

AXV11-C,
ADV11-C

AXV11-C/ADV11-C DIAG
CVAXABO

AH-S895B-MC
FICHE 1 OF 1

OCT 1983
COPYRIGHT © 81-83
MADE IN USA



IDENTIFICATION

Product Code: AC-S893B-MC
Product Name: CVAXAB0 AXV11-C/ADV11-C DIAG
Date: Feb. 1983
Maintainer: RAY SHOOP

Copyright (C) 1981,1983

Digital Equipment Corporation, Maynard, Mass.

The information in this document is subject to change without notice and should not be construed as a commitment by digital equipment corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by digital or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

1.0 ABSTRACT

The ADV11-C is a double height module that contains a 12 bit analog to digital (AD) converter and a 16 channel input multiplexer (MUX). The AXV11-C is the same board with the addition of two digital to analog (DAC) converters.

This diagnostic tests the AXV11-C or ADV11-C module with or without the test fixture. The program also allows interconnection to the AAV11-C D to A and KVV11-C CLOCK modules. The program does not test all the functions of the AAV11-C or KVV11-C. It only uses these devices to supply signals to test the AXV11-C/ADV11-C.

When started, the diagnostic will ask several questions that the operator must answer. A set of tests are listed and this statement is printed out: "Type the letter or number then depress 'RETURN'". The following chart indicates which letter corresponds to which test:

W: The Analog Wraparound subtests (requires test fixture)

L: Logic Subtests of AXV11-C/ADV11-C

A: Auto test (requires test fixture)

A. Logic subtests

B. Analog wraparound subtests

1: Print values of selected analog input channel and gain

2: Print values of scanned analog input channels and gains

3: AXV11-C A to D input echoed to AXV11-C D to A output

4: AXV11-C D to A ramp

5: AXV11-C D to A calibration

6: AXV11-C D to A square waves

7: AXV11-C D to A output echoed to AXV11-C A to D input

2.0 REQUIREMENTS

2.1 Equipment

PDP11 COMPUTER WITH 8K OF MEMORY
I/O Console Terminal
AXV11-C Module (A0026) or
ADV11-C Module (A8000)
AAV11-C Module (A6006) <optional>
KWV11-C Module (M4002) <optional>
Test fixture (30-18692-00) <optional>

2.2 Storage

This program uses 8K of memory and is "chainable" using XXDP or APT. When run in "CHAIN" mode, only the LOGIC sub-tests will be executed. If the operator desires to run the wraparound sections under XXDP/APT, location "\$DEV" (approx addr 1252) should be changed.

BIT0	1	KWV11-C CLK OVF CONNECTED TO AXV11-C RTC TRIG.
BIT1	2	KWV11-C CLK OVF TO AXV11-C EXT TRIG. (JUMPER 'F2')
BIT2	4	TEST FIXTURE CONNECTED TO AXV11-C CONNECTOR.
BIT3	10	AAV11-C CONNECTED TO AXV11-C TEST FIXTURE.
BIT4	20	BEVENT CONNECTED TO EXT. TRIG. (JUMPER 'F1')
BIT5	40	MODULE IS AN 'ADV11-C' TYPE.

(BITS 1 AND 4 CANNOT BOTH BE SET)
(IF BIT 3 IS SET, BIT 2 MUST ALSO BE SET)

3.0 LOADING PROCEDURE

Procedure for loading normal binary files should be followed.

4.0 STARTING PROCEDURE

4.1 Control Switch Settings

Standard PDP-11 Format

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
SW9 =1	001000	Loop on error
SW8 =1	000400	Loop on test in SWR <7:0>

Location 200 is the starting address of the diagnostic. Location 204 is the restart address.

4.2 Test Fixture (30-18692-00)

The channels listed below are expressed in OCTAL (8). The test fixture provides connection from the KVV11-C for "RTC IN" and "EXT TRIG" in addition to a voltage to each of the A to D input channels.

ADV11-C ONLY

CH00,04,10	(+ F.S.)
CH01,05,11	(+1/2 F.S.)
CH02,06,12	(+1/4 F.S.)
CH03,07	(+1/8 F.S.)
CH13	(+ F.S.)
CH14	(0 VOLTS)
CH15	(0 VOLTS)
CH16	(0 VOLTS)
CH17	(0 VOLTS)

ADV11-C TO AAV11-C

	CH00,04,10	(+ F.S.)
	CH01,05,11	(+1/2 F.S.)
	CH02,06,12	(+1/4 F.S.)
	CH03,07	(+1/8 F.S.)
	CH13	(+ F.S.)
AAV11-C DACA -	CH14	VARIABLE
DACB -	CH15	WITH
DACC -	CH16	AAV11-C
DACD -	CH17	OUTPUT

AXV11-C ONLY

AXV11-C DACA -	CH00,04,10	(+ F.S.)
	CH01,05,11	(+1/2 F.S.)
	CH02,06,12	(+1/4 F.S.)
	CH03,07	(+1/8 F.S.)
AXV11-C DACB -	CH13	(+ F.S.)
	CH14	(0 VOLTS)
	CH15	(0 VOLTS)
	CH16	(0 VOLTS)
	CH17	(0 VOLTS)

AXV11-C TO AAV11-C

AXV11-C DACA -	CH00,04,10	(+ F.S.)
	CH01,05,11	(+1/2 F.S.)
	CH02,06,12	(+1/4 F.S.)
	CH03,07	(+1/8 F.S.)
AXV11-C DACB -	CH13	(+ F.S.)
AAV11-C DACA -	CH14	VARIABLE
DACB -	CH15	WITH
DACC -	CH16	AAV11-C
DACD -	CH17	OUTPUT

4.3 MODULE JUMPER-POST CONFIGURATION

The following is the list of jumpers or posts for the AXV11-C and ADV11-C.

JUMPER	AXV11-C	ADV11-C
A12	I	I
A11	R	R
A10	R	R
A09	R	R
A08	I	I
A07	R	R
A06	R	R
A05	R	R
A04	R	R
A03	R	R
D1	R	R
D4	I	I
D5	R	R
D6	I	I
E1	R	R
E2	R	R
E3	R	R
E4	R	R
E5	R	R
E6	I	I
F1	R	R
F2	I	I
P6	I	I
P7	I	I
V4	R	R
V5	R	R
V6	R	R
V7	R	R
V8	I	I
POSTS	AXV11-C	ADV11-C
A	A3-A5	A4-A5
B	B1-B5	B4-B5
C	C1-C2	C1-C2
D	D1-D3	D1-D3
P	P1-P2	P1-P2
P	P8-P9	P8-P9

5.0 OPERATING PROCEDURE

The program heading is typed and a series of questions will be asked. The answers will control certain sub-tests. It is IMPORTANT that the answers are correct or errors will be reported. The list of tests available will be printed out followed by a message "Type letter or number then depress 'RETURN':". Then type the letter or number of the test to be run, according to the table listed and depress 'RETURN'.

The control character, ^C, is set aside for interrupting a test and transferring control to the beginning of the diagnostic (^C). During the logic tests while a reset is being performed, ^C will not be executed until after the RESET has been completed, therefore continue typing ^C until it is successful.

Location SWREG (176) is used as a software switch register. To modify the contents of SWREG, type ^G. The program responds with the current contents of SWREG and a slash. Type the desired new contents of SWREG followed by a carriage return.

If 'W' is typed, the program will run through the analog sub-test and analog wraparound sub-tests, printing "END PASS" when it has completed an entire pass.

If 'A' is typed, the program will execute the logic tests and analog wraparound sub-tests, printing "END PASS" when it has completed an entire pass.

If 'L' is typed, the program will execute the logic tests, printing "END PASS" when it has completed an entire pass.

If "1-7" is typed, the program will execute the sub-tests and will not stop until terminated by the operator.

5.1 End of Pass Timeouts

At end of pass, the following timeout will occur:

'END PASS 1.

6.0 ERRORS

This program uses the Diagnostic "SYSMAC" package for error reporting and typeout. The error information consists of the following:

ERRPC: Location at which an error was detected.
STREG: Address of the status register.
ADBUFF: Address of the buffer
CHANL: Channel value
NOMINAL: Expected correct data
TOLERANCE: The acceptable deviation from the nominal
ACTUAL: Actual data
EXPECTED: Expected correct data

7.0 MISCELLANEOUS

7.1 Execution Time

Execution time for each of the tests is:

Analog Wraparound Test:

20 seconds if using only ADV11-C
1 minute if using only AXV11-C
4 minutes if using AXV11-C connected to AAV11-C

Logic Test: 10 Seconds for first pass

1 Minute for additional passes

Auto Test: 30 seconds if using only ADV11-C

1 Minute first pass if using only AXV11-C

2 Minutes additional passes

4 Minutes first pass AXV11-C to AAV11-C

5 Minutes additional passes

7.2 Status Register and Vector Addresses

When testing more than one ADV11-C/AXV11-C, the operator must change the BUS and VECTOR addresses of the program. The ADV11-C/AXV11-C status register address must be in \$BASE (1250), its vector address must be in \$VECT1 (1244).

8.0 RESTRICTIONS

8.1 Testing

The test fixture must be present when running the auto test and the wraparound test.

8.2 Starting Restriction

If a free-running clock, such as 60Hz from the power supply, is attached to the BEVNT bus line on both Rev level C/D and E systems, an interrupt to location 100 will occur when using the "G" and "L" commands prior to executing the first instruction. Therefore this program can not disable the BEVNT bus line by inhibiting interrupts.

User systems requiring a free-running clock attached to the BEVNT bus line can temporarily avoid this situation by setting the PSW(RS) to 200, instead of using the "G" command, load the PC (R7) with the starting address and use the proceed "P" command. Before using the "L" command, the PSW(RS) can be set to 200 to avoid receiving the BEVNT interrupt after loading the ABS loader.

8.3 Possible Program "BOMBS"

The first test of the logic subtest check to see if the ADV11 responds to the expected address. If the ADV11 does not respond, a buss error occurs.

For more information on the next subject, see JAN. 1976 LSI-11 ENGINEERING BULLETIN issued by The Digital Components Group.

Bus errors may alter the preset contents of location 4 before the trap is executed, thereby transferring program control to area in the program that was not set up to handle the trap. If this happens, the program will "BOMB" and possibly rewrite parts of itself.

9.0 PROGRAM DESCRIPTION

9.1 Logic Sub-tests

These 21 logic subtests run sequentially without further operator intervention. The purpose is to check that each of the status register bits that are read/write can be loaded and properly read back; that initialize clears: the clock start enable bit, the external start enable bit, the gain select bits, the done flag, the done interrupt enable bit, the error interrupt enable bit, the error flag, and the A/D start bit. It also checks that the A/D done flag sets at end of conversion and clears when the converted value is read. It checks the DONE and ERROR interrupt logic. Additional tests are provided to verify that 'RTC IN' and 'EXT TRIG' operate correctly. Provision for 'B EVENT' and Manual Trigger are also provided.

9.2 AXV11-C/ADV11-C Analog Wraparound Sub-tests (REQUIRES TEST FIXTURE)

These 14 analog sub-tests verify correct operation of the AXV11-C/ADV11-C A to D input multiplexer. The test fixture delivers a voltage source to each of the input channels. The actual converted value is compared to the expected value. If the actual exceeds the tolerance allowed an error is reported. If an AXV11-C module, the sub-tests will verify the operation of the D to A converters. The DAC outputs are connected to AD channel 0 and 13. The program will load each DAC and verify the D to A output values. If the AAV11-C is present, the program will verify proper operation of the analog outputs are connected to AD channels 14 - 17.

- 8 sub-tests if ADV11-C only.
- 8 sub-tests if AXV11-C only.
- 11 sub-tests if ADV11-C to AAV11-C
- 12 sub-tests if AXV11-C to AAV11-C

9.3 AXV11-C I/O Sub-section

These sub-sections allow the operator to verify correct operation of the module by viewing the converted values and output signals. They provide the necessary handlers to calibrate the A to D and D to A channels. Provision is also made to verify module interconnection and different jumper configurations than what is used in the main test section.

1. I/O SUB-SECTION - Print values of selected A/D channel
The routine enables the operator to convert a selected channel plus gain and report the value. The routine allows the operator to calibrate the A to D converter or just verify the input voltage.

2. I/O SUB-SECTION - Scanning A/D channels and gain
The routine enables the operator to view the converted value across all channels and gains.

3. I/O SUB-SECTION - AXV11-C A to D input to AXV11-C DAC output
The routine converts the voltage on a selected channel and loads the result into the AXV11-C D to A outputs.

4. I/O SUB-SECTION - AXV11-C D to A ramp output
The routine loads a ramp pattern into the D to A output registers. This allows the operator to view the output levels of the AXV11-C DACS.

