1 ITERATION
(1) 005210 005737 001202                TST    $PASS          ;:TEST IF FIRST PASS
(3) 005214 001005                BNE    TST45          ;:BR IF NOT
(1)
(1) 005216 004537 005764                JSR    R5,GAIDAC     ;LOAD AND EXECUTE DAC GAIN ADJ.
(1) 005222 001424                DAC1   ;DAC ADDRESS
(1) 005224 010431                ADJR35 ;RES. TO ADJUST
(1) 005226 000014                14     ;CHANNEL # FOR RESULTS
(1)
(5)
(4)
(4)
(3) 005230 000004
(3)
(2) 005232 012737 000001 001160      MOV    #1,$TIMES      ;;DO 1 ITERATION
(1) 005240 004537 006104                JSR    R5,CALDAC     ;LOAD AND EXECUTE CALIBRATION
(1) 005244 001424                DAC1   ;DAC ADDRESS
(1) 005246 000014                14     ;CHANNEL # FOR RESULTS
```

```
6217
6218
(4)
(4)
(3) 005250 000004
(3)
(2) 005252 012737 000001 001160
(1) 005260 005737 001202
(3) 005264 001006
(1)
(1) 005266 004537 005634
(1) 005272 001426
(1) 005274 010762
(1) 005276 010640
(1) 005300 000016
(1)
(5)
(4)
(4)
(3) 005302 000004
(3)
(2) 005304 012737 000001 001160
(1) 005312 005737 001202
(3) 005316 001005
(1)
(1) 005320 004537 005764
(1) 005324 001426
(1) 005326 010464
(1) 005330 000016
(1)
(5)
(4)
(4)
(3) 005332 000004
(3)
(2) 005334 012737 000001 001160
(1) 005342 004537 006104
(1) 005346 001426
(1) 005350 000016

*****
*TEST 46 DAC2 OFFSET ADJUSTMENT
*****
TST46: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST $PASS ;:TEST IF FIRST PASS
BNE TST47 ;:BR IF NOT
JSR R5,OFFDAC ;LOAD AND EXECUTE DAC OFFSET ADJ.
DAC2 ;DAC ADDRESS
SELD2 ;TYPEOUT ADDRESS
ADJR48 ;RES. TO ADJUST
16 ;RESULT CHANNEL #

*****
*TEST 47 DAC2 GAIN ADJUSTMENT
*****
TST47: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST $PASS ;:TEST IF FIRST PASS
BNE TST50 ;:BR IF NOT
JSR R5,GAIDAC ;LOAD AND EXECUTE DAC GAIN ADJ.
DAC2 ;DAC ADDRESS
ADJR36 ;RES. TO ADJUST
16 ;CHANNEL # FOR RESULTS

*****
*TEST 50 DAC2 CALIBRATION
*****
TST50: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
JSR R5,CALDAC ;LOAD AND EXECUTE CALIBRATION
DAC2 ;DAC ADDRESS
16 ;CHANNEL # FOR RESULTS
```

6220
6221
(4)
(4)
(3)
(3)
(2)
(1)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4)
(4)
(3)
(3)
(2)
(1)
(3)
(1)
(1)
(1)
(1)
(1)
(5)
(4)
(4)
(3)
(3)
(2)
(1)
(1)
(1)
(1)

005352 000004
005354 012737 000001 001160
005362 005737 001202
005366 001006
005370 004537 005634
005374 001430
005376 011000
005400 010673
005402 000015

005404 000004
005406 012737 000001 001160
005414 005737 001202
005420 001005
005422 004537 005764
005426 001430
005430 010517
005432 000015

005434 000004
005436 012737 000001 001160
005444 004537 006104
005450 001430
005452 000015

```
*****  
*TEST 51 DAC3 OFFSET ADJUSTMENT  
*****  
TST51: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
TST $PASS ;TEST IF FIRST PASS  
BNE TST52 ;;BR IF NOT  
  
JSR R5,OFFDAC ;LOAD AND EXECUTE DAC OFFSET ADJ.  
DAC3 ;DAC ADDRESS  
SELD3 ;TYPEOUT ADDRESS  
ADJR49 ;RES. TO ADJUST  
15 ;RESULT CHANNEL #  
  
*****  
*TEST 52 DAC3 GAIN ADJUSTMENT  
*****  
TST52: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
TST $PASS ;TEST IF FIRST PASS  
BNE TST53 ;;BR IF NOT  
  
JSR R5,GAIDAC ;LOAD AND EXECUTE DAC GAIN ADJ.  
DAC3 ;DAC ADDRESS  
ADJR37 ;RES. TO ADJUST  
15 ;CHANNEL # FOR RESULTS  
  
*****  
*TEST 53 DAC3 CALIBRATION  
*****  
TST53: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
JSR R5,CALDAC ;LOAD AND EXECUTE CALIBRATION  
DAC3 ;DAC ADDRESS  
15 ;CHANNEL # FOR RESULTS
```

6223
6224
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 005454
(1) 005454 000004
(1) 005456 005037 001102
(1) 005462 005037 001160
(1) 005466 005237 001202
(1) 005472 042737 100000 001202
(1) 005500 005327
(1) 005502 000001
(1) 005504 003022
(1) 005506 012737
(1) 005510 000001
(1) 005512 005502
(1) 005514 104401 005561
(2) 005520 013746 001202
(2) 005524 104405
(1) 005526 104401 005556
(1) 005532 013700 000042
(1) 005536 001405
(1) 005540 000005
(1) 005542 004710
(1) 005544 000240
(1) 005546 000240
(1) 005550 000240
(1) 005552
(1) 005552 000137
(1) 005554 005576
(1)
(1)
(1) 005556 377 377 000
(1) 005561 015 042412 042116
(1) 005566 050040 051501 020123
(1) 005574 000043

.SBTTL END OF PASS ROUTINE

::*****
:*INCREMENT THE PASS NUMBER (\$PASS)
:*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO INIT7

\$EOP:
SCOPE
CLR \$STSTM ;;ZERO THE TEST NUMBER
CLR \$TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC \$PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,\$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
\$EOPCT: .WORD 1
BGT \$DOAGN ;;YES
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
\$ENDCT: .WORD 1
\$EOPCT
TYPE \$SENDMG ;;TYPE 'END PASS #'
MOV \$PASS,-(SP) ;;SAVE \$PASS FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE \$ENULL ;;TYPE A NULL CHARACTER
\$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ \$DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
\$ENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11
\$DOAGN:
JMP @(PC)+ ;;RETURN
\$RTNAD: .WORD INIT7

\$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
\$SENDMG: .ASCIZ <15><12>/END PASS #/

6225
6226 005576 005737 007020 INIT7: TST WFTST ;TEST IF ON TESTER
6227 005602 001012 BNE 1\$
6228 005604 104401 010176 TYPE, ERRTOT
6229 005610 013746 001112 MOV \$ERTTL,-(SP)
6230 005614 104405 TYPDS
6231 005616 104401 010210 TYPE, MESGD
6232 005622 013746 007004 MOV BADUNT,-(SP) ;;SAVE BADUNT FOR TYPEOUT
(1) 005626 104406 TYPBN ;;GO TYPE--BINARY ASCII
6233 005630 000137 002044 1\$: JMP INIT1 ;TEST IT AGAIN

```

6235
6236      .SBTTL  SUBROUTINE TO ADJUST THE DAC'S OFFSET POTS
6237
6238 005634 012537 005762 OFFDAC: MOV      (R5)+,10$      ;GET BUS ADDRESS
6239 005640 017737 000116 005762 MOV      @10$,10$      ;
6240 005646 013737 005762 007002 MOV      10$,DACBAD    ;LOAD BUS ADDRESS IF ERROR
6241 005654 012537 005672 MOV      (R5)+,11$     ;GET POINTER TO ASCII MESSAGE
6242 005660 012537 005712 MOV      (R5)+,12$     ;GET POINTER TO RES. MESSAGE
6243 005664 012537 005732 MOV      (R5)+,13$     ;GET AND SAVE CHANNEL #
6244 005670 104401 TYPE                                ;TELL OPERATOR TO SELECT DAC N
6245 005672 010726 11$: SELDO
6246 005674 004537 006212 JSR      R5,SNDVLT     ;LOAD A VOLTAGE OF N5.1200 VOLTS.
6247 005700 011064 N51200
6248 005702 012777 000000 000052 MOV      #0000,@10$   ;LOAD THE SELECTED DAC TO NULL
6249 005710 104401 TYPE                                ;TELL OPERATOR TO ADJUST RXR FOR NULL
6250 005712 010552 12$: ADJR46
6251 005714 004737 006422 1$: JSR      PC,CSPACE   ;WAIT UNTIL THE IS READY
6252 005720 012737 000000 001124 MOV      #0000,$GDDAT ;LOAD EXPECTED VALUE
6253 005726 004537 006302 JSR      R5,CONVRT    ;SAMPLE THE CHANNEL
6254 005732 000013 13$: 13
6255 005734 012737 000002 007016 MOV      #2,SPREAD
6256 005742 004737 006520 JSR      PC,COMPAR    ;TEST RESULTS
6257 005746 000404 BR      2$            ;:BR IF WITHIN THE LIMIT
6258 005750 104006 ERROR 6 ;SELECTED DAC OFFSET POT WAS NOT ADJUSTED INCORRECTLY
6259 005752 104401 TYPE
6260 005754 011016 TRYAGN
6261 005756 000756 BR      1$            ;LOOP AGAIN
6262 005760 000205 2$: RTS      R5 ;EXIT
6263 005762 000000 10$: 0
6264
6265      .SBTTL  SUBROUTINE TO ADJUST THE GAIN ADJUSTMENT POTS
6266
6267 005764 012537 006102 GAIDAC: MOV      (R5)+,10$      ;GET BUS ADDRESS
6268 005770 017737 000106 006102 MOV      @10$,10$      ;
6269 005776 013737 006102 007002 MOV      10$,DACBAD    ;LOAD BUS ADDRESS IF ERROR
6270 006004 012537 006032 MOV      (R5)+,11$     ;GET ASCII RES. ADDRESS
6271 006010 012537 006052 MOV      (R5)+,12$     ;GET CHANNEL #
6272 006014 004537 006212 JSR      R5,SNDVLT     ;LOAD + 5.1175 VOLTS
6273 006020 011077 P51175
6274 006022 012777 007777 000052 MOV      #7777,@10$   ;LOAD THE DAC
6275 006030 104401 TYPE                                ;TELL OPERATOR WHICH RXR TO ADJUST FOR NULL
6276 006032 010376 11$: ADJR34
6277 006034 004737 006422 1$: JSR      PC,CSPACE   ;WAIT FOR OPERATOR
6278 006040 012737 007777 001124 MOV      #7777,$GDDAT ;LOAD EXPECTED
6279 006046 004537 006302 JSR      R5,CONVRT    ;CONVERT THE VALUE
6280 006052 000013 12$: 13
6281 006054 012737 000002 007016 MOV      #2,SPREAD    ;LOAD LIMIT
6282 006062 004737 006520 JSR      PC,COMPAR    ;TEST RESULTS
6283 006066 000404 BR      2$            ;:BR IF WITHIN LIMITS
6284 006070 104007 ERROR 7 ;SELECTED DAC GAIN POT WAS NOT ADJUSTED PROPERLY
6285 006072 104401 TYPE
6286 006074 011016 TRYAGN
6287 006076 000756 BR      1$            ;EXIT
6288 006100 000205 2$: RTS      R5
6289 006102 000000 10$: 0

```