5. I/O SUB-SECTION - AXV11-C D to A calibration
The routine loads the maximum negative full scale value to the dac's. The operator can then verify with test equipment, the proper output voltage. When the operator has verified the level, he depresses the "RETURN". The program will then load mid-scale code into the DAC. Again once the level has been verified, the operator depresses "RETURN". The program will load maximum full scale code into the DAC.

6. I/O SUB-SECTION - AXV11-C D to A square wave
The routine produces a "SQUARE WAVE" pattern on the DAC outputs. The operator can observe the output levels for distortion.

7. I/O SUB-SECTION - AXV11-C DAC output to A to D input
The routine loads a count pattern into the D to A registers. The output is connected to the A to D input. The resulting print out should show the tracking of output to input codes.

15	BASIC DEFINITIONS
16	OPERATIONAL SWITCH SETTINGS
22	TRAP CATCHER
(1)	STARTING ADDRESS(ES)
54	ACT11 HOOKS
56	APT PARAMETER BLOCK
57	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
95	MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
158	INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE
162	INITIALIZE THE COMMON TAGS
173	DIALOGUE TO DETERMINE WHICH TEST TO RUN
174	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
259	
260	START OF LOGIC TESTS - SECTION
261	
264	T1 ADDRESS THE 4 BUS ADDRESSES OF THE AXV11-C
270	T2 FLOAT A ONE THRU MULTIPLEXER (BITS 11-8)
278	T3 LOAD AND READ BACK ERROR I.E. BIT14
282	T4 LOAD AND READ BACK INTERRUPT ENABLE BIT6
288	T5 LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BIT5
292	T6 LOAD AND READ BACK EXTERNAL START ENABLE BIT4
297	T7 LOAD AND READ BACK GAIN SELECT 0
301	T10 LOAD AND READ BACK GAIN SELECT 1
306	T11 LOAD AND READ BACK ERROR FLAG (BIT15)
310	T12 TEST INIT CLEARS BITS 2-6,14
319	T13 TEST INIT CLEARS ERROR FLAG
325	T14 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
336	T15 TEST INIT CLEARS DONE FLAG
346	T16 TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
354	T17 GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
376	T20 TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET
389	T21 TEST ERROR FLAG SETS IF 2ND CONVERSION IS STARTED WHILE A/D DONE IS SET
401	T22 TEST CLOCK OVERFLOW STARTS A/D (IF KVV11-C IS AVAILABLE)
414	T23 TEST EXTERNAL TRIGGER STARTS A/D (IF KW11-C IS CONNECTED TO EXT START TAB)
428	T24 TEST EXTERNAL TRIGGER STARTS A/D (IF MANUAL TRIGGER IS CONNECTED TO EXT START TAB)
446	T25 TEST ERROR FLAG SETS IF 2ND CONV. STARTED BEFORE DONE FLAG SETS (KVV11-C)
465	T26 TEST 'B EVENT' STARTS A/D (IF JUMPER 'F1' IS PRESENT)
477	T27 END OF ADV11-C LOGIC TESTS
481	
482	END OF LOGIC TESTS - SECTION
493	
494	START OF ADV11-C ANALOG WRAPAROUND SECTION
495	
497	T30 SETUP TO RUN ANALOG WRAPAROUND TEST
511	T31 COMPARE CHANNEL 0 (F.S.) AGAINST 1 (1/2 FS), 2 (1/4 FS), 3 (1/8)
543	T32 COMPARE CHANNEL 0 (F.S.) AGAINST OTHER F.S. CHANNELS (4 AND 10)
568	T33 COMPARE CHANNEL 1 (1/2 F.S.) AGAINST OTHER 1/2 F.S. CHANNELS (5 AND 11)
593	T34 COMPARE CHANNEL 2 (1/4 F.S.) AGAINST OTHER 1/4 F.S. CHANNELS (6 AND 12)
617	T35 COMPARE CHANNEL 3 (1/8 F.S.) AGAINST CHANNEL 7 (1/8 F.S.)
633	T36 RELATIVE GAIN TEST USING CHANNEL 3 (1/8 F.S.)
669	T37 IF ADV11-C VERIFY CH13 IS AT + F.S.
680	
681	END OF ADV11-C ANALOG WRAPAROUND SECTION

682
683 START OF AXV11-C ANALOG WRAPAROUND SECTION
684
686 T40 AXV11-C ANALOG WRAPAROUND TEST (DAC 'A' TO A/D CHAN 0)
714 T41 AXV11-C ANALOG WRAPAROUND TEST (DAC 'B' TO A/D CHAN 13)
740
741 END OF AXV11-C ANALOG WRAPAROUND SECTION
744
745 START OF AXV11-C/ADV11-C NON-WRAPAROUND ANALOG SECTION
746
748 T42 VERIFY CH14, 15, 16 AND 17 ARE AT +-0 F.S.
785
786 START OF AAV11-C TO AXV11-C ANALOG WRAPAROUND SECTION
787
789 T43 AAV11-C ANALOG WRAPAROUND TEST (DAC 'A' TO A/D CHAN 14)
820 T44 AAV11-C ANALOG WRAPAROUND TEST (DAC 'B' TO A/D CHAN 15)
849 T45 AAV11-C ANALOG WRAPAROUND TEST (DAC 'C' TO A/D CHAN 16)
878 T46 AAV11-C ANALOG WRAPAROUND TEST (DAC 'D' TO A/D CHAN 17)
903 T47 END OF AAV11-C TO AXV11-C ANALOG WRAPAROUND
906
907 END OF ADV11-C ANALOG WRAPAROUND - SECTION
908
909 START OF EXTERNAL TEST SECTION
910
914 I/O SUB-SECTION '1' REPORT THE CONVERTED A/D VALUES
946 I/O SUB-SECTION '2' SCANNING CHANNELS AND GAIN SELECT - SECTION
1002 I/O SUB-SECTION '3' AXV11-C A/D INPUT ECHO TO AXV11-C D/A OUTPUT
1025 I/O SUB-SECTION '4' AXV11-C D/A RAMPS
1049 I/O SUB-SECTION '5' AXV11-C D/A CALIBRATION
1070 I/O SUB-SECTION '6' AXV11-C D/A SQUARE WAVE
1084 I/O SUB-SECTION '7' AXV11-C D/A OUTPUT TO A/D INPUT
1106
1107 END OF EXTERNAL TESTS SECTION
1108
1109 LOGIC TEST SECTION
1116 AUTO TEST
1123 WRAPAROUND TEST
1129 DMT TEST STARTUP
1155 ROUTINE TO INITILIZE THE BUS AND VECTOR ADDRESSES
1262 END OF PASS ROUTINE
1264 ASCII MESSAGES
1332 TTY INPUT ROUTINE
1334 READ AN OCTAL NUMBER FROM THE TTY
1336 POWER DOWN AND UP ROUTINES
1338 SCOPE HANDLER ROUTINE
1339 ERROR HANDLER ROUTINE
1340 ERROR MESSAGE TYPEOUT ROUTINE
1342 TYPE ROUTINE
1343 APT COMMUNICATIONS ROUTINE
1345 BINARY TO OCTAL (ASCII) AND TYPE
1346 BINARY TO ASCII AND TYPE ROUTINE
1347 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1349 TRAP DECODER
(3) TRAP TABLE

```

1      :DEVELOPED USING SYSMAC.C4
14     :TITLE MAINDEC-11-CVAXA-B
(1)    :*COPYRIGHT (C) 1983
(1)    :*DIGITAL EQUIPMENT CORP.
(1)    :*MAYNARD, MASS. 01754
(1)    :*
(1)    :*PROGRAM BY R.SHOOP
(1)    :*
(1)    :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)    :*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
(1)    :*
15     :SBTTL BASIC DEFINITIONS
(1)
(1)    :*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)    001100 STACK= 1100
(1)    :EQUIV EMT,ERROR      ::BASIC DEFINITION OF ERROR CALL
(1)    :EQUIV IOT,SCOPE     ::BASIC DEFINITION OF SCOPE CALL
(1)
(1)    :*MISCELLANEOUS DEFINITIONS
(1)    000011 HT= 11        ::CODE FOR HORIZONTAL TAB
(1)    000012 LF= 12        ::CODE FOR LINE FEED
(1)    000015 CR= 15        ::CODE FOR CARRIAGE RETURN
(1)    000200 CRLF= 200     ::CODE FOR CARRIAGE RETURN-LINE FEED
(1)    177776 PS= 177776   ::PROCESSOR STATUS WORD
(1)    :EQUIV PS,PSW
(1)    177774 STKLMT= 177774 ::STACK LIMIT REGISTER
(1)    177772 PIRQ= 177772  ::PROGRAM INTERRUPT REQUEST REGISTER
(1)    177570 DSWR= 177570  ::HARDWARE SWITCH REGISTER
(1)    177570 DDISP= 177570 ::HARDWARE DISPLAY REGISTER
(1)
(1)    :*GENERAL PURPOSE REGISTER DEFINITIONS
(1)    000000 R0= %0        ::GENERAL REGISTER
(1)    000001 R1= %1        ::GENERAL REGISTER
(1)    000002 R2= %2        ::GENERAL REGISTER
(1)    000003 R3= %3        ::GENERAL REGISTER
(1)    000004 R4= %4        ::GENERAL REGISTER
(1)    000005 R5= %5        ::GENERAL REGISTER
(1)    000006 R6= %6        ::GENERAL REGISTER
(1)    000007 R7= %7        ::GENERAL REGISTER
(1)    000006 SP= %6        ::STACK POINTER
(1)    000007 PC= %7        ::PROGRAM COUNTER
(1)
(1)    :*PRIORITY LEVEL DEFINITIONS
(1)    000000 PR0= 0        ::PRIORITY LEVEL 0
(1)    000040 PR1= 40       ::PRIORITY LEVEL 1
(1)    000100 PR2= 100      ::PRIORITY LEVEL 2
(1)    000140 PR3= 140      ::PRIORITY LEVEL 3
(1)    000200 PR4= 200      ::PRIORITY LEVEL 4
(1)    000240 PR5= 240      ::PRIORITY LEVEL 5
(1)    000300 PR6= 300      ::PRIORITY LEVEL 6
(1)    000340 PR7= 340      ::PRIORITY LEVEL 7
(1)
(1)    :*'SWITCH REGISTER' SWITCH DEFINITIONS
(1)    100000 SW15= 100000
(1)    040000 SW14= 40000
(1)    020000 SW13= 20000

```


(1)	010000	SW12=	10000
(1)	004000	SW11=	4000
(1)	002000	SW10=	2000
(1)	001000	SW09=	1000
(1)	000400	SW08=	400
(1)	000200	SW07=	200
(1)	000100	SW06=	100
(1)	000040	SW05=	40
(1)	000020	SW04=	20
(1)	000010	SW03=	10
(1)	000004	SW02=	4
(1)	000002	SW01=	2
(1)	000001	SW00=	1
(1)		.EQUIV	SW09,SW9
(1)		.EQUIV	SW08,SW8
(1)		.EQUIV	SW07,SW7
(1)		.EQUIV	SW06,SW6
(1)		.EQUIV	SW05,SW5
(1)		.EQUIV	SW04,SW4
(1)		.EQUIV	SW03,SW3
(1)		.EQUIV	SW02,SW2
(1)		.EQUIV	SW01,SW1
(1)		.EQUIV	SW00,SW0

(1) ;*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) ;*BASIC "CPU" TRAP VECTOR ADDRESSES

(1)	000004	ERRVEC=	4	::TIME OUT AND OTHER ERRORS
(1)	000010	RESVEC=	10	::RESERVED AND ILLEGAL INSTRUCTIONS
(1)	000014	TBITVEC=	14	::"T" BIT

```
(1)      000014      TRTVEC= 14      ;;TRACE TRAP
(1)      000014      BPTVEC= 14      ;;BREAKPOINT TRAP (BPT)
(1)      000020      IOTVEC= 20      ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)      000024      PWRVEC= 24      ;;POWER FAIL
(1)      000030      EMTVEC= 30      ;;EMULATOR TRAP (EMT) **ERROR**
(1)      000034      TRAPVEC=34      ;;"TRAP" TRAP
(1)      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
(1)      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
(1)      000240      PIRQVEC=240     ;;PROGRAM INTERRUPT REQUEST VECTOR
16      .SBTTL      OPERATIONAL SWITCH SETTINGS
(1)      ;*
(1)      ;*          SWITCH          USE
(1)      ;*          -----
(1)      ;*          15          HALT ON ERROR
(1)      ;*          14          LOOP ON TEST
(1)      ;*          13          INHIBIT ERROR TYPEOUTS
(1)      ;*          11          INHIBIT ITERATIONS
(1)      ;*          10          BELL ON ERROR
(1)      ;*          9          LOOP ON ERROR
(1)      ;*          8          LOOP ON TEST IN SWR<7:0>
17      170400      ABASE= 170400
18      000400      AVECT1= 400
19      000200      APRIOR= 200
20
21
22      .SBTTL      TRAP CATCHER
(1)      ;*=0
(1)      000000      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)      ;*=174
(1)      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
(1)      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
(1)      .SBTTL      STARTING ADDRESS(ES)
(1)      000200      000137      001522      JMP @#BEGIN0      ;;JUMP TO STARTING ADDRESS OF PROGRAM
23      000204      000137      001530      JMP @#BEGIN2      ;RESTART ADDRESS
24
25      ;*=100
26      000100      000104      000340      000002      ;104,340,2      ;'B EVENT" HANDLER
27
28      ;*=140
29      000140      170000      000300      ;170000,300      ;'KXT11" ODT BREAK HANDLER
30
31      000000      CHAN00= 00
32      000001      CHAN01= 01
33      000002      CHAN02= 02
34      000003      CHAN03= 03
35      000004      CHAN04= 04
36      000005      CHAN05= 05
37      000006      CHAN06= 06
38      000007      CHAN07= 07
39      000010      CHAN10= 10
40      000011      CHAN11= 11
41      000012      CHAN12= 12
42      000013      CHAN13= 13
```


43	000014	CHAN14= 14
44	000015	CHAN15= 15
45	000016	CHAN16= 16
46	000017	CHAN17= 17
47		
48	000000	GAIN00= 00
49	000004	GAIN01= 04
50	000010	GAIN10= 10
51	000014	GAIN11= 14

52
53
54 .SBTTL ACT11 HOOKS

```

(1)
(2)
(1)
(1)
(1) 000046 000144
(1) 000052 010374
(1) 000052 000052
(1) 000052 000000
(1) 000052 000144
(1) 000052 001000
(1)
(2)
(1)
(1)
(1) 000024 001000
(1) 000024 000024
(1) 000024 000200
(1) 000044 000044
(1) 000044 001000
(1) 000044 001000
(1)
(2)
(1)
(1)
(1)
(1) 001000
(1) 001000 000000
(1) 001002 001174
(1) 001004 000550
(1) 001006 000132
(1) 001010 000550
(1) 001012 000031
  
```

```

*****
:HOOKS REQUIRED BY ACT11
$SVPC=      ;SAVE PC
.=46
$ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
.=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 360.   ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 90.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 360.   ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
  
```

57

(1)

(2)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(2)

(2)

(2)

(3)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

.SBTTL COMMON TAGS

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

SCMTAG: .=1100

.WORD 0
\$TSTNM: .BYTE 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
\$SWR: .WORD DSWR
\$DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$TIMES: 0
\$ESCAPE: 0
\$BELL: .ASCIIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIIZ <12>

::START OF COMMON TAGS

::CONTAINS THE TEST NUMBER
::CONTAINS ERROR FLAG
::CONTAINS SUBTEST ITERATION COUNT
::CONTAINS SCOPE LOOP ADDRESS
::CONTAINS SCOPE RETURN FOR ERRORS
::CONTAINS TOTAL ERRORS DETECTED
::CONTAINS ITEM CONTROL BYTE
::CONTAINS MAX. ERRORS PER TEST
::CONTAINS PC OF LAST ERROR INSTRUCTION
::CONTAINS ADDRESS OF 'GOOD' DATA
::CONTAINS ADDRESS OF 'BAD' DATA
::CONTAINS 'GOOD' DATA
::CONTAINS 'BAD' DATA
::RESERVED--NOT TO BE USED
::AUTOMATIC MODE INDICATOR
::INTERRUPT MODE INDICATOR
::ADDRESS OF SWITCH REGISTER
::ADDRESS OF DISPLAY REGISTER
::TTY KBD STATUS
::TTY KBD BUFFER
::TTY PRINTER STATUS REG. ADDRESS
::TTY PRINTER BUFFER REG. ADDRESS
::CONTAINS NULL CHARACTER FOR FILLS
::CONTAINS # OF FILLER CHARACTERS REQUIRED
::INSERT FILL CHARS. AFTER A 'LINE FEED'
::"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
::MAX. NUMBER OF ITERATIONS
::ESCAPE ON ERROR ADDRESS
::CODE FOR BELL
::QUESTION MARK
::CARRIAGE RETURN
::LINE FEED

.SBTTL APT MAILBOX-ETABLE

\$EVEN
\$MAIL: .WORD
\$MSGTY: .WORD AMSGTY
\$FATAL: .WORD AFATAL
\$TESTN: .WORD ATESTN
\$PASS: .WORD APASS
\$DEVCT: .WORD ADEVCT
\$UNIT: .WORD AUNIT
\$MSGAD: .WORD AMSGAD
::APT MAILBOX
::MESSAGE TYPE CODE
::FATAL ERROR NUMBER
::TEST NUMBER
::PASS COUNT
::DEVICE COUNT
::I/O UNIT NUMBER
::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	::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	000400	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001246	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001250	170400	\$BASE: .WORD	ABASE	::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)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;*      EM      ::POINTS TO THE ERROR MESSAGE
(1) ;*      DH      ::POINTS TO THE DATA HEADER
(1) ;*      DT      ::POINTS TO THE DATA
(1) ;*      DF      ::POINTS TO THE DATA FORMAT
(1)
(1) $ERRTB:
(1)
59 001256
60
61
70 ;ITEM 1
71 001256 013247 EM1 ;STATUS REG. ERROR
72 001260 013367 DH1 ;ERRPC STREG EXPECTED ACTUAL
73 001262 013536 DT1 ;$ERRPC, STREG, $GDDAT, $BDDAT
74 001264 013576 DF1
75
76
77 ;ITEM 2
78 001266 013271 EM2 ;FAILED TO INTERRUPT
79 001270 013506 DH3 ;ERRPC STREG ACTUAL
80 001272 013566 DT3 ;$ERRPC, STREG, $BDDAT
81 001274 013576 DF1
82
83 ;ITEM 3
84 001276 013315 EM3 ;UNEXPECTED INTERRUPT
85 001300 013506 DH3 ;ERRPC STREG
86 001302 013566 DT3 ;$ERRPC, STREG
87 001304 013576 DF1
88
89 ;ITEM 4
90 001306 013342 EM4 ;ERROR ON A/D CHANNEL
91 001310 013427 DH2 ;ERRPC STREG CHAN NOMINAL TOL ACTUAL
92 001312 013550 DT2 ;$ERRPC,STREG,CHANL,$GDDAT,$SPREAD,$BDDAT
93 001314 013576 DF1

```



```

95      .SBTTL      MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
96      001316      170400      STREG:      ABASE      ;ADDRESS OF STATUS REGISTER
97      001320      170401      ADST1:      ABASE+1      ;UPPER BYTE OF STATUS REG.
98      001322      170402      ADBUFF:      ABASE+2      ;ADDRESS OF A/D BUFFER
99      001324      170404      DACA:      ABASE+4      ;ADDRESS OF D TO A 'A'
100     001326      170406      DACB:      ABASE+6      ;ADDRESS OF D TO A 'B'
101     001330      000400      VECTOR:      AVECT1      ;VECTOR ADDRESS
102     001332      000402      VECTR1:      AVECT1+2
103     001334      000404      VECTR2:      AVECT1+4      ;ERROR VECTOR ADDRESS
104     001336      000406      VECTR3:      AVECT1+6
105     001340      170420      KWCSR:      170420      ;CLOCK STATUS/CONTROL REGISTER
106     001342      170422      KWBPR:      170422      ;CLOCK PRESET/COUNTER REGISTER
107     001344      170440      DAC0:      170440      ;AAV11-C DAC 'A' ADDRESS
108     001346      170442      DAC1:      170442      ;
109     001350      170444      DAC2:      170444      ;
110     001352      170446      DAC3:      170446      ;
111     001354      000020      VWRAP:      20
112     001356      001000      BARF:      BIT9      ;DELAY FACTOR
113     001360      000000      TEMP:      0      ;WORK AREA
114     001362      000000      CHANL:      0      ;CHANNEL VALUE
115     001364      000000      SPREAD:      C      ;DEVIATION FROM THE NOMINAL
116     001366      000000      TC1:      0      ;NON-ZERO, AXV11-C TEST FIXTURE IS INSTALLED
117     001370      000000      TC2:      0      ;NON-ZERO, AAV11-C TO AXV11-C CABLE IN INSTALLED
118     001372      000000      ADV11C:      0      ;NON-ZERO, MODULE IS ADV11-C (NO DAC'S ON BOARD)
119     001374      000000      KWAD:      0      ;NON-ZERO, CLOCK CONNECTED TO RTC IN
120     001376      000000      KWEX:      0      ;NON-ZERO, JUMPER F2 IS INSTALLED AND CLOCK CONNECTED TO EXT TRIG
121     001400      000000      MAEX:      0      ;NON-ZERO, JUMPER F2 IS INSTALLED AND MANUAL TRIGGER IS CONNECTED
122     001402      000000      BTEX:      0      ;NON-ZERO, JUMPER F1 IS INSTALLED
123
124     001404      UNEXP:
(1)     001404      012737      001420      001162      MOV      #1$, $ESCAPE      ;;ESCAPE TO 1$ ON ERROR
125     001412      005237      001103      INC      $ERFLG
126     001416      104003      ERROR      3
127     001420      005037      001162      1$:      CLR      $ESCAPE      ;RETURN ESCAPE TO NORMAL
128     001424      000002      RTI      ;UNEXPECTED INTERRUPT
129
130      ;SUBROUTINE TO DELAY AN AMOUNT OF CPU TIME
131
132     001426      013700      001356      STALL:      MOV      BARF, R0      ;GET DELAY FACTOR
133     001432      005300      1$:      DEC      R0      ;DELAY
134     001434      001376      BNE      1$
135     001436      000207      RTS      PC      ;EXIT

```

```

137
138 001440 022776 000001 000000 RETURN: CMP #1,@0(SP) ;DOES IT RETURN TO A WAIT?
139 001446 001002 BNE 1$ ;NO
140 001450 062716 000002 ADD #2,(SP) ;BUMP RETURN ADDRESS
141 001454 000002 1$: RTI
142
143 ;SUBROUTINE TO ASK QUESTIONS OF THE OPERATOR
144 001456 012537 001470 ASKTA: MOV (R5)+,10$ ;GET THE ASCII POINTER
145 001462 104401 001171 TYPE , $CRLF ;MAKE A FRESH LINE
146 001466 104401 TYPE ;TELL THE OPERATOR A MESSAGE
147 001470 011537 10$: MSKWAD
148 001472 104412 RDLIN
149 001474 012600 MOV (SP)+,R0 ;GET ANSWER
150 001476 005075 000000 CLR @ (R5) ;IF ANSWER IS NOT A 'Y', CLEAR MESSAGE FLAG
151 001502 042710 000040 BIC #40,(R0) ;ENSURE UPPER CASE
152 001506 122710 000131 CMPB #'Y',(R0) ;TEST IF 'Y'
153 001512 001001 BNE 1$ ;BR IF NOT
154 001514 005235 INC @ (R5)+ ;SET YES FLAG
155 001516 005725 1$: TST (R5)+ ;BUMP EXIT
156 001520 000205 RTS R5 ;EXIT
  
```



```

158 .SBTTL INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
159 BEGIN0: CLR TEMP ;CLEAR RESTART FLAG
160 BR BEGST
161 BEGIN2: INC TEMP ;SET RESTART FLAG
162 BEGST:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) MOV #CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
(1) CLR (R6)+ ;:CLEAR MEMORY LOCATION
(1) CMP #SWR,R6 ;:DONE?
(1) BNE -6 ;:LOOP BACK IF NO
(1) MOV #STACK,SP ;:SETUP THE STACK POINTER
(1) ;:INITIALIZE A FEW VECTORS
(1) MOV #SSCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
(1) MOV #340,@#IOTVEC+2 ;:LEVEL 7
(1) MOV #SEERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
(1) MOV #340,@#EMTVEC+2 ;:LEVEL 7
(1) MOV #STRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
(1) MOV #340,@#TRAPVEC+2 ;:LEVEL 7
(1) MOV #SPWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
(1) MOV #340,@#PWRVEC+2 ;:LEVEL 7
(1) MOV SENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
(1) CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
(1) CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
(1) MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
(1) MOV #.,$LPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) MOV #.,$LPERR ;: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) MOV @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
(2) MOV #64$,@#ERRVEC ;:SET UP ERROR VECTOR
(2) MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
(2) MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
(2) CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
(2) BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;:AND THE HARDWARE SWR IS NOT = -1
(2) BR 65$ ;:BRANCH IF NO TIMEOUT
(2) 64$: MOV #65$,(SP) ;:SET UP FOR TRAP RETURN
(2) 65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
(2) MOV #DISPREG,DISPLAY
(2) 66$: MOV (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
(1)
(2) 67$: CLR $PASS ;:CLEAR PASS COUNT
(2) BITB #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
(2) BEQ 67$ ;:YES,USE NON-APT SWITCH
(2) MOV #SSWREG,SWR ;:NO,USE APT SWITCH REGISTER
(2) 67$: MOV #300,@#IOTVEC+2 ;:KXT11
(2) MOV #300,@#EMTVEC+2 ;:FIX
(2) MOV #300,@#TRAPVEC+2 ;:FOR LOWER
(2) MOV #300,@#PWRVEC+2 ;:PWS LEVELS
(2) MOV #5046,$TYPE ;:A WAY TO LOWER
(2) MOV #12746,$TYPE+2 ;:PS FOR
(2) MOV # $TYPE+12,$TYPE+4
(2) MOV #RTI,$TYPE+6 ;:TTY OUTPUT
163 002004 012737 000300 000022
164 002012 012737 000300 000032
165 002020 012737 000300 000036
166 002026 012737 000300 000026
167 002034 012737 005046 016232
168 002042 012737 012746 016234
169 002050 012737 016244 016236
170 002056 012737 000002 016240

```

MAINDEC-11-CVAXA-B MACY11 30G(1063) 25-FEB-83 08:19 PAGE 4-1 K 2
CVAXAB.P11 13-DEC-82 09:32 INITIALIZE THE COMMON TAGS

SEQ 0023

171 002064 004737 013646

JSR

PC,\$TKINT

;INIT THE CONSOLE VECTORS