```

6291
6292
6293
6294 006104 012537 006210
6295 006110 017737 000074 006210
6296 006116 013737 006210 007002
6297 006124 012537 006150
6298
6299 006130 012777 007400 000052
6300 006136 012737 007400 001124
6301
6302 006144 004537 006302
6303 006150 000013
6304
6305 006152 012737 000003 007016
6306 006160 004737 006520
6307 006164 000401
6308 006166 104010
6309 006170 162777 000400 000012
6310 006176 162737 000400 001124
6311 006204 001357
6312 006206 000205
6313
6314 006210 000000
6315
6316
6317
6318 006212 012500
6319 006214 112001
6320 006216 001421
6321 006220 110177 000576
6322 006224 012701 001000
6323 006230 005301
6324 006232 001376
6325 006234 052777 000200 000560
6326 006242 012701 001000
6327 006246 005301
6328 006250 001376
6329 006252 042777 000200 000542
6330 006260 000755
6331
6332 006262 012701 000000
6333 006266 152777 000177 000526
6334 006274 005301
6335 006276 001376
6336 006300 000205
6337

.SBTTL SUBROUTINE TO TEST THE D/A CALIBRATION
CALDAC: MOV (R5)+,10$ ;GET BUS ADDRESS
MOV @10$,10$ ;
MOV 10$,DACBAD ;LOAD BUS ADDRESS IF ERROR
MOV (R5)+,11$ ;GET CHANNEL #
MOV #7400,@10$ ;LOAD THE DAC
MOV #7400,$GDDAT ;LOAD THE EXPECTED VALUE
1$: JSR R5,CONVRT ;SAMPLE THE CHANNEL
11$: 13
MOV #3,SPREAD ;LOAD TOLERANCE
JSR PC,COMPAR ;TEST THE RESULTS
BR 2$ ;;BR
ERROR 10 ;NON-LINEARITY IN DAC DETECTED
2$: SUB #400,@10$ ;ADJUST THE CONTENTS
SUB #400,$GDDAT ;ADJUST THE EXPECTED
BNE 1$ ;;BR IF NOT DONE
RTS R5 ;EXIT

10$: 0

.SBTTL SUBROUTINE TO LOAD A VOLTAGE INTO THE VOLTAGE SOURCE
SNDVLT: MOV (R5)+,R0 ;LOAD THE POINTER
2$: MOVB (R0)+,R1 ;GET SOME DATA
BEQ 3$ ;BR IF TERM
MOVB R1,@FILZ ;LOAD THE DATA
MOV #1000,R1
5$: DEC R1 ;DELAY
BNE 5$
BIS #BIT7,@FILZ ;SET BIT 7
MOV #1000,R1 ;LOAD DELAY
1$: DEC R1 ;DELAY
BNE 1$
BIC #BIT7,@FILZ
BR 2$
3$: MOV #0,R1 ;LOAD DELAY
BISB #177,@FILZ ;DISABLE BITS
4$: DEC R1 ;DELAY
BNE 4$
RTS R5 ;EXIT

```



```

6339
6340      .SBTTL  SUBROUTINE TO CONVERT CHANNEL N ON THE TESTER A/D
6341
6342      006302  012537  006414      CONVRT:  MOV      (R5)+,10$      ;GET THE CHANNEL #
6343      006306  000337  006414      SWAB      10$
6344      006312  042737  170377  006414      BIC      #170377,10$      ;MASK OUT OTHER BITS
6345      006320  013777  006414  000500      MOV      10$,@ADCS      ;SELECT CHANNEL
6346      006326  005037  006416      CLR      11$
6347      006332  012737  000200  006420      MOV      #BIT7,12$      ;LOAD SHIFT COUNTER
6348      006340  105277  000462      1$:      INCB     @ADCS      ;CONVERT CHANNEL
6349      006344  105777  000456      2$:      TSTB     @ADCS      ;WAIT FOR DONE
6350      006350  100375
6351      006352  067737  000452  006416      BPL      2$
6352      006360  006237  006420      ADD      @ADBR,11$      ;UPDATE CONVERSION
6353      006364  001365      ASR      12$      ;FINISHED ?
6354      006366  000257      BNE      1$
6355      006370  006037  006416      CCC
6356      006374  006237  006416      ROR      11$
6357      006400  006237  006416      ASR      11$
6358      006404  013737  006416  001126      ASR      11$      ;JUSTIFY DATA
6359      006412  000205      MOV      11$, $BDDAT      ;LOAD ACTUAL <ADJUSTED>
6360      RTS      R5      ;EXIT
6361      006414  000000      10$:     0
6362      006416  000000      11$:     0
6363      006420  000000      12$:     0
6364
6365      .SBTTL  SUBROUTINE TO LOOP UNTIL OPERATOR TYPES AN "SPACE"
6366
6367      006422  104401  007746      CSPACE:  TYPE,    LDSPAC      ;TELL OPERATOR TO HIT SPACE BAR
6368      006426  012737  000014  006516      3$:      MOV      #14,11$      ;LOAD DELAY COUNTER
6369      006434  005037  006514      CLR      10$
6370      006440  105777  172500      1$:      TSTB     @STKS      ;WAIT FOR OPERATOR
6371      006444  100410      BMI      2$      ;BR IF FLAG IS SET
6372      006446  005337  006514      DEC      10$      ;DELAY
6373      006452  001372      BNE      1$      ;BR IF NOT DONE
6374      006454  005337  006516      DEC      11$      ;DELAY AGAIN
6375      006460  001367      BNE      1$
6376      006462  104014      ERROR    14      ;WAKE UP OPERATOR
6377      006464  000760      BR       3$      ;LOOP
6378      006466  017737  172454  006514      2$:      MOV      @STKB,10$      ;READ THE CHARACTER
6379      006474  042737  177600  006514      BIC      #177600,10$      ;MASK OF OTHER BITS
6380      006502  022737  000040  006514      CMP      #40,10$      ;TEST FOR "SPACE"
6381      006510  001346      BNE      3$      ;LOOP
6382      006512  000207      RTS      PC      ;EXIT
6383      006514  000000      10$:     0
6384      006516  000010      11$:     BIT3
6385
  
```

```

6387
6388
6389
6390 006520 010046
6391 006522 010146
6392 006524 013700 001124
6393 006530 013701 001126
6394 006534 160100
6395 006536 100001
6396 006540 005400
6397 006542 020037 007016
6398 006546 003405
6399 006550 012601
6400 006552 012600
6401 006554 062716 000002
6402 006560 000207
6403
6404 006562 012601
6405 006564 012600
6406 006566 000207
6407
6414
6415
6416 006570 012706 001100
6417 006574 004737 001746
6418 006600 013700 001422
6419 006604 004737 006642
6420 006610 013700 001424
6421 006614 004737 006642
6422 006620 013700 001426
6423 006624 004737 006642
6424 006630 013700 001430
6425 006634 004737 006642
6426 006640 000757
6427
6428 006642 005010
6429 006644 062710 000010
6430 006650 005710
6431 006652 001374
6432 006654 000207

.SBTTL SUBROUTINE TO COMPARE TWO LOCATIONS BY THE SPREAD
COMPAR: MOV R0,-(SP) ;SAVE R0
MOV R1,-(SP) ;SAVE R1
MOV $GDDAT,R0 ;GET EXPECTED VALUE
MOV $BDDAT,R1 ;GET THE UNKNOWN
SUB R1,R0 ;SUBTRACT
BPL 8$
NEG R0
8$: CMP R0,SPREAD ;TEST IF DIFFERENCE IF > SHAN SPREAD
BLE 10$
9$: MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R0 ;RESTORE R0
ADD #2,(SP) ;MAKE AN ERROR EXIT
RTS PC ;EXIT

10$: MOV (SP)+,R1
MOV (SP)+,R0
RTS PC ;EXIT FOR GOOD LIMIT TEST

.SBTTL FULL SCALE RAMP ON EACH RAMP
FULRMP: MOV #STACK,SP ;LOAD POINTER
JSR PC,LDTRAP ;LOAD BUS ADDRESS
1$: MOV DAC0,R0 ;GET BUS ADDRESS
JSR PC,10$ ;LOAD THE RAMP ON DAC #1
MOV DAC1,R0 ;GET BUS ADDRESS
JSR PC,10$ ;LOAD THE RAMP ON DAC #1
MOV DAC2,R0 ;GET BUS ADDRESS
JSR PC,10$ ;LOAD THE RAMP ON DAC #2
MOV DAC3,R0 ;GET THE BUS ADDRESS
JSR PC,10$ ;LOAD THE RAMP ON DAC #3
BR 1$ ;BR BACK

10$: CLR (R0) ;CLEAR DAC
11$: ADD #10,(R0) ;UPDATE THE DATA
TST (R0) ;TEST IF DONE
BNE 11$ ;BR IF NOT
RTS PC ;EXIT

```

6434
 6435
 6436
 6437
 6438
 6439
 6440
 6441
 6442
 6443
 6444
 6445
 6446
 6447
 6448
 6449
 6450
 6451
 6452
 6453
 6454
 6455
 6456
 6457
 6458
 6459
 6460
 6461
 6462
 6463
 6464
 6465
 6466
 6467
 6468
 6469
 6470
 6471
 6472
 6473
 6474
 6475
 6476
 6477
 6478
 6479
 6480
 6481
 6482
 6483
 6489

006656 012706 001100
 006662 004737 001746
 006666 104410
 006670 017700 172244
 006674 010077 172522
 006700 010077 172520
 006704 010077 172516
 006710 010077 172514
 006714 000764
 006716 012706 001100
 006722 004737 001746
 006726 104410
 006730 017700 172204
 006734 004737 006750
 006740 005000
 006742 004737 006750
 006746 000767
 006750 010077 172446
 006754 010077 172444
 006760 010077 172442
 006764 010077 172440
 006770 012700 000020
 006774 005300
 006776 100376
 007000 000207
 007002 174440
 007004 000000
 007006 000004
 007010 000001
 007012 000000
 007014 000000
 007016 000000
 007020 000000
 007022 167772
 007024 167774
 007026 170500
 007030 170502
 007032 005744
 007034 000144
 007036 002034

.SBTTL STATIC DAC CALIBRATION

STATIC: MOV #STACK,SP ;LOAD STACK POINTER
 JSR PC,LDTRAP ;LOAD BUS ADDRESSES
 1\$: CKSWR ;TEST FOR CTRL G
 MOV @SWR,RO ;READ SWITCHES
 MOV RO,@DAC0 ;LOAD DAC #0
 MOV RO,@DAC1 ;LOAD DAC #1
 MOV RO,@DAC2 ;LOAD DAC #2
 MOV RO,@DAC3 ;LOAD DAC #3
 BR 1\$

.SBTTL DYNAMIC DAC CALIBRATION

DYNCAL: MOV #STACK,SP ;LOAD STACK POINTER
 JSR PC,LDTRAP ;LOAD BUS ADDRESSES
 1\$: CKSWR ;TEST FOR CTRL G
 MOV @SWR,RO ;READ SWR
 JSR PC,10\$;LOAD THE SWR VALUE TO ALL DACS
 CLR RO ;CLEAR RO
 JSR PC,10\$;LOAD ALL DAC'S WITH 0
 BR 1\$
 10\$: MOV RO,@DAC0 ;LOAD DAC #0
 MOV RO,@DAC1 ;LOAD DAC #1
 MOV RO,@DAC2 ;LOAD DAC #2
 MOV RO,@DAC3 ;LOAD DAC #3
 MOV #20,RO ;LOAD DELAY COUNTER
 11\$: DEC RO ;DELAY
 BPL 11\$;WAIT
 RTS PC ;EXIT

DACBAD: ABASE
 BADUNT: 0
 NUMBOK: 4
 MASKNM: BIT0
 TEMP: 0
 \$TEMP: 0
 SPREAD: 0
 WFTST: 0
 FILZ: 167772
 DRIN: 167774
 ADCS: 170500
 ADBR: 170502
 V5744: 5744
 V144: 144
 V2034: 2034

6491
6492
6493
6494
6495

.SBTTL ASCII MESSAGES

TITLE: .ASCIZ <15><12><12>'AAV11 DIAGNOSTIC TEST, (MAINDEC-11-CNAAA-A0)'<<15><12>

007040 005015 040412 053101
007046 030461 042040 040511
007054 047107 051517 044524
007062 020103 042524 052123
007070 020054 046450 044501
007076 042116 041505 030455
007104 026461 047103 040501
007112 026501 030101 006451
007120 000012

6496 007122 052502 020123 044524 EM1: .ASCIZ /BUS TIME-OUT WHEN REFERENCING A DAC ADDRESS/
007130 042515 047455 052125
007136 053440 042510 020116
007144 042522 042506 042522
007152 041516 047111 020107
007160 020101 040504 020103
007166 042101 051104 051505
007174 000123

6497 007176 040504 030103 051040 EM2: .ASCIZ /DAC0 REGISTER IN ERROR/
007204 043505 051511 042524
007212 020122 047111 042440
007220 051122 051117 000

6498 007225 104 041501 020061 EM3: .ASCIZ /DAC1 REGISTER IN ERROR/
007232 042522 044507 052123
007240 051105 044440 020116
007246 051105 047522 000122

6499 007254 040504 031103 051040 EM4: .ASCIZ /DAC2 REGISTER IN ERROR/
007262 043505 051511 042524
007270 020122 047111 042440
007276 051122 051117 000

6500 007303 104 041501 020063 EM5: .ASCIZ /DAC3 REGISTER IN ERROR/
007310 042522 044507 052123
007316 051105 044440 020116
007324 051105 047522 000122

6501 007332 042523 042514 052103 EM6: .ASCIZ /SELECTED DAC OFFSET POT WAS ADJUSTED INCORRECTLY/
007340 042105 042040 041501
007346 047440 043106 042523
007354 020124 047520 020124
007362 040527 020123 042101
007370 052512 052123 042105
007376 044440 041516 051117
007404 042522 052103 054514
007412 000

6502 007413 123 046105 041505 EM7: .ASCIZ /SELECTED DAC GAIN POT WAS ADJUSTED INCORRECTLY/
007420 042524 020104 040504
007426 020103 040507 047111
007434 050040 052117 053440
007442 051501 040440 045104
007450 051525 042524 020104
007456 047111 047503 051122
007464 041505 046124 000131

6503 007472 042523 042514 052103 EM10: .ASCIZ /SELECTED DAC HAS A LINEARITY PROBLEM/
007500 042105 042040 041501

	007506	044040	051501	040440	
	007514	046040	047111	040505	
	007522	044522	054524	050040	
	007530	047522	046102	046505	
	007536	000			
6504	007537	053	032461	053040	EM11: .ASCIZ /+15 VOLT SUPPLY IS INCORRECT/
	007544	046117	020124	052523	
	007552	050120	054514	044440	
	007560	020123	047111	047503	
	007566	051122	041505	000124	
6505	007574	030455	020065	047526	EM12: .ASCIZ /-15 VOLT SUPPLY IS INCORRECT/
	007602	052114	051440	050125	
	007610	046120	020131	051511	
	007616	044440	041516	051117	
	007624	042522	052103	000	
6506	007631	104	041501	021440	EM13: .ASCIZ /DAC #3 DIGITAL OUTPUT BITS IN ERROR/
	007636	020063	044504	044507	
	007644	040524	020114	052517	
	007652	050124	052125	041040	
	007660	052111	020123	047111	
	007666	042440	051122	051117	
	007674	000			
6507	007675	007	003407	040527	EM14: .ASCIZ <7><7><7>/WAKE UP OPERATOR AND ADJUST THE POT/<7><7>
	007702	042513	052440	020120	
	007710	050117	051105	052101	
	007716	051117	040440	042116	
	007724	040440	045104	051525	
	007732	020124	044124	020105	
	007740	047520	003524	000007	
6508	007746	042011	050105	042522	LDSPAC: .ASCIZ / DEPRESS THE "SPACE-BAR" WHEN DONE/
	007754	051523	052040	042510	
	007762	021040	050123	041501	
	007770	026505	040502	021122	
	007776	053440	042510	020116	
	010004	047504	042516	000	
6509	010011	007	005015	042524	MSGSW: .ASCII <7><15><12>/TESTER SW1-2 AND SW1-5 ONLY MUST BE ON/
	010016	052123	051105	051440	
	010024	030527	031055	040440	
	010032	042116	051440	030527	
	010040	032455	047440	046116	
	010046	020131	052515	052123	
	010054	041040	020105	047117	
6510	010062	006407	051412	031127	.ASCII <7><15><12>/SW2-2 SW2-4 AND SW2-5 MUST BE ON/
	010070	031055	051440	031127	
	010076	032055	040440	042116	
	010104	051440	031127	032455	
	010112	046440	051525	020124	
	010120	042502	047440	116	
6511	010125	015	003412	047503	.ASCIZ <15><12><7>/CONNECT AAV11 TO J09 OF TESTER ONLY/<15><12>
	010132	047116	041505	020124	
	010140	040501	030526	020061	
	010146	047524	045040	034460	
	010154	047440	020106	042524	
	010162	052123	051105	047440	
	010170	046116	006531	000012	
6512	010176	021440	042440	051122	ERRTOT: .ASCIZ / # ERRORS/

6513	010204	051117	000123	052440	MESGD: .ASCIZ / BAD UNITS /
	J10210	041040	042101	000040	
6514	010216	044516	051524		FOUND1: .BYTE 15,12
6515	010224	015 012	043517	040522	.ASCIZ /PROGRAM DETECTED /
	010234	020115	042504	042524	
6516	010242	052103	042105	000040	FOUND2: .ASCIZ /(8) AAV11(S) /
	010250	034050	004451	040501	
	010256	030526	024061	024523	
	010264	020040	000		
6517	010267	105	051122	041520	DH1: .ASCIZ /ERRPC BUSADR EXPECT WAS/
	010274	041011	051525	042101	
	010302	020122	042440	050130	
	010310	041505	020124	020040	
	010316	040527	000123		
6518	010322	051105	050122	004503	DH2: .ASCIZ /ERRPC BUSADR/
	010330	052502	040523	051104	
	010336	000			
6519	010337	105	051122	041520	DH6: .ASCIZ /ERRPC BUSADR EXPECT WAS SPREAD/
	010344	041011	051525	042101	
	010352	004522	054105	042520	
	010360	052103	053411	051501	
	010366	051411	051120	040505	
	010374	000104			
6523	010376	005015	040440	045104	ADJR34: .ASCIZ <15><12>/ ADJUST R34 FOR A NULL /
(1)	010404	051525	020124	031522	
(1)	010412	020064	047506	020122	
(1)	010420	020101	052516	046114	
(1)	010426	020040	000		
6524	010431	015	020012	042101	ADJR35: .ASCIZ <15><12>/ ADJUST R35 FOR A NULL /
(1)	010436	052512	052123	051040	
(1)	010444	032463	043040	051117	
(1)	010452	040440	047040	046125	
(1)	010460	020114	000040		
6525	010464	005015	040440	045104	ADJR36: .ASCIZ <15><12>/ ADJUST R36 FOR A NULL /
(1)	010472	051525	020124	031522	
(1)	010500	020066	047506	020122	
(1)	010506	020101	052516	046114	
(1)	010514	020040	000		
6526	010517	015	020012	042101	ADJR37: .ASCIZ <15><12>/ ADJUST R37 FOR A NULL /
(1)	010524	052512	052123	051040	
(1)	010532	033463	043040	051117	
(1)	010540	040440	047040	046125	
(1)	010546	020114	000040		
6527	010552	005015	040440	045104	ADJR46: .ASCIZ <15><12>/ ADJUST R46 FOR A NULL /
(1)	010560	051525	020124	032122	
(1)	010566	020066	047506	020122	
(1)	010574	020101	052516	046114	
(1)	010602	020040	000		
6528	010605	015	020012	042101	ADJR47: .ASCIZ <15><12>/ ADJUST R47 FOR A NULL /
(1)	010612	052512	052123	051040	
(1)	010620	033464	043040	051117	
(1)	010626	040440	047040	046125	
(1)	010634	020114	000040		
6529	010640	005015	040440	045104	ADJR48: .ASCIZ <15><12>/ ADJUST R48 FOR A NULL /
(1)	010646	051525	020124	032122	

```

(1) 010654 020070 047506 020122
(1) 010662 020101 052516 046114
(1) 010670 020040 000
6530 010673 015 020012 042101 ADJR49: .ASCIZ <15><12>/ ADJUST R49 FOR A NULL /
(1) 010700 052512 052123 051040
(1) 010706 034464 043040 051117
(1) 010714 040440 047040 046125
(1) 010722 020114 000040
6531
6535 010726 005015 042523 042514 SELD0: .ASCIZ <15><12>/SELECT DAC0/
(1) 010734 052103 042040 041501
(1) 010742 000060
6536 010744 005015 042523 042514 SELD1: .ASCIZ <15><12>/SELECT DAC1/
(1) 010752 052103 042040 041501
(1) 010760 000061
6537 010762 005015 042523 042514 SELD2: .ASCIZ <15><12>/SELECT DAC2/
(1) 010770 052103 042040 041501
(1) 010776 000062
6538 011000 005015 042523 042514 SELD3: .ASCIZ <15><12>/SELECT DAC3/
(1) 011006 052103 042040 041501
(1) 011014 000063
6539 011016 005015 042101 052512 TRYAGN: .ASCIZ <15><12>/ADJUST THAT SAME POT AGAIN PLEASE/<15><12>
011024 052123 052040 040510
011032 020124 040523 042515
011040 050040 052117 040440
011046 040507 047111 050040
011054 042514 051501 006505
011062 000012
6540 000001
6541 000003
6542 011064 001
6543 011065 116 030465 030062 N51200: .BYTE STX
.ASCII /N512000V/
011072 030060 126
6544 011075 003 000
6545 011077 001
6546 011100 032520 030461 032467 P51175: .BYTE ETX,0
.ASCII /P511750V/
011106 053060
6547 011110 003 000
6548 .BYTE ETX,0
.EVEN
6549 011112 001116 001126 000000 DT1: $ERRPC,$BDDAT,0
6550 011120 001116 001422 001124 DT2: $ERRPC,DAC0,$GDDAT,$BDDAT,0
011126 001126 000000
6551 011132 001116 001424 001124 DT3: $ERRPC,DAC1,$GDDAT,$BDDAT,0
011140 001126 000000
6552 011144 001116 001426 001124 DT4: $ERRPC,DAC2,$GDDAT,$BDDAT,0
011152 001126 000000
6553 011156 001116 001430 001124 DT5: $ERRPC,DAC3,$GDDAT,$BDDAT,0
011164 001126 000000
6554 011170 001116 007002 001124 DT6: $ERRPC,DACBAD,$GDDAT,$BDDAT,SPREAD,0
011176 001126 007016 000000
6555 011204 000000 000000 000000 DF0: 0,0,0,0,0,0,0,0
011212 000000 000000 000000
011220 000000 000000

```

6556
 6557
 (1) .SBTTL BINARY TO ASCII AND TYPE ROUTINE