```

173                                     .SBTTL      DIALOGUE TO DETERMINE WHICH TEST TO RUN
174                                     .SBTTL      TYPE PROGRAM NAME
(1)                                     ::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002070 005227 177777              INC      #-1          ::FIRST TIME?
(1) 002074 001053                    BNE      68$          ::BRANCH IF NO
(1) 002076 022737 010374 000042      CMP      #SENDAD,@#42  ::ACT-11?
(1) 002104 001447                    BEQ      68$          ::BRANCH IF YES
(1) 002106 104401 002154              TYPE      69$          ::TYPE ASCII STRING
(2)                                     .SBTTL      GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002112 005737 000042              TST      @#42          ::ARE WE RUNNING UNDER XXDP/ACT?
(2) 002116 001012                    BNE      70$          ::BRANCH IF YES
(2) 002120 123727 001214 000001      CMPB     $ENV,#1        ::ARE WE RUNNING UNDER APT?
(2) 002126 001406                    BEQ      70$          ::BRANCH IF YES
(2) 002130 023727 001140 000176      CMP      SWR,#SWREG    ::SOFTWARE SWITCH REG SELECTED?
(2) 002136 001005                    BNE      71$          ::BRANCH IF NO
(2) 002140 104407                    GTSWR                     ::GET SOFT-SWR SETTINGS
(2) 002142 000403                    BR      71$
(2) 002144 112737 000001 001134      MOV      #1,$AUTOB    ::SET AUTO-MODE INDICATOR
(2) 002152 70$:
(1) 002152 000424                    BR      68$          ::GET OVER THE ASCII
(1)                                     ::69$: .ASCIIZ <CRLF># CVAXAB AXV11-C/ADV11-C DIAGNOSTIC #<CRLF>
(1) 002224 68$:
175 002224 004737 007540              JSR      PC,FXONE      :INITIALIZE ADDRESSES
176 002230 005737 001360              TST      TEMP          :ARE WE RESTARTING THE PROGRAM
177 002234 001062                    BNE      40$          :BR IF YES
178 002236 005737 001134              TST      $AUTOB        :IS IT CHAINED?
179 002242 001402                    BEQ      1$
180 002244 000137 007412              JMP      BEGIN ;RUN ONLY THE LOGIC TEST AND SELECTED WRAPAROUND IF APT/XXDP CHA
181 002250 004537 001456              JSR      R5,ASKTA      :ASK OPERATOR ABOUT DIFFERENT CONFIG.
182 002254 011537                    MSKWAD      :IS KWV11-C CONNECTED TO CLOCK START
183 002256 001374                    KWAD
184 002260 000240                    NOP
185 002262 005037 001400              CLR      MAEX          :ENSURE CLEARED FLAG
186 002266 004537 001456              JSR      R5,ASKTA      :ASK IF KWV11-C CONNECTED TO EXT. START
187 002272 011621                    MSKWEX
188 002274 001376                    KWEX
189 002276 000403                    BR      2$
190 002300 000415                    BR      4$
191 002302 005037 001402              CLR      BTEX          :IF ANSWER WAS YES, BYPASS NEXT QUESTION
192 002306 004537 001456              JSR      R5,ASKTA      :ENSURE CLEARED FLAG
193 002312 011730                    MSMAEX      :ASK IF MANUAL TRIGGER IS CONNECTED TO EXT. START
194 002314 001400                    MAEX
195 002316 000401                    BR      3$
196 002320 000405                    BR      4$
197 002322 004537 001456              JSR      R5,ASKTA      :ASK IF B EVENT IS CONNECTED TO EXT TRIG
198 002326 012106                    MSBTEX
199 002330 001402                    BTEX
200 002332 000240                    NOP
201 002334 004537 001456              JSR      R5,ASKTA      :ASK IF MODULE IS ADV11-C
202 002340 012201                    MSADV
203 002342 001372                    ADV11C
204 002344 000240                    NOP
205 002346 004537 001456              JSR      R5,ASKTA      :ASK IF TEST FIXTURE #1 IS INSTALLED
206 002352 012230                    MSTC1
207 002354 001366                    TC1
208 002356 000240                    NOP

```

```
209 002360 004537 001456      11$: JSR      R5,ASKTA      ;ASK IF TEST CONNECTOR #2 IS INSTALLED
210 002364 012307
211 002366 001370
212 002370 000240
213 002372 000240      12$: NOP
214 002374 000240      20$: NOP
215 002376 104401 012377      30$: TYPE, MSG70      ;TELL THE OPERATOR THE TESTS AVAILABLE
216 002402 104401 011431      40$: TYPE, MSG71
217
218 002406 104412      ;ROUTINE TO ASK OPERATOR WHAT SUB-SECTION TO EXECUTE
219 002410 052777 000100 176526 TRYAG: RDLIN
220 002416 005046
221 002420 012746 002426      BIS      #100,ASTKS
222 002424 000002      CLR      -(SP)      ;CLEAR PSW
223 002426 012600      1$: MOV      (SP)+,R0
224 002430 011000      MOV      (R0),R0      ;GET THE 1ST CHARACTER
225 002432 042700 177600      BIC      #177600,R0      ;REMOVE EXTRA BITS
226 002436 012701 002464      MOV      #OKCHAR,R1      ;LOAD POINTER TO GOOD CHARACTER LIST
227 002442 020021      2$: CMP      R0,(R1)+      ;CHECK IF VALID CHARACTER
228 002444 001002      BNE      3$      ;BR IF NOT
229 002446 011101      MOV      (R1),R1      ;GET THE ADDRESS
230 002450 000111      JMP      @R1      ;DO THE SELECTED SUB-TEST
231 002452 005721      3$: TST      (R1)+      ;BUMP THE POINTER
232 002454 001372      BNE      2$      ;BR IF MORE CHARACTERS
233 002456 104401 011131      6$: TYPE, QUEST
234 002462 000751      BR      TRYAG      ;WAIT FOR CHARACTER
235
236      ;TABLE OF VALID MENU CHARACTERS AND STARTING ADDRESS
237 002464 000141      OKCHAR: 141      ;LOWER CASE 'A'
238 002466 007352      BEGINA
239 002470 000154      154      ;LOWER CASE 'L'
240 002472 007334      BEGINL
241 002474 000167      167      ;LOWER CASE 'W'
242 002476 007374      BEGINW
243 002500 000101      'A
244 002502 007352      BEGINA
245 002504 000114      'L
246 002506 007334      BEGINL
247 002510 000127      'W
248 002512 007374      BEGINW
249 002514 000061 006340      '1, IOTST1
250 002520 000062 006514      '2, IOTST2
251 002524 000063 006716      '3, IOTST3
252 002530 000064 007024      '4, IOTST4
253 002534 000065 007114      '5, IOTST5
254 002540 000066 007202      '6, IOTST6
255 002544 000067 007250      '7, IOTST7
256 002550 000000 000000 000000      0,0,0,0
257 002556 000000
```


263 002560

264

(3)

(3)

(2)

265

266

267

268

269

270

(3)

(3)

(2)

271

272

273

274

275

276

277

278

(3)

(3)

(2)

279

280

281

282

(3)

(3)

(2)

283

284

285

286

287

288

(3)

(3)

(2)

289

290

291

292

(3)

(3)

(2)

293

294

295

002560

002566

002574

002600

002604

002610

002614

002616

002624

002626

002630

002634

002642

002644

002646

002654

002656

002660

002662

002670

002676

002700

002702

002704

002712

002714

002716

002720

002726

002730

012737

012737

005777

005777

005777

005777

000004

012737

104415

104001

006337

023727

001370

000004

012737

104415

104001

000004

012777

012737

104415

104001

000004

012737

104415

104001

000004

012737

104415

104001

002560

000001

176516

176516

176514

176512

000400

001124

001124

010000

040000

001124

001404

000100

000040

000020

001106

001102

001124

001124

010000

001124

001124

176440

001124

001124

001124

BEG1:

*TEST 1 ADDRESS THE 4 BUS ADDRESSES OF THE AXV11-C

TST1: MOV #TST1,\$LPADR
MOV #STN-1,\$STNM ;LOAD TEST NUMBER
TST @STREG ;ADDRESS A/D STATUS REGISTER
TST @ADBUFF ;ADDRESS A/D DATA BUFFER
TST @DACA ;ADDRESS D TO A 'A'
TST @DACB ;ADDRESS D TO A 'B'

*TEST 2 FLOAT A ONE THRU MULTIPLEXER (BITS 11-8)

TST2: SCOPE
MOV #BIT8,\$GDDAT ;LOAD FIRST BIT
2\$: CHKIT
ERROR 1 ;FAILED TO LOAD + READ BIT
1\$: ASL \$GDDAT ;GET NEXT BIT
CMP \$GDDAT,#BIT12 ;FINISHED?
BNE 2\$;NO,GO TO NEXT TEST

*TEST 3 LOAD AND READ BACK ERROR I.E. BIT14

TST3: SCOPE
MOV #BIT14,\$GDDAT
CHKIT
ERROR 1 ;FAILED TO LOAD + READ ERROR I.E.

*TEST 4 LOAD AND READ BACK INTERRUPT ENABLE BIT6

TST4: SCOPE
MOV #UNEXP,@VECTOR ;SETUP FOR UNEXPECTED INTERRUPT
MOV #BIT6,\$GDDAT ;LOAD EXPECTED DATA
CHKIT
ERROR 1 ;FAILED TO LOAD + READ INTERRUPT ENABLE

*TEST 5 LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BIT5

TST5: SCOPE
MOV #BIT5,\$GDDAT ;LOAD EXPECTED DATA
CHKIT
ERROR 1 ;FAILED TO LOAD + READ CLOCK OVERFLOW START ENABLE

*TEST 6 LOAD AND READ BACK EXTERNAL START ENABLE BIT4

TST6: SCOPE
MOV #BIT4,\$GDDAT ;LOAD EXPECTED DATA
CHKIT
ERROR 1 ;FAILED TO LOAD + READ EXT. START ENABLE