```

(1) 011370 001002      BNE      5$      ::FALL THROUGH IF 0
(1) 011372 105716      TSTB     (SP)    ::STILL DOING LEADING 0'S?
(1) 011374 100407      BMI      7$      ::BR IF YES
(1) 011376 106316      5$: ASLB     (SP)    ::MSD?
(1) 011400 103003      BCC      6$      ::BR IF NO
(1) 011402 116663      MOVB     1(SP),-1(R3) ::YES--SET THE SIGN
(1) 011410 052702      6$: BIS     #'0,R2  ::MAKE THE BCD DIGIT ASCII
(1) 011414 052702      7$: BIS     #' ,R2  ::MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 011420 110223      MOVB     R2,(R3)+  ::PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 011422 005720      TST      (R0)+   ::JUST INCREMENTING
(1) 011424 020027      000010  CMP     R0,#10   ::CHECK THE TABLE INDEX
(1) 011430 002746      BLT      2$      ::GO DO THE NEXT DIGIT
(1) 011432 003002      BGT      8$      ::GO TO EXIT
(1) 011434 010502      MOV      R5,R2   ::GET THE LSD
(1) 011436 000764      BR       6$      ::GO CHANGE TO ASCII
(1) 011440 105726      8$: TSTB     (SP)+  ::WAS THE LSD THE FIRST NON-ZERO?
(1) 011442 100003      BPL      9$      ::BR IF NO
(1) 011444 116663      177777 177776  MOVB     -1(SP),-2(R3) ::YES--SET THE SIGN FOR TYPING
(1) 011452 105013      9$: CLRB     (R3)   ::SET THE TERMINATOR
(3) 011454 012605      MOV      (SP)+,R5  ::POP STACK INTO R5
(3) 011456 012603      MOV      (SP)+,R3  ::POP STACK INTO R3
(3) 011460 012602      MOV      (SP)+,R2  ::POP STACK INTO R2
(3) 011462 012601      MOV      (SP)+,R1  ::POP STACK INTO R1
(3) 011464 012600      MOV      (SP)+,R0  ::POP STACK INTO R0
(1) 011466 104401      011514  TYPE     $DBLK  ::NOW TYPE THE NUMBER
(1) 011472 016666      000002 000004  MOV      2(SP),4(SP) ::ADJUST THE STACK
(1) 011500 012616      MOV      (SP)+,(SP)
(1) 011502 000002      RTI
(1) 011504 023420      $DTBL: 10000.
(1) 011506 001750      1000.
(1) 011510 000144      100.
(1) 011512 000012      10.
(1) 011514 000004      $DBLK: .BLKW 4
6559 .SBTTL SCOPE HANDLER ROUTINESTARS
(1) *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) *AND LOAD THE TEST NUMBER($STNM) 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) $SCOPE:
(1) 011524 104410      CKSWR
(1) 011526 104410      CKSWR ::TEST FOR CHANGE IN SOFT-SWR
(1) 011530 032777      040000 167402 1$: BIT     #BIT14,@SWR  ::LOOP ON PRESENT TEST?
(1) 011536 001114      BNE     $OVER    ::YES IF SW14=1
(1) 011540 000416      $XTSTR: BR     6$  ::IF RUNNING ON THE "XOR" TESTER CHANGE
(1) 011542 013746      000004  MOV     @#ERRVEC,-(SP) ::THIS INSTRUCTION TO A "NOP" (NOP=240)
(1) 011546 012737      011566 000004  MOV     #5$,@#ERRVEC  ::SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 011554 005737      177060  TST     @#177060     ::SET FOR TIMEOUT
(1) 011554 005737      177060  TST     @#177060     ::TIME OUT ON XOR?

```

```

(1) 011560 012637 000004      MOV      (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1) 011564 000463              BR      $$VLAD          ;;GO TO THE NEXT TEST
(1) 011566 022626      5$:    CMP      (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
(1) 011570 012637 000004      MOV      (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1) 011574 000423              BR      7$             ;;LOOP ON THE PRESENT TEST
(1) 011576              6$:;#####END OF CODE FOR THE XOR TESTER#####
(1) 011576 032777 000400 167334  BIT      #BIT08,@SWR    ;;LOOP ON SPEC. TEST?
(1) 011604 001404              BEQ      2$             ;;BR IF NO
(1) 011606 127737 167326 001102  CMPB    @SWR,$STNM     ;;ON THE RIGHT TEST?   SWR<7:0>
(1) 011614 001465              BEQ      $OVER        ;;BR IF YES
(1) 011616 105737 001103      2$:    TSTB    $ERFLG     ;;HAS AN ERROR OCCURRED?
(1) 011622 001421              BEQ      3$             ;;BR IF NO
(1) 011624 123737 001115 001103  CMPB    $ERMAX,$ERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 011632 101015              BHI      3$             ;;BR IF NO
(1) 011634 032777 001000 167276  BIT      #BIT09,@SWR    ;;LOOP ON ERROR?
(1) 011642 001404              BEQ      4$             ;;BR IF NO
(1) 011644 013737 001110 001106  7$:    MOV      $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 011652 000446              BR      $OVER
(1) 011654 105037 001103      4$:    CLRB    $ERFLG     ;;ZERO THE ERROR FLAG
(1) 011660 005037 001160      CLR     $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 011664 000415              BR      1$             ;;ESCAPE TO THE NEXT TEST
(1) 011666 032777 004000 167244  3$:    BIT      #BIT11,@SWR  ;;INHIBIT ITERATIONS?
(1) 011674 001011              BNE     1$             ;;BR IF YES
(1) 011676 005737 001202      TST     $PASS        ;;IF FIRST PASS OF PROGRAM
(1) 011702 001406              BEQ     1$             ;;INHIBIT ITERATIONS
(1) 011704 005237 001104      INC     $ICNT        ;;INCREMENT ITERATION COUNT
(1) 011710 023737 001160 001104  CMP     $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 011716 002024              BGE     $OVER        ;;BR IF MORE ITERATION REQUIRED
(1) 011720 012737 000001 001104  1$:    MOV     #1,$ICNT    ;;REINITIALIZE THE ITERATION COUNTER
(1) 011726 013737 012004 001160  MOV     $MXCNT,$TIMES  ;;SET NUMBER OF ITERATIONS TO DO
(1) 011734 105237 001102      $SVLAD: INCB   $STNM     ;;COUNT TEST NUMBERS
(1) 011740 113737 001102 001200  MOVB   $STNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1) 011746 011637 001106      MOV     (SP),$LPADR   ;;SAVE SCOPE LOOP ADDRESS
(1) 011752 011637 001110      MOV     (SP),$LPERR   ;;SAVE ERROR LOOP ADDRESS
(1) 011756 005037 001162      CLR     $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 011762 112737 000001 001115  MOVB   #1,$ERMAX     ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 011770 013777 001102 167144  $OVER: MOV     $STNM,@DISPLAY  ;;DISPLAY TEST NUMBER
(1) 011776 013716 001106      MOV     $LPADR,(SP)  ;;FUDGE RETURN ADDRESS
(1) 012002 000002              RTI                    ;;FIXES PS
(1) 012004 003720      $MXCNT: 2000.        ;;MAX. NUMBER OF ITERATIONS
6560 .SBTTL ERROR HANDLER ROUTINE

```

```

(1)
(2)
(1) *****
(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) *SW10=1      BELL ON ERROR
(1) *SW09=1      LOOP ON ERROR
(1) *CALL
(1) *          ERROR  N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1) $ERROR:
(1) 012006      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(1) 012006 104410

```

```

(2) 012010 053737 007010 007004      BIS      MASKNM,BADUNT
(1) 012016 105237 001103      7$:     INCB      $ERFLG      ;; SET THE ERROR FLAG
(1) 012022 001775          BEQ      7$              ;; DON'T LET THE FLAG GO TO ZERO
(1) 012024 013777 001102 167110      MOV      $TSTNM,@DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
(1) 012032 032777 002000 167100      BIT      #BIT10,@SWR     ;; BELL ON ERROR?
(1) 012040 001402          BEQ      1$              ;; NO - SKIP
(1) 012042 104401 001164          TYPE     ,SBELL         ;; RING BELL
(1) 012046 005237 001112      1$:     INC      $ERTTL      ;; COUNT THE NUMBER OF ERRORS
(1) 012052 011637 001116      MOV      (SP), $ERRPC    ;; GET ADDRESS OF ERROR INSTRUCTION
(1) 012056 162737 000002 001116      SUB      #2, $ERRPC
(1) 012064 117737 167026 001114      MOV      @ $ERRPC, $ITEMB  ;; STRIP AND SAVE THE ERROR ITEM CODE
(1) 012072 032777 020000 167040      BIT      #BIT13,@SWR     ;; SKIP TYPEOUT IF SET
(1) 012100 001004          BNE      20$           ;; SKIP TYPEOUTS
(1) 012102 004737 012202          JSR      PC, $ERRTYP     ;; GO TO USER ERROR ROUTINE
(1) 012106 104401 001171          TYPE     , $CRLF
(1) 012112          20$:
(1) 012112 122737 000001 001214      CMPB     #APTENV, $ENV    ;; RUNNING IN APT MODE
(1) 012120 001007          BNE      2$              ;; NO, SKIP APT ERROR REPORT
(1) 012122 113737 001114 012134      MOV      $ITEMB, 21$     ;; SET ITEM NUMBER AS ERROR NUMBER
(1) 012130 004737 014162          JSR      PC, $ATY4      ;; REPORT FATAL ERROR TO APT?
(1) 012134          21$:
(1) 012135          .BYTE    0
(1) 012136 000777          .BYTE    0
(1) 012140 005777 166774          22$:     BR      22$              ;; APT ERROR LOOP
(1) 012144 100002          2$:     TST      @SWR           ;; HALT ON ERROR
(1) 012146 000000          BPL      3$              ;; SKIP IF CONTINUE
(1) 012150 104410          HALT
(1) 012152 032777 001000 166760      CKSWR
(1) 012160 001402          3$:     BIT      #BIT09,@SWR    ;; TEST FOR CHANGE IN SOFT-SWR
(1) 012162 013716 001110          BEQ      4$              ;; LOOP ON ERROR SWITCH SET?
(1) 012166 005737 001162          MOV      $LPERR, (SP)   ;; BR IF NO
(1) 012172 001402          4$:     TST      $ESCAPE     ;; FUDGE RETURN FOR LOOPING
(1) 012174 013716 001162          BEQ      5$              ;; CHECK FOR AN ESCAPE ADDRESS
(1) 012200          MOV      $ESCAPE, (SP)  ;; BR IF NONE
(1) 012200 000002          5$:     MOV
RTI      ;; RETURN
6561 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)
(1)
(1)
(1)
(1)
(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) $ERRTYP:
(1) 012202 104401 001171          TYPE     , $CRLF         ;; "CARRIAGE RETURN" & "LINE FEED"
(1) 012206 010046          MOV      R0, -(SP)      ;; SAVE R0
(1) 012210 005000          CLR      R0             ;; PICKUP THE ITEM INDEX
(1) 012212 153700 001114          BISB     @ $ITEMB, R0
(1) 012216 001004          BNE      1$              ;; IF ITEM NUMBER IS ZERO, JUST
(1) 012220 013746 001116          MOV      $ERRPC, -(SP)  ;; TYPE THE PC OF THE ERROR
(2) 012224 104402          ;; SAVE $ERRPC FOR TYPEOUT
(2) 012226 000426          ;; ERROR ADDRESS
(1) 012230 005300          1$:     TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 012232 006300          BR      6$              ;; GET OUT
(1) 012234 006300          DEC      R0             ;; ADJUST THE INDEX SO THAT IT WILL
(1) 012234 006300          ASL      R0             ;; WORK FOR THE ERROR TABLE
(1) 012234 006300          ASL      R0

```