```
297 (3) *****
298 (3) :*TEST 7 LOAD AND READ BACK GAIN SELECT 0
299 (2) 002732 000004 *****
300 002734 012737 000004 001124 TST7: SCOPE
301 002742 104415 MOV #BIT2,$GDDAT ;LOAD EXPECTED DATA
302 002744 104001 CHKIT
303 ERROR 1 ;FAILED TO LOAD + READ BACK GAIN SELECT 0
304 *****
305 :*TEST 10 LOAD AND READ BACK GAIN SELECT 1
306 (3) *****
307 (3) :*TEST 10 LOAD AND READ BACK GAIN SELECT 1
308 (2) 002746 000004 *****
309 002750 012737 000010 001124 TST10: SCOPE
310 002756 104415 MOV #BIT3,$GDDAT ;LOAD EXPECTED
311 002760 104001 CHKIT
312 ERROR 1 ;FAILED TO LOAD + READ BACK GAIN SELECT 1
313 *****
314 :*TEST 11 LOAD AND READ BACK ERROR FLAG (BIT15)
315 (3) *****
316 (3) :*TEST 11 LOAD AND READ BACK ERROR FLAG (BIT15)
317 (2) 002762 000004 *****
318 002764 012737 100000 001124 TST11: SCOPE
319 002772 104415 MOV #BIT15,$GDDAT ;LOAD EXPECTED DATA
320 002774 104001 CHKIT
321 ERROR 1 ;FAILED TO LOAD + READ BACK ERROR FLAG
322 *****
323 :*TEST 12 TEST INIT CLEARS BITS 2-6,14
324 (3) *****
325 (3) :*TEST 12 TEST INIT CLEARS BITS 2-6,14
326 (2) 002776 000004 *****
327 (1) 003000 012737 000300 001160 TST12: SCOPE
328 003006 005037 001124 MOV #300,$TIMES ;DO 300 ITERATIONS
329 003012 012777 040174 176276 CLR $GDDAT ;LOAD EXPECTED DATA
330 003020 000005 MOV #40174,@STREG ;SET STATUS REGISTER
331 003022 052777 000100 176114 RESET ;INITIALIZE
332 003030 017737 176262 001126 BIS #100,@$TKS ;SET INTRPT. ENABLE
333 003036 001401 MOV @STREG,$BDDAT ;READ STATUS REGISTER
334 003040 104001 BEQ TST13 ;NEXT TEST
335 ERROR 1 ;RESET FAILED TO CLEAR AD ST. REG. BITS
336 *****
337 :*TEST 13 TEST INIT CLEARS ERROR FLAG
338 (3) *****
339 (3) :*TEST 13 TEST INIT CLEARS ERROR FLAG
340 (2) 003042 000004 *****
341 (1) 003044 012737 000300 001160 TST13: SCOPE
342 003052 012777 100000 176236 MOV #300,$TIMES ;DO 300 ITERATIONS
343 003060 000005 MOV #BIT15,@STREG ;SET BIT 15
344 003062 052777 000100 176054 RESET ;ISSUE INIT
345 003070 104414 BIS #100,@$TKS ;SET INTRPT. EN. FOR KEYBOARD
346 003072 104001 CHECK
347 ERROR 1 ;BUS INIT FAILED TO CLEAR A/D DONE FLAG
348 *****
349 :*TEST 14 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
350 (3) *****
351 (3) :*TEST 14 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
352 (2) 003074 000004 *****
353 003076 017700 176220 TST14: SCOPE
354 003102 005277 176210 MOV @ADBUFF,R0 ;READ DATA
355 003106 012737 000200 001124 INC @STREG ;START CONVERSION
356 003114 004737 001426 MOV #BIT7,$GDDAT ;LOAD EXPECTED
357 003120 042777 100000 176170 JSR PC,STALL ;DELAY AN AMOUNT OF TIME
358 003126 104414 BIC #BIT15,@STREG ;MASK OUT ERROR BIT
359 003130 104001 CHECK
360 ERROR 1 ;A/D DONE FLAG FAILED TO SET
```



```

333
334 003132 017700 176164      MOV      @ADBUFF,RO      ; OR BITO FAILED TO CLEAR
335                                     ;CLEAR DONE FLAG FOR ITERATIONS
336      ;*****
337      ;*TEST 15      TEST INIT CLEARS DONE FLAG
338      ;*****
339      (3)
340      (2) 003136 000004      TST15: SCOPE
341      (1) 003140 012737 000300 001160      MOV      #300,$TIMES      ;;DO 300 ITERATIONS
342      003146 005037 001124      CLR      $GDDAT      ;CLEAR EXPECTED
343      003152 005277 176140      INC      @STREG      ;START CONVERSION
344      003156 105777 176134      2$:      TSTB     @STREG
345      003162 100375      BPL      2$
346      003164 000005      RESET
347      003166 104414      CHECK
348      003170 104001      ERROR      1      ;DONE FLAG FAILED TO CLEAR
349      003172 052777 000100 175744      BIS      #100,@$TKS      ;SET INTRPT. EN. BIT
350      345
351      346      ;*****
352      (3)      ;*TEST 16      TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
353      (3)      ;*****
354      (2) 003200 000004      TST16: SCOPE
355      347 003202 005277 176110      INC      @STREG      ;SET A/D START CONVERSION BIT
356      348 003206 105777 176104      1$:      TSTB     @STREG      ;WAIT FOR FLAG
357      349 003212 100375      BPL      1$
358      350 003214 017700 176102      MOV      @ADBUFF,RO      ;READ CONVERTED VALUE
359      351 003220 104414      CHECK
360      352 003222 104001      ERROR      1      ;DONE FLAG FAILED TO CLEAR
  
```

```

354      (3)
355      (3)
356      (2) 003224 000004
357      (1)
358      (1)
359      (1) 003226 012700 000017
360      (1) 003232 004737 010144
361      003236 005046
362      003240 012746 003246
363      003244 000002
364      003246 012777 003322 176054 3$:
365      003254 012777 000200 176050
366      003262 012777 000101 176026
367      003270 105777 176022 2$:
368      003274 100375
369      003276 017737 176014 001126
370      003304 012737 000300 001124
371      003312 104002
372      003314 004737 010216
373      003320 000414
374      003322 022626
375      003324 012777 001404 175776 1$:
376      003332 005046
377      003334 012746 003342
378      003340 000002
379      003342 004737 010216 4$:
380      003346 005777 175750
381      (3)
382      (3)
383      (2) 003352 000004
384      (1)
385      (1)
386      (1)
387      (1) 003354 012700 000020
388      (1) 003360 004737 010144
389      003364 012777 003424 175742
390      003372 012777 140000 175716
391      003400 017737 175712 001126
392      003406 012737 140000 001124
393      003414 104002
394      003416 004737 010216
395      003422 000753
396      003424 022626
397      003426 004737 010216
398      003432 005077 175660

*****
*TEST 17 GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
*****
TST17: SCOPE
* 'ENTERING TEST 17' TYPED OUT TO TELL YOU THE NEXT
* TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
* THERE IS DANGER THAT THE 'Q BUSS' COULD GET 'HUNG' WHILE
* EXECUTING TEST '17'.
MOV #17,R0 ;GET TEST NO.
JSR PC,DUMW ;PRINT MESSAGE
CLR -(SP) ;RESET PRIORITY
MOV #3$,-(SP)
RTI
MOV #1$,@VECTOR ;INTERRUPT VECTOR ADDRESS
MOV #200,@VECTR1 ;SET UP NEW PSW
MOV #BIT6!BIT0,@STREG ;SET INTERRUPT ENABLE BIT + START CONVERSION
TSTB @STREG ;WAIT FOR DONE
BPL 2$ ;FLAG TO SET
MOV @STREG,$BDDAT ;READ STATUS REGISTER
MOV #BIT7!BIT6,$GDDAT ;GOOD DATA
ERROR 2 ;FAILED TO INTERRUPT ON DONE
JSR PC,DUMC ;TYPE COMPLETED
BR TST20 ;BRANCH TO NEXT TEST
CMP (SP)+,(SP)+ ;RESET STACK POINTER
MOV #UNEXP,@VECTOR ;SET UP FOR UNEXPECTED INTERRUPT
CLR -(SP) ;CLEAR PSW
MOV #4$,-(SP)
RTI
JSR PC,DUMC ;TYPE COMPLETED
TST @ADBUFF ;CLEAR DONE BIT
*****
*TEST 20 TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET
*****
TST20: SCOPE
* 'ENTERING TEST 20' TYPED OUT TO TELL YOU THE NEXT
* TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
* THERE IS DANGER THAT THE 'Q BUSS' COULD GET 'HUNG' WHILE
* EXECUTING TEST '20'.
MOV #20,R0 ;GET TEST NO.
JSR PC,DUMW ;PRINT MESSAGE
MOV #1$,@VECTR2 ;SETUP VECTOR ADDRESS
MOV #BIT15!BIT14,@STREG ;CAUSE AN INTERRUPT
MOV @STREG,$BDDAT ;BAD DATA
MOV #BIT15!BIT14,$GDDAT ;GOOD DATA
ERROR 2
JSR PC,DUMC ;TYPE COMPLETED
BR TST20
CMP (SP)+,(SP)+ ;POP STACK
JSR PC,DUMC
CLR @STREG

```



```

389
(3)
(3)
(2) 003436 000004
390 003440 012777 000001 175650
391 003446 105777 175644
392 003452 100375
393 003454 012737 100200 001124
394 003462 012777 000001 175626
395 003470 104414
396 003472 104001
397
398 003474 017700 175622
399 003500 005077 175612
400
401
(3)
(3)
(2) 003504 000004
402 003506 005737 001374
403 003512 001424
404 003514 012737 000240 001124
405 003522 113777 001124 175566
406 003530 012777 177776 175604
407 003536 012777 000011 175574
408 003544 004737 001426
409 003550 104414
410 003552 104001
411 003554 005777 175542
412 003560 005077 175532
413
414
(3)
(3)
(2) 003564 000004
415 003566 005737 001376
416 003572 001424
417 003574 012737 000220 001124
418 003602 113777 001124 175506
419 003610 012777 177776 175524
420 003616 012777 000011 175514
421 003624 004737 001426
422 003630 104414
423 003632 104001
424 003634 005777 175462
425 003640 005077 175452
426

```

```

*****
*TEST 21      TEST ERROR FLAG SETS IF 2ND CONVERSION IS STARTED WHILE A/D DONE IS SET
*****
TST21: SCOPE
MOV      #BIT0,@STREG      ;START CONVERSION
1$:      TSTB      @STREG      ;WAIT FOR
        BPL      1$
MOV      #BIT15!BIT7,$GDDAT ;LOAD EXPECTED VALUE
MOV      #BIT0,@STREG      ;START 2ND CONVERSION
CHECK
ERROR    1                  ;ERROR FLAG NOT SET WHEN 2ND
        ; CONVERSION WAS STARTED BEFORE READING BUFFER FROM FIRST
MOV      @ADBUFF,RO        ;CLEAR DONE FLAG
CLR      @STREG            ;CLEAR A/D CONTROL
*****
*TEST 22      TEST CLOCK OVERFLOW STARTS A/D (IF KVV11-C IS AVAILABLE)
*****
TST22: SCOPE
TST      KWAD              ;TEST IF OPERATOR SAID KVV11-C WAS CONNECTED
BEQ      TST23             ;BR IF NO CLOCK THERE
MOV      #BIT7!BIT5,$GDDAT ;LOAD EXPECTED A/D STATUS
MOVB     $GDDAT,@STREG     ;ENABLE THE A/D STATUS REGISTER
MOV      #177776,@KWBPR    ;LOAD KVV11-C CLOCK PRESET REGISTER
MOV      #11,@KWCSR        ;START CLOCK
JSR      PC,STALL          ;DELAY FOR A CLOCK TICK
CHECK
ERROR    1                  ;CHECK A/D STATUS AGAINST EXPECTED
TST      @ADBUFF           ;A/D DONE FAILED TO SET WITH CLOCK STARTS
CLR      @STREG            ;CLEAR A/D DONE
                          ;CLEAR A/D CONTROL
*****
*TEST 23      TEST EXTERNAL TRIGGER STARTS A/D (IF KW11-C IS CONNECTED TO EXT START TA
*****
TST23: SCOPE
TST      KWEX              ;TEST IF OPERATOR SAID KVV11-C WAS CONNECTED
BEQ      TST24             ;BR IF NO CLOCK THERE
MOV      #BIT7!BIT4,$GDDAT ;LOAD EXPECTED A/D STATUS
MOVB     $GDDAT,@STREG     ;ENABLE THE A/D STATUS REGISTER
MOV      #177776,@KWBPR    ;LOAD KVV11-C CLOCK PRESET REGISTER
MOV      #11,@KWCSR        ;START CLOCK
JSR      PC,STALL          ;DELAY FOR CLOCK TICKS
CHECK
ERROR    1                  ;CHECK A/D STATUS AGAINST EXPECTED
TST      @ADBUFF           ;A/D DONE FAILED TO SET WITH EXTERNAL STARTS
CLR      @STREG            ;CLEAR A/D DONE
                          ;CLEAR A/D CONTROL

```

428
(3)
(3)
(2) 003644 000004
429 003646 005737 001400
430 003652 001427
431 003654 005737 001202
432 003660 001024
433 003662 012737 000220 001124
434 003670 013777 001124 175420
435 003676 104401 012050
436 003702 104401 011330
437 003706 104412
438 003710 012600
439 003712 000240
440 003714 000240
441 003716 104414
442 003720 104001
443 003722 005777 175374
444 003726 005077 175364
445
446
(3)
(3)
(2) 003732 000004
447 003734 005737 001374
448 003740 001435
449 003742 012737 100240 001124
450 003750 012777 177776 175364
451 003756 112777 000040 175332
452 003764 017700 175332
453 003770 012777 000011 175342
454 003776 105777 175336
455 004002 100375
456 004004 152777 000001 175304
457
458 004012 017737 175300 001126
459 004020 100401
460 004022 104001
461
462 004024 017700 175272
463 004030 005077 175262

: *TEST 24 TEST EXTERNAL TRIGGER STARTS A/D (IF MANUAL TRIGGER IS CONNECTED TO EXT
: *****

TST24: SCOPE
TST MAEX ;TEST IF OPERATOR SAID MANUAL TRIGGER IS CONNECTED
BEQ TST25 ;:BR IF NO EXT. TRIGGER AVAILABLE
TST \$PASS ;TEST IF FIRST PASS OF PROGRAM
BNE TST25 ;:BR IF NOT FIRST PASS
MOV #BIT7!BIT4,\$GDDAT ;LOAD EXPECTED A/D STATUS
MOV \$GDDAT,@STREG ;ENABLE THE EXT START SIGNAL
TYPE ,MSGNEX ;TELL OPERATOR TO GENERATE EXT. TRIGGER
TYPE ,CRWR ;TELL OPERATOR ABOUT 'RETURN'
RDLIN
MOV (SP)+,RO ;REMOVE ANSWER OFF OF THE STACK
NOP
NOP
CHECK ;CHECK A/D STATUS AGAINST EXPECTED
ERROR 1 ;A/D DONE FAILED TO SET WITH EXTERNAL START
TST @ADBUFF ;CLEAR A/D DONE
CLR @STREG ;CLEAR A/D CONTROL

: *TEST 25 TEST ERROR FLAG SETS IF 2ND CONV. STARTED BEFORE DONE FLAG SETS (KWV11-C
: *****

TST25: SCOPE
TST KWAD ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
BEQ TST26 ;:BR IF NO CLOCK PRESENT
MOV #BIT15!BIT7!BITS,\$GDDAT ;LOAD EXPECTED
MOV #-2,@KWBPR ;LOAD CLOCK PRESET
MOVB #BITS,@STREG ;ENABLE CLOCK START
MOV @ADBUFF,RO ;ENSURE CLEARED A/D DONE
MOV #11,@KWCSR ;START CLOCK
1\$: TSTB @KWCSR ;WAIT FOR CLOCK READY
BPL 1\$
BISB #BIT0,@STREG ;CLOCK OVERFLOW SHOULD HAVE STARTED A/D
MOV @STREG,\$BDDAT ;TRY TO START IT AGAIN AND GET AN ERROR
BMI 2\$;READ A/D STATUS
ERROR 1 ;:BR IF ERROR BIT SET
2\$: MOV @ADBUFF,RO ;ERROR FLAG NOT SET WHEN 2ND CONVERT STARTED
CLR @STREG ;WHILE FIRST IS IN PROGRESS
;READ AND CLEAR A/D DONE
;CLEAR STATUS REG