```

(1) 012236 006300 ASL R0
(1) 012240 062700 001256 ADD #SERRTB,R0
(1) 012244 012037 012254 MOV (R0)+,2$
(1) 012250 001404 BEQ 3$
(1) 012252 104401 TYPE
(1) 012254 000000 2$: .WORD 0
(1) 012256 104401 001171 TYPE ,SCLF
(1) 012262 012037 012272 3$: MOV (R0)+,4$
(1) 012266 001404 BEQ 5$
(1) 012270 104401 TYPE
(1) 012272 000000 4$: .WORD 0
(1) 012274 104401 001171 TYPE ,SCLF
(1) 012300 011000 5$: MOV (R0),R0
(1) 012302 001004 BNE 7$
(1) 012304 012600 6$: MOV (SP)+,R0
(1) 012306 104401 001171 TYPE ,SCLF
(1) 012312 000207 RTS PC
(1) 012314 7$:
(2) 012314 013046 MOV @ (R0)+,-(SP)
(2) 012316 104402 TYPOC
(1) 012320 005710 TST (R0)
(1) 012322 001770 BR 6$
(1) 012324 104401 012332 TYPE ,8$
(1) 012330 000771 BR 7$
(1) 012332 020040 000 8$: .ASCIZ / /
(1) 012336 012336 .EVEN

```

.SBTTL POWER DOWN AND UP ROUTINES

```

6562
(1)
(2)
(1)
*****
:POWER DOWN ROUTINE
$PWRDN: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST UP
MOV #PR6,@#PWRVEC+2 ;;PRIO:6
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
MOV SP,$SAVR6 ;;SAVE SP
MOV #S$PWRUP,@#PWRVEC ;;SET UP VECTOR
HALT
BR -2 ;;HANG UP

```

```

(1)
(2)
*****
:POWER UP ROUTINE
$PWRUP: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
MOV $SAVR6,SP ;;GET SP
CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
1$: INC $SAVR6 ;;WAIT FOR THE INC
BNE 1$ ;;OF WORD
MOV (SP)+,@SWR ;;POP STACK INTO @SWR
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3

```

```

(3) 012446 012602                    MOV      (SP)+,R2                    ::POP STACK INTO R2
(3) 012450 012601                    MOV      (SP)+,R1                    ::POP STACK INTO R1
(3) 012452 012600                    MOV      (SP)+,R0                    ::POP STACK INTO R0
(1) 012454 012737      012336      000024      MOV      #$PWRDN,@#PWRVEC            ::SET UP THE POWER DOWN VECTOR
(1) 012462 012737      000300      000026      MOV      #PR6,@#PWRVEC+2            ::PRIO:6
(1) 012470 104401                    TYPE                                ::REPORT THE POWER FAILURE
(1) 012472 012510                    $PWRMG: .WORD      PWRMSG                    ::POWER FAIL MESSAGE POINTER
(1) 012474 012716                    MOV      (PC)+,(SP)                ::RESTART AT BEGIN
(1) 012476 001450                    $PWRAD: .WORD      BEGIN                    ::RESTART ADDRESS
(1) 012500 000002                    RTI                                ::
(1) 012502 000000                    $ILLUP: HALT                    ::THE POWER UP SEQUENCE WAS STARTED
(1) 012504 000776                    BR      .-2                    :: BEFORE THE POWER DOWN WAS COMPLETE
(1) 012506 000000                    $$SAVR6: 0                    ::PUT THE SP HERE
6563 012510 005015      042522      052123      PWRMSG: .ASCIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>
      012516 051101      044524      043516
      012524 040440      052106      051105
      012532 040440      050040      053517
      012540 051105      043040      044501
      012546 052514      042522      005015
      012554 000012

```

```

6564                    .EVEN
6565                    .SBTTL    BINARY TO OCTAL (ASCII) AND TYPE
6566                    ::*****
(1)                    ::THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                    ::OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                    ::$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                    ::CALL:
(1)                    ::*      MOV      NUM,-(SP)            ::NUMBER TO BE TYPED
(1)                    ::*      TYPOS                    ::CALL FOR TYPEOUT
(1)                    ::*      .BYTE      N                ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                    ::*      .BYTE      M                ::M=1 OR 0
(1)                    ::*                                ::1=TYPE LEADING ZEROS
(1)                    ::*                                ::0=SUPPRESS LEADING ZEROS
(1)                    ::$STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                    ::$TYPOS OR $TYPOC
(1)                    ::CALL:
(1)                    ::*      MOV      NUM,-(SP)            ::NUMBER TO BE TYPED
(1)                    ::*      TYPON                    ::CALL FOR TYPEOUT
(1)                    ::$STYPOC---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) 012556 017646      000000      C:3001      $TYPOS: MOV      @ (SP),-(SP)            ::PICKUP THE MODE
(1) 012562 116637      000001                    MOVB      1(SP),$OFILL            ::LOAD ZERO FILL SWITCH
(1) 012570 112637      013003                    MOVB      (SP)+,$SOMODE+1            ::NUMBER OF DIGITS TO TYPE
(1) 012574 062716      000002                    ADD      #2,(SP)                ::ADJUST RETURN ADDRESS
(1) 012600 000406                    BR      $TYPON
(1) 012602 112737      000001      013001      $TYPOC: MOVB      #1,$OFILL            ::SET THE ZERO FILL SWITCH
(1) 012610 112737      000006      013003                    MOVB      #6,$SOMODE+1            ::SET FOR SIX(6) DIGITS
(1) 012616 112737      000005      013000      $TYPON: MOVB      #5,$OCNT            ::SET THE ITERATION COUNT
(1) 012624 010346                    MOV      R3,-(SP)                ::SAVE R3

```

```

(1) 012626 010446      MOV      R4,-(SP)      ;;SAVE R4
(1) 012630 010546      MOV      R5,-(SP)      ;;SAVE R5
(1) 012632 113704 013003  MOVB     $OMODE+1,R4   ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 012636 005404      NEG      R4
(1) 012640 062704 000006  ADD      #6,R4        ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 012644 110437 013002  MOVB     R4,$OMODE     ;;SAVE IT FOR USE
(1) 012650 113704 013001  MOVB     $OFILL,R4     ;;GET THE ZERO FILL SWITCH
(1) 012654 016605 000012  MOV      12(SP),R5     ;;PICKUP THE INPUT NUMBER
(1) 012660 005003      CLR      R3           ;;CLEAR THE OUTPUT WORD
(1) 012662 006105      1$:     ROL      R5     ;;ROTATE MSB INTO 'C'
(1) 012664 000404      BR      3$           ;;GO DO MSB
(1) 012666 006105      2$:     ROL      R5     ;;FORM THIS DIGIT
(1) 012670 006105      ROL      R5
(1) 012672 006105      ROL      R5
(1) 012674 010503      MOV      R5,R3
(1) 012676 006103      3$:     ROL      R3     ;;GET LSB OF THIS DIGIT
(1) 012700 105337 013002  DECB     $OMODE       ;;TYPE THIS DIGIT?
(1) 012704 100016      BPL      7$           ;;BR IF NO
(1) 012706 042703 177770  BIC      #177770,R3   ;;GET RID OF JUNK
(1) 012712 001002      BNE      4$           ;;TEST FOR 0
(1) 012714 005704      TST      R4           ;;SUPPRESS THIS 0?
(1) 012716 001403      BEQ      5$           ;;BR IF YES
(1) 012720 005204      4$:     INC      R4     ;;DON'T SUPPRESS ANYMORE 0'S
(1) 012722 052703 000060  BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
(1) 012726 052703 000040  5$:     BIS      #' ,R3  ;;MAKE ASCII IF NOT ALREADY
(1) 012732 110337 012776  MOVB     R3,8$       ;;SAVE FOR TYPING
(1) 012736 104401 012776  TYPE     ,8$         ;;GO TYPE THIS DIGIT
(1) 012742 105337 013000  7$:     DECB     $OCNT  ;;COUNT BY 1
(1) 012746 003347      BGT      2$           ;;BR IF MORE TO DO
(1) 012750 002402      BLT      6$           ;;BR IF DONE
(1) 012752 005204      INC      R4           ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 012754 000744      BR      2$           ;;GO DO THE LAST DIGIT
(1) 012756 012605      6$:     MOV      (SP)+,R5  ;;RESTORE R5
(1) 012760 012604      MOV      (SP)+,R4     ;;RESTORE R4
(1) 012762 012603      MOV      (SP)+,R3     ;;RESTORE R3
(1) 012764 016666 000002 000004  MOV      2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
(1) 012772 012616      MOV      (SP)+,(SP)
(1) 012774 000002      RTI
(1) 012776 000      8$:     .BYTE   0        ;;RETURN
(1) 012777 000      .BYTE   0        ;;STORAGE FOR ASCII DIGIT
(1) 013000 000      $OCNT:  .BYTE   0        ;;TERMINATOR FOR TYPE ROUTINE
(1) 013001 000      $OFILL: .BYTE   0        ;;OCTAL DIGIT COUNTER
(1) 013002 000000  $OMODE: .WORD   0        ;;ZERO FILL SWITCH
(1) 013002 000000  .SBTTL  TYPE ROUTINE ;;NUMBER OF DIGITS TO TYPE

(1)
(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)          ;*
(1) 013004 105737 001157 $TYPE: TSTB $TFPLG      ;; IS THERE A TERMINAL?
(1) 013010 100002          BPL      1$          ;; BR IF YES
(1) 013012 000000          HALT          ;; HALT HERE IF NO TERMINAL
(1) 013014 000430          BR      3$          ;; LEAVE
(1) 013016 010046          1$:  MOV      RO,-(SP)      ;; SAVE RO
(1) 013020 017600 000002  MOV      @2(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
(1) 013024 122737 000001 001214  CMPB   #APTENV,$ENV      ;; RUNNING IN APT MODE
(1) 013032 001011          BNE      62$          ;; NO,GO CHECK FOR APT CONSOLE
(1) 013034 132737 000100 001215  BITB   #APTSPOOL,$ENVM    ;; SPOOL MESSAGE TO APT
(1) 013042 001405          BEQ      62$          ;; NO,GO CHECK FOR CONSOLE
(1) 013044 010037 013054  MOV      RO,61$          ;; SETUP MESSAGE ADDRESS FOR APT
(1) 013050 004737 014152  JSR    PC,$ATY3          ;; SPOOL MESSAGE TO APT
(1) 013054 000000          61$:  .WORD   0          ;; MESSAGE ADDRESS
(1) 013056 132737 000040 001215 62$:  BITB   #APTCSUP,$ENVM    ;; APT CONSOLE SUPPRESSED
(1) 013064 001003          BNE      60$          ;; YES,SKIP TYPE OUT
(1) 013066 112046          2$:  MOVB   (RO)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 013070 001005          BNE      4$          ;; BR IF IT ISN'T THE TERMINATOR
(1) 013072 005726          TST    (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
(1) 013074 012600          60$:  MOV      (SP)+,RO      ;; RESTORE RO
(1) 013076 062716 000002  3$:  ADD      #2,(SP)          ;; ADJUST RETURN PC
(1) 013102 000002          RTI          ;; RETURN
(1) 013104 122716 000011  4$:  CMPB   #HT,(SP)          ;; BRANCH IF <HT>
(1) 013110 001430          BEQ      8$          ;; BRANCH IF NOT <CRLF>
(1) 013112 122716 000200  CMPB   #CRLF,(SP)
(1) 013116 001006          BNE      5$          ;; POP <CR><LF> EQUIV
(1) 013120 005726          TST    (SP)+          ;; TYPE A CR AND LF
(1) 013122 104401          TYPE
(1) 013124 001171          $CRLF
(1) 013126 105037 013262  CLRB   $CHARCNT          ;; CLEAR CHARACTER COUNT
(1) 013132 000755          BR      2$          ;; GET NEXT CHARACTER
(1) 013134 004737 013216  5$:  JSR    PC,$TYPEC          ;; GO TYPE THIS CHARACTER
(1) 013140 123726 001156  6$:  CMPB   $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
(1) 013144 001350          BNE      2$          ;; IF NO GO GET NEXT CHAR.
(1) 013146 013746 001154  MOV      $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
(1)          ;; AND THE NULL CHAR.
(1) 013152 105366 000001  7$:  DECB   1(SP)          ;; DOES A NULL NEED TO BE TYPED?
(1) 013156 002770          BLT    6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 013160 004737 013216  JSR    PC,$TYPEC          ;; GO TYPE A NULL
(1) 013164 105337 013262  DECB   $CHARCNT          ;; DO NOT COUNT AS A COUNT
(1) 013170 000770          BR      7$          ;; LOOP
(1)
(1)          ;HORIZONTAL TAB PROCESSOR
(1)
(1) 013172 112716 000040  8$:  MOVB   #' ,(SP)          ;; REPLACE TAB WITH SPACE
(1) 013176 004737 013216  9$:  JSR    PC,$TYPEC          ;; TYPE A SPACE
(1) 013202 132737 000007 013262  BITB   #7,$CHARCNT          ;; BRANCH IF NOT AT
(1) 013210 001372          BNE      9$          ;; TAB STOP
(1) 013212 005726          TST    (SP)+          ;; POP SPACE OFF STACK
(1) 013214 000724          BR      2$          ;; GET NEXT CHARACTER
(1) 013216 105777 165726  $TYPEC: TSTB   @STPS          ;; WAIT UNTIL PRINTER IS READY
(1) 013222 100375          BPL    $TYPEC
(1) 013224 116677 000002 165720  MOVB   2(SP),@STPB      ;; LOAD CHAR TO BE TYPED INTO DATA REG.

```

```

(1)
(1) 013232 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
(1) 013240 001003 BNE 1$ ;;BRANCH IF NO
(1) 013242 105037 013262 CLR B $CHARCNT ;;YES--CLEAR CHARACTER COUNT
(1) 013246 000406 BR $TYPEX ;;EXIT
(1) 013250 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
(1) 013256 001402 BEQ $TYPEX ;;BRANCH IF YES
(1) 013260 105227 INCB (PC)+ ;;COUNT THE CHARACTER
(1) 013262 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
(1) 013264 000207 $TYPEX: RTS PC

(1)
(1)
(1)
6568 .SBTTL TTY INPUT ROUTINE
(1)
(2) ;;*****
(1)
(1)
(1) .ENABL LSB
(1)
(2) ;;*****
(1) ;;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ;;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ;;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1) ;;*WHEN OPERATING IN TTY FLAG MODE.
(1) 013266 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
(1) 013274 001074 BNE 15$ ;;BRANCH IF NO
(1) 013276 105777 165642 TSTB @STKS ;;CHAR THERE?
(1) 013302 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
(1) 013304 117746 165636 MOVB @STKB, -(SP) ;;SAVE THE CHAR
(1) 013310 042716 177600 BIC #^C177, (SP) ;;STRIP-OFF THE ASCII
(1) 013314 022726 000007 CM? #7, (SP)+ ;;IS IT A CONTROL G?
(1) 013320 001062 BNE 15$ ;;NO, RETURN TO USER
(1) 013322 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
(1) 013330 001456 BEQ 15$ ;;BRANCH IF YES
(1)
(1) 013332 104401 014013 $GTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (^G)
(1) 013336 104401 014020 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
(2) 013342 013746 000176 MOV SWREG, -(SP) ;;SAVE SWREG FOR TYPEOUT
(2) 013346 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 013350 104401 014031 TYPE , $MNEW ;;PROMPT FOR NEW SWR
(1) 013354 005046 19$: CLR -(SP) ;;CLEAR COUNTER
(1) 013356 005046 CLR -(SP)
(1) ;;THE NEW SWR
(1) 013360 105777 165560 7$: TSTB @STKS ;;CHAR THERE?
(1) 013364 100375 BPL 7$ ;;IF NOT TRY AGAIN
(1)
(1) 013366 117746 165554 MOVB @STKB, -(SP) ;;PICK UP CHAR
(1) 013372 042716 177600 BIC #^C177, (SP) ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1) 013376 021627 000025 9$: CMP (SP), #25 ;;IS IT A CONTROL-U?
(1) 013402 001005 BNE 10$ ;;BRANCH IF NOT
(1) 013404 104401 014006 TYPE , $CNTLU ;;YES, ECHO CONTROL-U (^U)
(1) 013410 062706 000006 20$: ADD #6, SP ;;IGNORE PREVIOUS INPUT

```



```

(1) 013414 000757 BR 19$ ;;LET'S TRY IT AGAIN
(1)
(1)
(1) 013416 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
(1) 013422 001022 BNE 16$ ;;BRANCH IF NO
(1) 013424 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
(1) 013430 001403 BEQ 11$ ;;BRANCH IF YES
(1) 013432 016677 000002 165500 MOV 2(SP),@SWR ;;SAVE NEW SWR
(1) 013440 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
(1) 013444 104401 001171 14$: TYPE $,SCLRF ;;ECHO <CR> AND <LF>
(1) 013450 123727 001135 000001 CMPB $,INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
(1) 013456 001003 BNE 15$ ;;BRANCH IF NOT
(1) 013460 012777 000100 165456 MOV #100,@$TKS ;;RE-ENABLE TTY KBD INTERRUPTS
(1) 013466 000002 15$: RTI ;;RETURN
(1) 013470 004737 013216 16$: JSR PC,$TYPEC ;;ECHO CHAR
(1) 013474 021627 000060 CMP (SP),#60 ;;CHAR < 0?
(1) 013500 002420 BLT 18$ ;;BRANCH IF YES
(1) 013502 021627 000067 CMP (SP),#67 ;;CHAR > 7?
(1) 013506 003015 BGT 18$ ;;BRANCH IF YES
(1) 013510 042726 000060 BIC #60,(SP)+ ;;STRIP-OFF ASCII
(1) 013514 005766 000002 TST 2(SP) ;;IS THIS THE FIRST CHAR
(1) 013520 001403 BEQ 17$ ;;BRANCH IF YES
(1) 013522 006316 ASL (SP) ;;NO, SHIFT PRESENT
(1) 013524 006316 ASL (SP) ;; CHAR OVER TO MAKE
(1) 013526 006316 ASL (SP) ;; ROOM FOR NEW ONE.
(1) 013530 005266 000002 17$: INC 2(SP) ;;KEEP COUNT OF CHAR
(1) 013534 056616 177776 BIS -2(SP),(SP) ;;SET IN NEW CHAR
(1) 013540 000707 BR 7$ ;;GET THE NEXT ONE
(1) 013542 104401 001170 18$: TYPE $,SQUES ;;TYPE ?<CR><LF>
(1) 013546 000720 BR 20$ ;;SIMULATE CONTROL-U
(1) .DSABL LSB
(1)
(1)
(1)
(2)
(1) *****
(1) *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) *CALL:
(1) * RDCHR ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1) * RETURN HERE ;;CHARACTER IS ON THE STACK
(1) * ;;WITH PARITY BIT STRIPPED OFF
(1)
(1)
(1) $RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
(1) 013550 011646 000004 000002 MOV 4(SP),2(SP) ;;SAVE THE PS
(1) 013552 016666 165360 1$: TSTB @$TKS ;;WAIT FOR
(1) 013560 105777 165360 BPL 1$ ;;A CHARACTER
(1) 013564 100375 165354 000004 MOVB @$TKB,4(SP) ;;READ THE TTY
(1) 013566 117766 177600 000004 BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
(1) 013574 042766 000004 000023 CMP 4(SP),#23 ;;IS IT A CONTROL-S?
(1) 013602 026627 000004 BNE 3$ ;;BRANCH IF NO
(1) 013610 001013 165326 2$: TSTB @$TKS ;;WAIT FOR A CHARACTER
(1) 013612 105777 165326 BPL 2$ ;;LOOP UNTIL ITS THERE
(1) 013616 100375 165322 MOVB @$TKB,-(SP) ;;GET CHARACTER
(1) 013620 117746 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 013624 042716 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
(1) 013630 022627 000021 BNE 2$ ;;IF NOT DISCARD IT
(1) 013634 001366 BR 1$ ;;YES, RESUME
(1) 013636 000750
    