```

465
(3)
(3)
(2) 004034 000004
466 004036 005737 001402
467 004042 001416
468 004044 012737 000220 001124
469 004052 013777 001124 175236
470 004060 004737 001426
471 004064 104414
472 004066 104001
473 004070 005077 175222
474 004074 005777 175222
475
476
477
(3)
(3)
(2) 004100 000004
478 004102 000207
479
480
481
482
483
484
485
486 004104 013777 001124 175204
487 004112 017737 175200 001126
488 004120 023737 001124 001126
489 004126 001002
490 004130 062716 000002
491 004134 000002
492
493
494
495

```

```

*****
*TEST 26 TEST 'B EVENT' STARTS A/D (IF JUMPER 'F1' IS PRESENT)
*****
TST26: SCOPE
TST BTEX ;TEST IF OPERATOR SAID 'F1' IS INSTALLED
BEQ TST27 ;BR IF NOT THERE
MOV #BIT7!BIT4,$GDDAT ;LOAD EXPECTED A/D STATUS
MOV $GDDAT,@STREG ;ENABLE THE A/D STATUS REGISTER
JSR PC,STALL ;DELAY AN AMOUNT OF TIME
CHECK 1 ;CHECK A/D STATUS AGAINST EXPECTED
ERROR 1 ;A/D DONE FAILED TO SET WITH 'B EVENT'
CLR @STREG ;CLEAR A/D CONTROL
TST @ADBUFF ;CLEAR A/D DONE

```

```

*****
*TEST 27 END OF ADV11-C LOGIC TESTS
*****
TST27: SCOPE
RTS PC ;RETURN TO TEST SECTION

```

```

.SBTTL
.SBTTL END OF LOGIC TESTS - SECTION

```

```

::SUBROUTINE FOR LOGIC TESTS::
TESTIT: MOV $GDDAT,@STREG ;LOAD EXPECTED VALUE
TEST: MOV @STREG,$BDDAT ;READ ST. REG.
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BNE RETERR ;ERROR RETURN
ADD #2,(SP) ;BUMP RETURN ADDRESS TO GET AROUND ERROR
RETERR: RTI

```

```

.SBTTL
.SBTTL START OF ADV11-C ANALOG WRAPAROUND SECTION
.SBTTL

```


543					*****	
(3)					:TEST 32 COMPARE CHANNEL 0 (F.S.) AGAINST OTHER F.S. CHANNELS (4 AND 10)	
(3)					*****	
(2)	004336	000004			TST32:	SCOPE
(1)	004340	012737	000001	001160		MOV #1,\$TIMES ;DO 1 ITERATION
544	004346	005737	001366			TST TC1 ;TEST IF TEST FIXTURE IS INSTALLED
545	004352	001431				BEQ TST33 ;BR IF NOT
546	004354	004537	007742			JSR R5,CONVRT ;GET THE AVERAGE VALUE FOR
547	004360	000000				CHAN00 ;CHANNEL 0
548	004362	013737	001360	004410		MOV TEMP,4\$;SAVE CHANNEL 00 CONVERTED VALUE
549	004370	013737	001360	004430		MOV TEMP,10\$;
550						
551	004376	004537	007742			JSR R5,CONVRT ;GET THE AVERAGE VALUE FOR
552	004402	000004				CHAN04 ;CHANNEL 4
553	004404	004537	010100			JSR R5,COMPAR ;COMPARE RESULTS
554	004410	000000			4\$:	0 ;AGAINST THIS VALUE FOR CHANNEL 0
555	004412	010270				V2 ;USING A SPREAD OF 2 COUNTS
556	004414	104004				ERROR 4 ;ERROR ON A/D CHANNEL 4 - VALUE DID NOT
557						; EQUAL VALUE OF CHANNEL 0
558						
559	004416	004537	007742			JSR R5,CONVRT ;GET THE AVERAGE VALUE FOR
560	004422	000010				CHAN10 ;CHANNEL 10
561	004424	004537	010100			JSR R5,COMPAR ;COMPARE RESULTS
562	004430	000000			10\$:	0 ;AGAINST THIS VALUE FOR CHANNEL 0
563	004432	010270				V2 ;USING A SPREAD OF 2 COUNTS
564	004434	104004				ERROR 4 ;ERROR ON A/D CHANNEL 10 - VALUE DID NOT
565						; EQUAL VALUE OF CHANNEL 0

```

567
568
(3)
(3)
(2) 004436 000004
(1) 004440 012737 000001 001160
569 004446 005737 001366
570 004452 001431
571 004454 004537 007742
572 004460 000001
573 004462 013737 001360 004510
574 004470 013737 001360 004530
575
576 004476 004537 007742
577 004502 000005
578 004504 004537 010100
579 004510 000000
580 004512 010270
581 004514 104004
582
583
584 004516 004537 007742
585 004522 000011
586 004524 004537 010100
587 004530 000000
588 004532 010270
589 004534 104004
590
591

```

```

*****
*TEST 33 COMPARE CHANNEL 1 (1/2 F.S.) AGAINST OTHER 1/2 F.S. CHANNELS (5 AND 11)
*****
TST33: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
TST TC1 ;TEST IF TEST FIXTURE IS INSTALLED
BEQ TST34 ;BR IF NOT
JSR R5,CONVRT ;GET THE AVERAGE VALUE FOR
CHAN01 ;CHANNEL 1
MOV TEMP,4$ ;SAVE CHANNEL 1 CONVERTED VALUE
MOV TEMP,10$ ;SAVE IT AGAIN

JSR R5,CONVRT ;GET THE AVERAGE VALUE FOR
CHAN05 ;CHANNEL 5
JSR R5,COMPAR ;COMPARE RESULTS
4$: 0 ;AGAINST THIS VALUE FOR CHANNEL 1
V2 ;USING A SPREAD OF 2 COUNTS
ERROR 4 ;ERROR ON A/D CHANNEL 5 - VALUE DID NOT
; EQUAL VALUE OF CHANNEL 0

JSR R5,CONVRT ;GET THE AVERAGE VALUE FOR
CHAN11 ;CHANNEL 11
JSR R5,COMPAR ;COMPARE RESULTS
10$: 0 ;AGAINST THIS VALUE FOR CHANNEL 1
V2 ;USING A SPREAD OF 2 COUNTS
ERROR 4 ;ERROR ON A/D CHANNEL 11 - VALUE DID NOT
; EQUAL VALUE OF CHANNEL 1

```



```

593      ;*****
(3)      ;*TEST 34      COMPARE CHANNEL 2 (1/4 F.S.) AGAINST OTHER 1/4 F.S. CHANNELS (6 AND 12)
(3)      ;*****
(2) 004536 000004      TST34: SCOPE
(1) 004540 012737 000001 001160      MOV      #1,$TIMES      ;;DO 1 ITERATION
594 004546 005737 001366      TST      TC1      ;TEST IF TEST FIXTURE IS INSTALLED
595 004552 001431      BEQ      TST35      ;;BR IF NOT
596 004554 004537 007742      JSR      R5,CONVRT      ;GET THE AVERAGE VALUE FOR
597 004560 000002      CHAN02      ;CHANNEL 2
598 004562 013737 001360 004610      MOV      TEMP,4$      ;SAVE CHANNEL 2 CONVERTED VALUE
599 004570 013737 001360 004630      MOV      TEMP,10$      ;SAVE IT AGAIN
600
601 004576 004537 007742      JSR      R5,CONVRT      ;GET THE AVERAGE VALUE FOR
602 004602 000006      CHAN06      ;CHANNEL 6
603 004604 004537 010100      JSR      R5,COMPAR      ;COMPARE RESULTS
604 004610 000000      4$: 0      ;AGAINST THIS VALUE FOR CHANNEL 2D
605 004612 010270      V2      ;USING A SPREAD OF 2 COUNTS
606 004614 104004      ERROR 4      ;ERROR ON A/D CHANNEL 6 - VALUE DID NOT
607      ; EQUAL VALUE OF CHANNEL 2
608
609 004616 004537 007742      JSR      R5,CONVRT      ;GET THE AVERAGE VALUE FOR
610 004622 000012      CHAN12      ;CHANNEL 12
611 004624 004537 010100      JSR      R5,COMPAR      ;COMPARE RESULTS
612 004630 000000      10$: 0      ;AGAINST THIS VALUE FOR CHANNEL 2
613 004632 010270      V2      ;USING A SPREAD OF 2 COUNTS
614 004634 104004      ERROR 4      ;ERROR ON A/D CHANNEL 12 - VALUE DID NOT
615      ; EQUAL VALUE OF CHANNEL 2
616
617      ;*****
(3)      ;*TEST 35      COMPARE CHANNEL 3 (1/8 F.S.) AGAINST CHANNEL 7 (1/8 F.S.)
(3)      ;*****
(2) 004636 000004      TST35: SCOPE
(1) 004640 012737 000001 001160      MOV      #1,$TIMES      ;;DO 1 ITERATION
618 004646 005737 001366      TST      TC1      ;TEST IF TEST FIXTURE IS INSTALLED
619 004652 001416      BEQ      TST36      ;;BR IF NOT
620 004654 004537 007742      JSR      R5,CONVRT      ;GET THE AVERAGE VALUE FOR
621 004660 000003      CHAN03      ;CHANNEL 3
622 004662 013737 001360 004702      MOV      TEMP,4$      ;SAVE CHANNEL 3 CONVERTED VALUE
623
624 004670 004537 007742      JSR      R5,CONVRT      ;GET THE AVERAGE VALUE FOR
625 004674 000007      CHAN07      ;CHANNEL 7
626 004676 004537 010100      JSR      R5,COMPAR      ;COMPARE RESULTS
627 004702 000000      4$: 0      ;AGAINST THIS VALUE FOR CHANNEL 3
628 004704 010270      V2      ;USING A SPREAD OF 2 COUNTS
629 004706 104004      ERROR 4      ;ERROR ON A/D CHANNEL 7 - VALUE DID NOT
630      ; EQUAL VALUE OF CHANNEL 3
631

```