```

```

(1) 013640 026627 000004 000140 3$: CMP 4(SP),#140 ;;IS IT UPPER CASE?
(1) 013646 002407 BLT 4$ ;;BRANCH IF YES
(1) 013650 026627 000004 000175 CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
(1) 013656 003003 BGT 4$ ;;BRANCH IF YES
(1) 013660 042766 000040 000004 BIC #40,4(SP) ;;MAKE IT UPPER CASE
(1) 013666 000002 4$: RTI ;;GO BACK TO USER
(2)
(1)
(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) 013670 010346 $RDLIN: MOV R3,-(SP) ;;SAVE R3
(1) 013672 012703 013776 1$: MOV #$TTYIN,R3 ;;GET ADDRESS
(1) 013676 022703 014006 2$: CMP #$TTYIN+8.,R3 ;;BUFFER FULL?
(1) 013702 101405 BLOS 4$ ;;BR IF YES
(1) 013704 104411 RDCHR ;;GO READ ONE CHARACTER FROM THE TTY
(1) 013706 112613 MOV (SP)+,(R3) ;;GET CHARACTER
(1) 013710 122713 000177 10$: CMPB #177,(R3) ;;IS IT A RUBOUT
(1) 013714 001003 BNE 3$ ;;SKIP IF NOT
(1) 013716 104401 001170 4$: TYPE ,SQUES ;;TYPE A '?'
(1) 013722 000763 BR 1$ ;;CLEAR THE BUFFER AND LOOP
(1) 013724 111337 013774 3$: MOV (R3),9$ ;;ECHO THE CHARACTER
(1) 013730 104401 013774 TYPE ,9$
(1) 013734 122723 000015 CMPB #15,(R3)+ ;;CHECK FOR RETURN
(1) 013740 001356 BNE 2$ ;;LOOP IF NOT RETURN
(1) 013742 105063 177777 CLRB -1(R3) ;;CLEAR RETURN (THE 15)
(1) 013746 104401 001172 TYPE ,SLF ;;TYPE A LINE FEED
(1) 013752 012603 MOV (SP)+,R3 ;;RESTORE R3
(1) 013754 011646 MOV (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 013756 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 013764 012766 013776 000004 MOV #$TTYIN,4(SP)
(1) 013772 000002 RTI ;;RETURN
(1) 013774 000 9$: .BYTE 0 ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 013775 000 .BYTE 0 ;;TERMINATOR
(1) 013776 000010 $TTYIN: .BLKB 8. ;;RESERVE 8 BYTES FOR TTY INPUT
(1) 014006 052536 005015 000 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
(1) 014013 136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
(1) 014020 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 014026 020075 000
(1) 014031 040 047040 053505 $MNEW: .ASCIZ / NEW = /
(1) 014036 036440 000040
6569
(2)
(1)
(1) *SBTTL READ AN OCTAL NUMBER FROM THE TTY
(1) *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1) *CHANGE IT TO BINARY.
(1) *CALL:
(1) * RDOCT ;;READ AN OCTAL NUMBER
(1) * RETURN HERE ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1) *BITS ARE IN $SHIOCT ;;HIGH ORDER
(1) 014042 011646 $RDOCT: MOV (SP),-(SP) ;;PROVIDE SPACE FOR THE
(1) 014044 016666 000004 000002 MOV 4(SP),2(SP) ;;INPUT NUMBER
(3) 014052 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 014054 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
    
```

```

(3) 014056 010246
(1) 014060 104412
(1) 014062 012600
(1) 014064 005001
(1) 014066 005002
(1) 014070 112046
(1) 014072 001412
(1) 014074 006301
(1) 014076 006102
(1) 014100 006301
(1) 014102 006102
(1) 014104 006301
(1) 014106 006102
(1) 014110 042716 177770
(1) 014114 062601
(1) 014116 000764
(1) 014120 005726
(1) 014122 010166 000012
(1) 014126 010237 014142
(3) 014132 012602
(3) 014134 012601
(3) 014136 012600
(1) 014140 000002
(1) 014142 000000
6570
(1)
(2)
(1) 014144 112737 000001 014410
(1) 014152 112737 000001 014406
(1) 014160 000403
(1) 014162 112737 000001 014410
(1) 014170
(3) 014170 010046
(3) 014172 010146
(1) 014174 105737 014406
(1) 014200 001450
(1) 014202 122737 000001 001214
(1) 014210 001031
(1) 014212 132737 000100 001215
(1) 014220 001425
(1) 014222 017600 000004
(1) 014226 062766 000002 000004
(1) 014234 005737 001174
(1) 014240 001375
(1) 014242 010037 001210
(1)
(1) 014246 105720
(1) 014250 001376
(1) 014252 163700 001210
(1) 014256 006200
(1) 014260 010037 001212
(1) 014264 012737 000004 001174
(1) 014272 000413
(1) 014274 017637 000004 014320
(1) 014302 062766 000002 000004
(3) 014310 013746 177776
  
```

```

MOV R2,-(SP) ::PUSH R2 ON STACK
1$: RDLIN ::READ AN ASCIZ LINE
MOV (SP)+,R0 ::GET ADDRESS OF 1ST CHARACTER
CLR R1 ::CLEAR DATA WORD
CLR R2
2$: 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 #^C7,(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 #APTPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
BEQ 3$ ::IF NOT: BR
MOV @4(SP),R0 ::GET MESSAGE ADDR.
ADD #2,4(SP) ::BUMP RETURN ADDR.
1$: TST $MSGTYPE ::SEE IF DONE W/ LAST XMISSION?
BNE 1$ ::IF NOT: WAIT
MOV R0,$MSGAD
::PUT ADDR IN MAILBOX
2$: TSTB (R0)+ ::FIND END OF MESSAGE
BNE 2$
SUB $MSGAD,R0 ::SUB START OF MESSAGE
ASR R0 ::GET MESSAGE LNTH IN WORDS
MOV R0,$MSGGLT ::PUT LENGTH IN MAILBOX
MOV #4,$MSGTYPE ::TELL APT TO TAKE MSG.
BR 5$
3$: MOV @4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
ADD #2,4(SP) ::BUMP RETURN ADDRESS
MOV 177776,-(SP) ::PUSH 177776 ON STACK
  
```