```
633
(3)
(3)
(2) 004710 000004
(1) 004712 012737 000001 001160
634 004720 005737 001366
635 004724 001454
636 004726 012737 000000 010076
637 004734 004537 007746
638 004740 000003
639 004742 004537 010100
640 004746 004400
641 004750 001354
642 004752 104004
643
644 004754 012737 000004 010076
645 004762 004537 007746
646 004766 000003
647 004770 004537 010100
648 004774 005000
649 004776 001354
650 005000 104004
651
652 005002 012737 000010 010076
653 005010 004537 007746
654 005014 000003
655 005016 004537 010100
656 005022 006000
657 005024 001354
658 005026 104004
659
660 005030 012737 000014 010076
661 005036 004537 007746
662 005042 000003
663 005044 004537 010100
664 005050 007777
665 005052 001354
666 005054 104004
667
668
669
(3)
(3)
(2) 005056 000004
(1) 005060 012737 000001 001160
670 005066 012777 004000 174232
671 005074 005737 001372
672 005100 001410
673 005102 004537 007742
674 005106 000013
675 005110 004537 010100
676 005114 007777
677 005116 001354
678 005120 104004

*****
*TEST 36 RELATIVE GAIN TEST USING CHANNEL 3 (1/8 F.S.)
*****
TST36: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST TC1 ;;TEST IF AXV11 OR ADV11 CONNECTOR INSTALLED
BEQ TST37 ;;BR IF NO CONNECTOR
MOV #GAIN00,OTHER ;;SELECT GAIN OF 00
JSR R5,CONVRT ;;GET THE VALUE OF CHANNEL 03
CHAN03
JSR R5,COMPAR ;;TEST GAIN
4400 ;;EXPECTED VALUE
VWRAP ;;USING KNOWN SPREAD
ERROR 4 ;;GAIN SELECT OF 00 FAILED TO EQUAL EXPECTED VALUE

MOV #GAIN01,OTHER ;;SELECT GAIN OF 01
JSR R5,CONVRT ;;GET THE VALUE OF CHANNEL 03
CHAN03
JSR R5,COMPAR ;;TEST GAIN 01
5000 ;;EXPECTED VALUE
VWRAP ;;USING KNOWN SPREAD
ERROR 4 ;;GAIN SELECT OF 01 FAILED TO INCREASE
;; CONVERTED VALUE CORRECTLY
MOV #GAIN10,OTHER ;;SET GAIN SELECT = 10
JSR R5,CONVRT ;;GET VALUE OF CHANNEL 03
CHAN03
JSR R5,COMPAR ;;TEST GAIN 10 VALUE AGAINST 01
6000 ;;EXPECTED VALUE
VWRAP ;;USING KNOWN SPREAD
ERROR 4 ;;GAIN SELECT OF 10 FAILED TO INCREASE
;; CONVERTED VALUE CORRECTLY
MOV #GAIN11,OTHER ;;SET GAIN SELECT = 11
JSR R5,CONVRT ;;GET VALUE OF CHANNEL 03
CHAN03
JSR R5,COMPAR ;;TEST GAIN 11 VALUE AGAINST 10
7777 ;;EXPECTED VALUE
VWRAP ;;USING KNOWN SPREAD
ERROR 4 ;;GAIN SELECT OF 11 FAILED TO INCREASE
;; CONVERTED VALUE CORRECTLY

*****
*TEST 37 IF ADV11-C VERIFY CH13 IS AT + F.S.
*****
TST37: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #4000,ADACB ;;SET DAC 'B' TO MIDRANGE
TST ADV11C ;;TEST IF ADV11-C
BEQ TST40 ;;BR IF NOT ADV11-C
JSR R5,CONVRT ;;GET THE CONVERTED VALUE FOR CH13
CHAN13
JSR R5,COMPAR ;;TEST CH13 AGAINST EXPECTED
7777 ;;+ F.S.
VWRAP
ERROR 4 ;;CH13 WAS NOT PULLED UP TO +F.S.
```



```

680 .SB1TL
681 .SBTTL END OF ADV11-C ANALOG WRAPAROUND SECTION
682 .SBTTL
683 .SBTTL START OF AXV11-C ANALOG WRAPAROUND SECTION
684 .SBTTL
685
686 ::*****
687 :*TEST 40 AXV11-C ANALOG WRAPAROUND TEST (DAC 'A' TO A/D CHAN 0)
688 :*****
689 TST40: SCOPE
690 (3) 005122 000004
691 (3) 005124 012737 000001 001160
692
693 MOV #1,$TIMES ::DO 1 ITERATION
694 :AXV11-C DAC 'A' CONNECTED TO AXV11-C A/D CHANNEL 0
695 :AXV11-C TEST FIXTURE IS REQUIRED
696
697 TST TC1 ;TEST IF AXV11-C TEST FIXTURE IS PRESENT
698 BEQ TST41 ;:BR IF NO TEST FIXTURE
699 TST ADV11C ;TEST IF THE MODULE IS A ADV11-C
700 BNE TST41 ;:BR IF NO DAC'S PRESENT
701 MOV #0,2$ ;PRIME THE DAC OUTPUT VALUE
702 MOV 2$,@DACA ;PRIME THE DAC OUTPUT STAGE
703 MOV #0,@STREG ;INITIILIZE THE A/D STATUS REG
704 MOV @ADBUFF,R0 ;READ A/D VALUE AND CLEAR A/D DONE FLAG
705 JSR R5,CONVRT ;GET THE VALUE OF CHANNEL 0
706
707 JSR R5,COMPAR ;COMPARE AGAINST EXPECTED D/A VALUE
708 O ;EXPECTED
709 VWRAP ;SPREAD ALLOWED
710 BR 3$ ;CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
711 ADD #10,2$ ;UPDATE THE D/A OUTPUT VALUE
712 MOV 2$,@DACA ;UPDATE THE D/A OUTPUT VOLTAGE
713 CMP #10000,2$ ;TEST IF LAST STEP
714 BNE 1$
715 BR 4$ ;:BR TO NEXT TEST
716 ERROR 4$ ;CONVERTED A/D VALUE DID NOT EQUAL EXPECTED VALUE
717 MOV #7777,@DACA ;LOAD DAC 'A' TO +F.S.
  
```

```

713
714
(3)
(3)
(2) 005252 000004
(1) 005254 012737 000001 001160
715
716
717
718 005262 005737 001366
719 005266 001445
720 005270 005737 001372
721 005274 001042
722 005276 012737 000000 005336
723 005304 013777 005336 174014
724 005312 012777 000000 173776
725 005320 017700 173776
726 005324 004537 007742 1$:
727 005330 000013
728 005332 004537 010100 2$:
729 005336 000000
730 005340 001354
731 005342 000413
732 005344 062737 000010 005336
733 005352 013777 005336 173746
734 005360 022737 010000 005336
735 005366 001356
736 005370 000401
737 005372 104004 3$:
738 005374 012777 007777 173724 4$:
739
740
741

```

*TEST 41 AXV11-C ANALOG WRAPAROUND TEST (DAC 'B' TO A/D CHAN 13)

TST41: SCOPE
MOV #1,\$TIMES ::DO 1 ITERATION
:AXV11-C DAC 'B' CONNECTED TO AXV11-C A/D CHANNEL 13
:AXV11-C TEST CABLE IS REQUIRED
TST TC1 :TEST IF AXV11-C TEST FIXTURE IS PRESENT
BEQ TST42 ::BR IF NO TEST FIXTURE
TST ADV11C :TEST IF MODULE IS AN ADV11-C
BNE TST42 ::BR IF NO DAC'A PRESENT
MOV #0,2\$:PRIME THE DAC OUTPUT VALUE
MOV 2\$,@DACB :PRIME THE DAC OUTPUT STAGE
MOV #0,@STREG :INITIILIZE THE A/D STATUS REG
MOV @ADBUFF,R0 :READ A/D VALUE AND CLEAR A/D DONE FLAG
JSR R5,CONVRT :GET THE VALUE OF CHANNEL 13
CHAN13
JSR R5,COMPAR :COMPARE AGAINST EXPECTED D/A VALUE
O :EXPECTED
VWRAP :SPREAD ALLOWED
BR 3\$:CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
ADD #10,2\$:UPDATE THE D/A OUTPUT VALUE
MOV 2\$,@DACB :UPDATE THE D/A OUTPUT VOLTAGE
CMP #10000,2\$:TEST IF LAST STEP
BNE 1\$
BR 4\$::BR TO NEXT TEST
ERROR 4 :CONVERTED D/A VALUE DID NOT EQUAL EXPECTED
MOV #7777,@DACB :SET DAC 'B' TO + F.S.
.SBTTL
.SBTTL END OF AXV11-C ANALOG WRAPAROUND SECTION


```

743
744
745
746
747
748
(3)
(3)
(2) 005402 000004
(1) 005404 012737 000001 001160
749
750
751 005412 005737 001370
752 005416 001045
753 005420 012777 000000 173670
754 005426 017700 173670
755 005432 004537 007742
756 005436 000014
757 005440 004537 010100
758 005444 004000
759 005446 010270
760 005450 104004
761
762 005452 004537 007742
763 005456 000015
764 005460 004537 010100
765 005464 004000
766 005466 010270
767 005470 104004
768
769 005472 004537 007742
770 005476 000016
771 005500 004537 010100
772 005504 004000
773 005506 010270
774 005510 104004
775
776 005512 004537 007742
777 005516 000017
778 005520 004537 010100
779 005524 004000
780 005526 010270
781 005530 104004
782

      .SBTTL
      .SBTTL START OF AXV11-C/ADV11-C NON-WRAPAROUND ANALOG SECTION
      .SBTTL

      ;*****
      ;*TEST 42      VERIFY CH14, 15, 16 AND 17 ARE AT +-0 F.S.
      ;*****
      TST42: SCOPE
      MOV      #1,$TIMES      ;DO 1 ITERATION
      ;AAV11-C TEST CONNECTOR IS NOT REQUIRED (IN FACT WILL ERROR IF PRESENT)

      TST      TC2      ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
      BNE      TST43      ;BR IF TEST CONNECTOR
      MOV      #0,@STREG      ;INITIILIZE THE A/D STATUS REG
      MOV      @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG
      JSR      R5,CONVRT      ;GET THE VALUE OF CHANNEL 14
      CHAN14
      JSR      R5,COMPAR      ;COMPARE AGAINST EXPECTED VALUE
      4000      ;EXPECTED
      V2      ;SPREAD ALLOWED
      ERROR 4      ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

      JSR      R5,CONVRT      ;GET THE VALUE OF CHANNEL 15
      CHAN15
      JSR      R5,COMPAR      ;COMPARE AGAINST EXPECTED VALUE
      4000
      V2      ;SPREAD ALLOWED
      ERROR 4      ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

      JSR      R5,CONVRT      ;GET THE VALUE OF CHANNEL 16
      CHAN16
      JSR      R5,COMPAR      ;COMPARE AGAINST EXPECTED VALUE
      4000
      V2      ;SPREAD ALLOWED
      ERROR 4      ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

      JSR      R5,CONVRT      ;GET THE VALUE OF CHANNEL 17
      CHAN17
      JSR      R5,COMPAR      ;COMPARE AGAINST EXPECTED VALUE
      4000
      V2      ;SPREAD ALLOWED
      ERROR 4      ;CONVERTED VLAUE DID NOT EQUAL EXPECTED VALUE
  
```

```
784
785
786      .SBTTL
787      .SBTTL START OF AAV11-C TO AXV11-C ANALOG WRAPAROUND SECTION
788      .SBTTL
789      ::*****
790      : *TEST 43      AAV11-C ANALOG WRAPAROUND TEST (DAC 'A' TO A/D CHAN 14)
791      : *****
792      (3)
793      (3)
794      (2) 005532 000004
795      (1) 005534 012737 000001 001160
796      TST43: SCOPE
797      MOV      #1,$TIMES      ;;DO 1 ITERATION
798      ;AAV11-C TEST CONNECTOR IS REQUIRED
799
800      TST      TC2      ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
801      BEQ      TST44      ;;BR IF NO TEST CONNECTOR
802      MOV      #0,2$      ;PRIME THE DAC OUTPUT VALUE
803      MOV      #7777,$DAC0      ;PRIME THE DAC OUTPUT STAGE
804      MOV      #0,$STREG      ;INITIILIZE THE A/D STATUS REG
805      MOV      @ADBUFF,$R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG
806      NOP
807      NOP
808
809      1$: JSR      R5,$CONVRT      ;GET THE VALUE OF CHANNEL 14
810      CHAN14
811      JSR      R5,$COMPAR      ;COMPARE AGAINST EXPECTED D/A VALUE
812      2$: 0
813      VWRAP      ;SPREAD ALLOWED
814      BR      10$      ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
815      ADD      #10,2$      ;UPDATE THE D/A OUTPUT VALUE
816      MOV      2$,7$      ;COPY VALUE
817      COM      7$      ;INVERT DATA
818      BIC      #170000,7$      ;REMOVE EXTRA BITS
819      MOV      7,$DAC0      ;UPDATE THE D/A OUTPUT VOLTAGE
820      CMP      #10000,2$      ;TEST IF LAST STEP
821      BNE      1$
822      BR      TST44      ;;BR TO NEXT TEST
823
824      7$: 0
825      10$: ERROR 4      ;CONVERTED D/A VALUE DID NOT EQUAL EXPECTED
```