```

(1) 014314 004737 013004      JSR    PC,$TYPE      ;;CALL TYPE MACRO
(1) 014320 000000      4$:    .WORD    0
(1) 014322 105737 014410      5$:    TSTB    $FFLG      ;;SHOULD REPORT FATAL ERROR?
(1) 014326 001416      10$:   BEQ     12$      ;;IF NOT: BR
(1) 014330 005737 001214      TST    $ENV      ;;RUNNING UNDER APT?
(1) 014334 001413      BEQ    12$      ;;IF NOT: BR
(1) 014336 005737 001174      11$:   TST    $MSGTYPE  ;;FINISHED LAST MESSAGE?
(1) 014342 001375      BNE    11$      ;;IF NOT: WAIT
(1) 014344 017637 000004 001176  MOV    @4(SP),$FATAL  ;;GET ERROR #
(1) 014352 062766 000002 000004  ADD    #2,4(SP)      ;;BUMP RETURN ADDR.
(1) 014360 005237 001174      INC    $MSGTYPE     ;;TELL APT TO TAKE ERROR
(1) 014364 105037 014410      12$:   CLRB   $FFLG      ;;CLEAR FATAL FLAG
(1) 014370 105037 014407      CLRB   $LFLG      ;;CLEAR LOG FLAG
(1) 014374 105037 014406      CLRB   $MFLG      ;;CLEAR MESSAGE FLAG
(3) 014400 012601      MOV    (SP)+,R1     ;;POP STACK INTO R1
(3) 014402 012600      MOV    (SP)+,R0     ;;POP STACK INTO R0
(1) 014404 000207      RTS    PC           ;;RETURN
(1) 014406 000      $MFLG: .BYTE    0   ;;MESSG. FLAG
(1) 014407 000      $LFLG: .BYTE    0
(1) 014410 000      ;;LOG FLAG
(1) 014412 000      $FFLG: .BYTE    0   ;;FATAL FLAG
(1) 000200      .EVEN
(1) 000001      APTSIZE=200
(1) 000100      APTENV=001
(1) 000040      APTSPool=100
(1) 000040      APTCSUP=040
  
```

6571

6573

6574

(1)

(2)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(3)

(3)

(3)

(3)

(3)

(3)

(3)

(3)

(3)

(3)

(3)

(3)

(1)

(3)

(1)

(3)

(3)

(3)

(3)

6575

6576

6577

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

014412 010046
 014414 016600 000002
 014420 005740
 014422 111000
 014424 006300
 014426 016000 014446
 014432 000200

014434 011646
 014436 016666 000004 000002
 014444 000002

014446 014434
 014450 013004
 014452 012602
 014454 012556
 014456 012616
 014460 011300
 014462 011224
 014464 013336
 014466 013266
 014470 013550
 014472 013670
 014474 014042

014476
 000100
 000102 000300
 000140
 000140 170000
 000142 000300
 014476
 014476 104401 014504

.SBTTL TRAP DECODER

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

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

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

STRAP2: MOV (SP), -(SP) ;;MOVE THE PC DOWN
 MOV 4(SP), 2(SP) ;;MOVE THE PSW DOWN
 RTI ;;RESTORE THE PSW

.SBTTL TRAP TABLE

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

ROUTINE	STARTING ADDRESS	DESCRIPTION
\$TRPAD	TRAP+1(104401)	TTY TYPEOUT ROUTINE
\$TYPE	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOC	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPOS	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPON	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
\$TYPDS	TRAP+6(104406)	TYPE BINARY (ASCII) NUMBER
\$TYPBN		
\$GTSWR	TRAP+7(104407)	GET SOFT-SWR SETTING
\$CKSWR	TRAP+10(104410)	TEST FOR CHANGE IN SOFT-SWR
\$RDCHR	TRAP+11(104411)	TTY TYPEIN CHARACTER ROUTINE
\$RDLIN	TRAP+12(104412)	TTY TYPEIN STRING ROUTINE
\$RDOCT	TRAP+13(104413)	READ AN OCTAL NUMBER FROM TTY

;;THE FOLLOWING MACRO CALL WAS ADDED TO INIT 11/21 SPECIFIC VECTORS.
 POINT=.;
 =100 ;SAVE POINTER

\$CLKVEC ;LKVEC HANDLER
 300 ;INTERRUPT HANDLER PRI
 =140 ;BRKVEC
 170000 ;ODT START ADDRESS
 300 ;PRIORITY
 =POINT ;RESTORE POINTER
 \$CLKVEC: TYPE, CLKMES

(1) 014502 000000
(1) 014504 005015 045514 042526 CLKMES: .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
(1) 014512 020103 047111 042524
(1) 014520 051122 050125 020124
(1) 014526 020055 044504 041523
(1) 014534 047117 042516 052103
(1) 014542 046040 041524 000040
6578
6579 000001 .END

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
ABASE	= 174440	5690#	5713	5793	5794	5795	5796 6468
ACDW1	= 000000	5713					
ACDW2	= 000000	5713					
ACPUOP	= 000000	5713					
ADBR	007030	6351	6479#				
ADCS	007026	6345*	6348*	6349	6478#		
ADDOK	001440	5701	5829#				
ADDW0	= 000000	5713					
ADDW1	= 000000	5713					
ADDW10	= 000000	5713					
ADDW11	= 000000	5713					
ADDW12	= 000000	5713					
ADDW13	= 000000	5713					
ADDW14	= 000000	5713					
ADDW15	= 000000	5713					
ADDW2	= 000000	5713					
ADDW3	= 000000	5713					
ADDW4	= 000000	5713					
ADDW5	= 000000	5713					
ADDW6	= 000000	5713					
ADDW7	= 000000	5713					
ADDW8	= 000000	5713					
ADDW9	= 000000	5713					
ADEVCT	= 000000	5713					
ADEVN	= 000000	5713					
ADJR34	010376	6212	6276	6523#			
ADJR35	010431	6215	6524#				
ADJR36	010464	6218	6525#				
ADJR37	010517	6221	6526#				
ADJR46	010552	6212	6250	6527#			
ADJR47	010605	6215	6528#				
ADJR48	010640	6218	6529#				
ADJR49	010673	6221	6530#				
AENV	= 000000	5713					
AENVN	= 000000	5713					
AFATAL	= 000000	5713					
AMADR1	= 000000	5713					
AMADR2	= 000000	5713					
AMADR3	= 000000	5713					
AMADR4	= 000000	5713					
AMAMS1	= 000000	5713					
AMAMS2	= 000000	5713					
AMAMS3	= 000000	5713					
AMAMS4	= 000000	5713					
AMSGAD	= 000000	5713					
AMSGLG	= 000000	5713					
AMSGTY	= 000000	5713					
AMTYP1	= 000000	5713					
AMTYP2	= 000000	5713					
AMTYP3	= 000000	5713					
AMTYP4	= 000000	5713					
APASS	= 000000	5713					
APRIOR	= 000000	5713					
APTCSU	= 000040	6567	6570#				
APTENV	= 000001	6560	6567	6570#			
APTSIZ	= 000200	5837	6570#				

MAINDEC-11-CNA ⁵ AA-A		AAV11	DIAGNOSTIC	MACY11 30(1046)		27-DEC-82 14:27		PAGE 86-2		SEQ 0064		
CNA ⁵ AA.P11		27-DEC-82 14:25	CROSS REFERENCE TABLE --	USER SYMBOLS								
DDISP =	177570	5688#	5713	5837								
DF0	011204	5721	5727	5733	5739	5745	5751	5759	5765	5771	5777	5783
DH1	010267	5725	5731	5737	5743	5781	6517#					6555#
DH2	010322	5719	6518#									
DH6	010337	5749	5757	5763	5769	5775	6519#					
DISPLA	001142	5713#	5837*	6559*	6560*							
DISPRE	000174	5695#	5837									
DRIN	007024	6182	6477#									
DSWR =	177570	5688#	5713	5837								
DT1	011112	5720	6549#									
DT2	011120	5726	6550#									
DT3	011132	5732	6551#									
DT4	011144	5738	6552#									
DT5	011156	5744	5782	6553#								
DT6	011170	5750	5758	5764	5770	5776	6554#					
DYNCAL	006716	5698	6450#									
EMTVEC=	000030	5688#	5837*									
EM1	007122	5718	6496#									
EM10	007472	5762	6503#									
EM11	007537	5768	6504#									
EM12	007574	5774	6505#									
EM13	007631	5780	6506#									
EM14	007675	5786	6507#									
EM2	007176	5724	6497#									
EM3	007225	5730	6498#									
EM4	007254	5736	6499#									
EM5	007303	5742	6500#									
EM6	007332	5748	6501#									
EM7	007413	5756	6502#									
ERRTOT	010176	6228	6512#									
ERRVEC=	000004	5688#	5837*	5889*	5928*	5936*	5938*	6559*				
ETX =	000003	6541#	6544	6547								
EVER	001420	5792#	5834*	5903	5915*	5916*	5918	6135				
FILZ	007022	6321*	6325*	6329*	6333*	6476#						
FOUND1	010224	5908	6514#									
FOUND2	010250	5914	6516#									
FULRMP	006570	5696	6416#									
GAIDAC	005764	6212	6215	6218	6221	6267#						
GNS =	***** U	5695	6574									
GTSWR =	104407	6574#										
HT =	000011	5688#	6567									
INIT1	002044	5839	5871#	6233								
INIT7	005576	6224	6226#									
IOTVEC=	000020	5688#	5837*									
LDSPAC	007746	6367	6508#									
LDTRAP	001746	5844#	5871	5923	6417	6439	6451					
LF =	000012	5688#	6567									
LKVEC =	000100	5688#										
MASKNM	007010	5887*	5924*	6142*	6471#	6560						
MESGD	010210	6231	6513#									
MSGSW	010011	5879	6509#									
MTEST	002102	5873	5875	5886#								
NUMBOK	007006	5829*	5831*	5895	6470#							
N51200	011064	6247	6542#									
ODTST =	170000	5688#										
OFFDAC	005634	6212	6215	6218	6221	6238#						

SW7 = 000200	5688#			
SW8 = 000400	5688#			
SW9 = 001000	5688#			
TBITVE= 000014	5688#			
TEMP 007012	5833*	5872	6123*	6472#
TESTER 001432	5704	5827#		
TITLE 007040	5881	6495#		
TKVEC = 000060	5688#			
TPVEC = 000064	5688#			
TRAPVE= 000034	5688#	5837*		
TRTVEC= 000014	5688#			
TRYAGN 011016	6260	6286	6539#	
TST1 002306	5902	5927#	6144	
TST10 002752	5973	5975#		
TST11 003010	5980	5984#		
TST12 003062	5993#			
TST13 003154	5994#			
TST14 003250	5997#			
TST15 003304	6002	6005#		
TST16 003342	6010	6013#		
TST17 003414	6023#			
TST2 002364	5939#			
TST20 003506	6026#			
TST21 003602	6028#			
TST22 003636	6033	6036#		
TST23 003674	6041	6044#		
TST24 003746	6056#			
TST25 004040	6057#			
TST26 004134	6060#			
TST27 004316	6086	6090#		
TST3 002420	5944	5947#		
TST30 004362	6096	6100#		
TST31 004426	6106	6111#		
TST32 004464	6116	6120#		
TST33 004530	6126	6133#		
TST34 004624	6136	6171#		
TST35 004646	6173	6177#		
TST36 004740	6192#			
TST37 005002	6198	6201#		
TST4 002456	5952	5955#		
TST40 005044	6207	6212#		
TST41 005076	6212#			
TST42 005126	6212#			
TST43 005146	6215#			
TST44 005200	6215#			
TST45 005230	6215#			
TST46 005250	6218#			
TST47 005302	6218#			
TST5 002530	5966#			
TST50 005332	6218#			
TST51 005352	6221#			
TST52 005404	6221#			
TST53 005434	6221#			
TST6 002622	5967#			
TST7 002716	5968#			
TYPBN = 104406	6232	6574#		

.HEADE	61#	5681#	5687	
.INIT	5658#	5688#	6577	
.SETUP	1213#	5682#	5715	5836
.SWRHI	104#	5683#	5694	
.SWRLO	5694#			
.SACT1	5064#	5685#	5710	
.SAPT8	5109#	5685#	5713#	
.SAPTH	5370#	5685#	5712	
.SAPTY	5547#	5685#	6570	
.SASTA	5417#			
.SCATC	932#	5681#	5695	
.SCMTA	1047#	5681#	5713	
.SDB2D	4686#			
.SDB2O	4812#			
.SDIV	4587#			
.SEOP	2214#	5681#	6224	
.SERRO	2700#	5681#	6560	
.SERRT	2896#	5683#	6561	
.SMULT	4523#			
.SPARM	5682#			
.SPOWE	4229#	5682#	6562	
.SRAND	4307#			
.SRDDE	3891#			
.SRDOC	3797#	5684#	6569	
.SREAD	3395#	5682#	6568	
.SR2AZ	4958#			
.SSAVE	3969#	5682#		
.SSB2D	4771#			
.SSB2O	4874#			
.SSCOP	2454#	5682#	6559	
.SSIZE	4361#			
.SSPAC	5682#			
.SSUPR	4913#			
.SSWDO	5682#			
.STRAP	4073#	5682#	6574	
.STYPB	3287#	5683#	6557	
.STYPD	3209#	5683#	6558	
.STYPE	2985#	5681#	5682#	6567
.STYPO	3112#	5681#	6566	
.\$40CA	972#			

. ABS. 014550 000

ERRORS DETECTED: 0

CNAAA,CNAAA/CRF/NL:TOC=CNMAC2.SML,CNAAA.P11
RUN-TIME: 16 14 1 SECONDS
RUN-TIME RATIO: 240/32=7.4
CORE USED: 33K (66 PAGES)

		IDENTIFICATION	
B	1		
C	1		
D	1		
E	1	SWITCH REGISTER	
F	1	NEOUS	
G	1	TEST DESCRIPTIONS	
H	1	-A	AAV11 DIAGNOSTIC
I	1	-A	AAV11 DIAGNOSTIC
J	1	-A	AAV11 DIAGNOSTIC
K	1	-A	AAV11 DIAGNOSTIC
L	1	-A	AAV11 DIAGNOSTIC
M	1	-A	AAV11 DIAGNOSTIC
N	1	-A	AAV11 DIAGNOSTIC
B	2	-A	AAV11 DIAGNOSTIC
C	2	-A	AAV11 DIAGNOSTIC
D	2	-A	AAV11 DIAGNOSTIC
E	2	-A	AAV11 DIAGNOSTIC
F	2	-A	AAV11 DIAGNOSTIC
G	2	-A	AAV11 DIAGNOSTIC
H	2	-A	AAV11 DIAGNOSTIC
I	2	-A	AAV11 DIAGNOSTIC
J	2	-A	AAV11 DIAGNOSTIC
K	2	-A	AAV11 DIAGNOSTIC
L	2	-A	AAV11 DIAGNOSTIC
M	2	-A	AAV11 DIAGNOSTIC
N	2	-A	AAV11 DIAGNOSTIC
B	3	-A	AAV11 DIAGNOSTIC
C	3	-A	AAV11 DIAGNOSTIC
D	3	-A	AAV11 DIAGNOSTIC
E	3	-A	AAV11 DIAGNOSTIC
F	3	-A	AAV11 DIAGNOSTIC
G	3	-A	AAV11 DIAGNOSTIC
H	3	-A	AAV11 DIAGNOSTIC
I	3	-A	AAV11 DIAGNOSTIC
J	3	-A	AAV11 DIAGNOSTIC
K	3	-A	AAV11 DIAGNOSTIC
L	3	-A	AAV11 DIAGNOSTIC
M	3	-A	AAV11 DIAGNOSTIC
N	3	-A	AAV11 DIAGNOSTIC
B	4	-A	AAV11 DIAGNOSTIC
C	4	-A	AAV11 DIAGNOSTIC
D	4	-A	AAV11 DIAGNOSTIC
E	4	-A	AAV11 DIAGNOSTIC
F	4	-A	AAV11 DIAGNOSTIC
G	4	-A	AAV11 DIAGNOSTIC
H	4	-A	AAV11 DIAGNOSTIC
I	4	-A	AAV11 DIAGNOSTIC
J	4	-A	AAV11 DIAGNOSTIC
K	4	-A	AAV11 DIAGNOSTIC
L	4	-A	AAV11 DIAGNOSTIC
M	4	-A	AAV11 DIAGNOSTIC
N	4	-A	AAV11 DIAGNOSTIC
B	5	-A	AAV11 DIAGNOSTIC
C	5	-A	AAV11 DIAGNOSTIC
D	5	-A	AAV11 DIAGNOSTIC
E	5	-A	AAV11 DIAGNOSTIC
F	5	-A	AAV11 DIAGNOSTIC
G	5	-A	AAV11 DIAGNOSTIC
H	5	-A	AAV11 DIAGNOSTIC
I	5	-A	AAV11 DIAGNOSTIC

J	5	-A	AAV11	DIAGNOSTIC
K	5	-A	AAV11	DIAGNOSTIC
L	5	-A	AAV11	DIAGNOSTIC
M	5	-A	AAV11	DIAGNOSTIC
N	5	-A	AAV11	DIAGNOSTIC
B	6	-A	AAV11	DIAGNOSTIC
C	6	-A	AAV11	DIAGNOSTIC
D	6	-A	AAV11	DIAGNOSTIC
E	6	-A	AAV11	DIAGNOSTIC
F	6	-A	AAV11	DIAGNOSTIC
G	6	-A	AAV11	DIAGNOSTIC
H	6	-A	AAV11	DIAGNOSTIC