```
819
820      ::*****
821      ::*TEST 44      AAV11-C ANALOG WRAPAROUND TEST (DAC 'B' TO A/D CHAN 15)
822      ::*****
823      TST44: SCOPE
824      MOV      #1,$TIMES      ;;DO 1 ITERATION
825      ;AAV11-C TEST CONNECTOR IS REQUIRED
826      TST      TC2      ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
827      BEQ      TST45      ;;BR IF NO TEST CONNECTOR
828      MOV      #0,2$      ;PRIME THE DAC OUTPUT VALUE
829      MOV      #7777,$DAC1      ;PRIME THE DAC OUTPUT STAGE
830      MOV      #0,$STREG      ;INITIILIZE THE A/D STATUS REG
831      MOV      @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG
832      1$: JSR      R5,CONVRT      ;GET THE VALUE OF CHANNEL 15
833      CHAN15
834      JSR      R5,COMPAR      ;COMPARE AGAINST EXPECTED D/A VALUE
835      2$: 0
836      VWRAP      ;SPREAD ALLOWED
837      BR      10$      ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
838      ADD      #10,2$      ;UPDATE THE D/A OUTPUT VALUE
839      MOV      2$,7$      ;COPY VALUE
840      COM      7$      ;INVERT DATA
841      BIC      #170000,7$      ;REMOVE EXTRA BITS
842      MOV      7,$DAC1      ;UPDATE THE D/A OUTPUT VOLTAGE
843      CMP      #10000,2$      ;TEST IF LAST STEP
844      BNE      1$
845      BR      TST45      ;;BR TO NEXT TEST
846      7$: 0
847      10$: ERROR      4      ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
```

```
848
849
(3)
(3)
(2) 006032 000004
(1) 006034 012737 000001 001160
850
851
852 006042 005737 001370
853 006046 001450
854 006050 012737 000000 006110
855 006056 012777 007777 173264
856 006064 012777 000000 173224
857 006072 017700 173224
858
859 006076 004537 007742
860 006102 000016
861 006104 004537 010100
862 006110 000000
863 006112 001354
864 006114 000424
865 006116 062737 000010 006110
866 006124 013737 006110 006164
867 006132 005137 006164
868 006136 042737 170000 006164
869 006144 013777 006164 173176
870 006152 022737 010000 006110
871 006160 001346
872 006162 000402
873 006164 000000
874 006166 104004
875

*****
*TEST 45 AAV11-C ANALOG WRAPAROUND TEST (DAC 'C' TO A/D CHAN 16)
*****
TST45: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
;AAV11-C TEST CONNECTOR IS REQUIRED

TST TC2 ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
BEQ TST46 ;BR IF NO TEST CONNECTOR
MOV #0,2$ ;PRIME THE DAC OUTPUT VALUE
MOV #7777,@DAC2 ;PRIME THE DAC OUTPUT STAGE
MOV #0,@STREG ;INITIILIZE THE A/D STATUS REG
MOV @ADBUFF,R0 ;READ A/D VALUE AND CLEAR A/D DONE FLAG

1$: JSR R5,CONVRT ;GET THE VALUE OF CHANNEL 16
CHAN16
JSR R5,COMPAR ;COMPARE AGAINST EXPECTED D/A VALUE

2$: 0
VWRAP ;SPREAD ALLOWED
BR 10$ ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
ADD #10,2$ ;UPDATE THE D/A OUTPUT VALUE
MOV 2$,7$ ;COPY VALUE
COM 7$ ;INVERT DATA
BIC #170000,7$ ;REMOVE EXTRA BITS
MOV 7$,@DAC2 ;UPDATE THE D/A OUTPUT VOLTAGE
CMP #10000,2$ ;TEST IF LAST STEP
BNE 1$
BR TST46 ;BR TO NEXT TEST

7$: 0
10$: ERROR 4 ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
```



```

877
878
(3)
(3)
(2) 006170 000004
(1) 006172 012737 000001 001160
879
880 006200 005737 001370
881 006204 001450
882 006206 012737 000000 006246
883 006214 012777 007777 173130
884 006222 012777 000000 173066
885 006230 017700 173066
886
887 006234 004537 007742
888 006240 000017
889 006242 004537 010100
890 006246 000000
891 006250 001354
892 006252 000424
893 006254 062737 000010 006246
894 006262 013737 006246 006322
895 006270 005137 006322
896 006274 042737 170000 006322
897 006302 013777 006322 173042
898 006310 022737 010000 006246
899 006316 001346
900 006320 000402
901 006322 000000
902 006324 104004
903
(3)
(3)
(2) 006326 000004
(1) 006330 012737 000001 001160
904 006336 000207
912

*****
*TEST 46 AAV11-C ANALOG WRAPAROUND TEST (DAC 'D' TO A/D CHAN 17)
*****
TST46: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
;AAV11-C TEST CONNECTOR IS REQUIRED
TST TC2 ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
BEQ TST47 ;;BR IF NO TEST CONNECTOR
MOV #0,2$ ;PRIME THE DAC OUTPUT VALUE
MOV #7777,@DAC3 ;PRIME THE DAC OUTPUT STAGE
MOV #0,@STREG ;INITIILIZE THE A/D STATUS REG
MOV @ADBUFF,R0 ;READ A/D VALUE AND CLEAR A/D DONE FLAG

1$: JSR R5,CONVRT ;GET THE VALUE OF CHANNEL 17
CHAN17
JSR R5,COMPAR ;COMPARE AGAINST EXPECTED D/A VALUE

2$: 0
VWRAP ;SPREAD ALLOWED
BR 10$ ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
ADD #10,2$ ;UPDATE THE D/A OUTPUT VALUE
MOV 2$,7$ ;COPY DATA
COM 7$ ;INVERT DATA
BIC #170000,7$ ;REMOVE EXTRA BITS
MOV 7$,@DAC3 ;UPDATE THE D/A OUTPUT VOLTAGE
CMP #10000,2$ ;TEST IF LAST S
BNE 1$
BR TST47 ;;BR TO NEXT TEST

7$: 0
10$: ERROR 4 ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
*****
*TEST 47 END OF AAV11-C TO AXV11-C ANALOG WRAPAROUND
*****
TST47: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
RTS PC ;EXIT AND RETURN TO CALLING ROUTINE

```

				.SBITL	I/O SUB-SECTION '1'	REPORT THE CONVERTED A/D VALUES
914						
915						
916	006340	005077	172752	IOTST1:	CLR @STREG	:CLEAR STATUS REGISTER
917	006344	104401	010430		TYPE ,MSIO1	:TYPE OUT HEADING
918	006350	005046			CLR -(SP)	:CLEAR PSW
919	006352	012746	006360		MOV #77\$,-(SP)	
920	006356	000002			RTI	
921	006360	104401	011154	77\$:	TYPE ,CCHAN	:ASK OPERATOR FOR CHANNEL
922	006364	104413			RDOCT	
923	006366	012637	006454		MOV (SP)+,10\$	:GET ANSWER
924	006372	042737	177760	006454	BIC #177760,10\$	:REMOVE EXTRA BITS
925	006400	104401	011214		TYPE ,GCHAN	:ASK OPERATOR FOR GAIN
926	006404	104413			RDOCT	
927	006406	012637	010076		MOV (SP)+,OTHER	:GET ANSWER
928	006412	006137	010076		ROL OTHER	:MOVE TO BITS
929	006416	006137	010076		ROL OTHER	:2 + 3
930	006422	042737	177763	010076	BIC #177763,OTHER	:REMOVE ANY UNWANTED BITS
931	006430	104401	011121	1\$:	TYPE ,CH	
932	006434	013746	006454		MOV 10\$,-(SP)	::SAVE 10\$ FOR TYPEOUT
(1)						::TYPE CHANNEL
(1)	006440	104403			TYPOS	::GO TYPE--OCTAL ASCII
(1)	006442	002			.BYTE 2	::TYPE 2 DIGIT(S)
(1)	006443	000			.BYTE 0	::SUPPRESS LEADING ZEROS
933	006444	012702	000010	2\$:	MOV #10,R2	:TYPEOUT COUNTER
934	006450	004537	007746	3\$:	JSR R5,CONVTR	:GET AN AVERAGED VALUE FOR THIS CHANNEL
935	006454	000000		10\$:	0	
936	006456	104401	011124	4\$:	TYPE ,SPACE	
937	006462	013746	001360		MOV TEMP,-(SP)	::SAVE TEMP FOR TYPEOUT
(1)						::PRINT OCTAL CONVERTED VALUE
(1)	006466	104403			TYPOS	::GO TYPE--OCTAL ASCII
(1)	006470	004			.BYTE 4	::TYPE 4 DIGIT(S)
(1)	006471	001			.BYTE 1	::TYPE LEADING ZEROS
938	006472	012701	010000		MOV #10000,R1	
939	006476	005301		5\$:	DEC R1	
940	006500	001376			BNE 5\$	
941	006502	005302			DEC R2	:DECREMENT THE COUNTER
942	006504	001361			BNE 3\$	:NO CARRIAGE RETURN
943	006506	104401	001171		TYPE ,\$CRLF	:CARRIAGE RETURN
944	006512	000746			BR 1\$	:REPEAT CONVERSION

				.SBITL	I/O SUB-SECTION '2'	SCANNING CHANNELS AND GAIN SELECT - SECTION
946						
947						
948	006514	104401	010506	IOTST2: TYPE	,MSIO2	:TELL OPERATOR THE SECTION NAME
949						
950	006520	005002		CLR	R2	:INITILIZE THE CHANNEL SCANNER
951	006522	005003		CLR	R3	:INITILIZE THE GAIN SELECT VALUE
952						
953	006524	104401	001171	1\$: TYPE	,\$CRLF	:MAKE A FRESH OUTPUT LINE
954	006530	012704	000007	MOV	#7,R4	:LOAD LINE WIDTH COUNTER
955						
956	006534	104401	011121	TYPE	,CH	:SHOW 'CH' TEXT
957						
958	006540	010246		MOV	R2,-(SP)	:LOAD THE CHANNEL CODE
959	006542	104403		TYPOS		
960	006544	002	001	.BYTE	2,1	
961						
962	006546	104401	011146	TYPE	,ADOT	:SEPERATE CH FROM GS
963						
964	006552	112737	000060 011150	MOVB	#'0,AZERO	:LOAD ASCII 0
965	006560	132703	000010	BITB	#10,R3	:TEST IF GS1 = 1
966	006564	001402		BEQ	2\$	:BR IF NOT SET
967	006566	105237	011150	INCB	AZERO	:MAKE IT A ONE
968	006572	104401	011150	2\$: TYPE	,AZERO	:REPORT GS1 STATUS
969						
970	006576	112737	000060 011150	MOVB	#'0,AZERO	:LOAD ASCII 0
971	006604	132703	000004	BITB	#4,R3	:TEST IF GS0 = 1
972	006610	001402		BEQ	3\$	:BR IF NOT SET
973	006612	105237	011150	INCB	AZERO	:MAKE IT A ONE
974	006616	104401	011150	3\$: TYPE	,AZERO	:REPORT GS0 STATUS
975						
976	006622	010200		MOV	R2,R0	:GET CURRENT CHANNEL VALUE
977	006624	000300		SWAB	R0	:MOVE TO MUX POSITION
978	006626	050300		BIS	R3,R0	:ADD THE GAIN SELECT BITS
979	006630	010077	172462	MOV	R0,@STREG	:SELECT MUX AND GAIN BITS
980	006634	105277	172456	4\$: INCB	@STREG	:START CONVERSION
981	006640	105777	172452	5\$: TSTB	@STREG	:WAIT FOR A/D DONE
982	006644	100375		BPL	5\$	
983						
984	006646	104401	011124	TYPE	,SPACE	:ENSURE SOME OUTPUT ROOM
985	006652	017746	172444	MOV	@ADBUFF,-(SP)	:READ CONVERTED VALUE AND SAVE FOR TYP0UT
986	006656	104403		TYPOS		
987	006660	004	001	.BYTE	4,1	
988						
989	006662	105304		DECB	R4	:FINISHED A LINE ACROSS THE PAGE
990	006664	001363		BNE	4\$	:BR AND CONVERT WITH CURRENT GAIN AND CHANNEL
991						
992	006666	005202		INC	R2	:BUMP CHANNEL VALUE
993	006670	062703	000004	ADD	#4,R3	:BUMP GAIN SELECT VALUE
994	006674	042703	177763	BIC	#177763,R3	:REMOVE EXTRA BITS
995	006700	122702	000020	CMPB	#20,R2	:TEST IS LAST CHANNEL
996	006704	001307		BNE	1\$	:BR IF NOT
997	006706	005002		CLR	R2	:INITILIZE THE CHANNEL
998	006710	104401	001171	TYPE	,\$CRLF	:INSERT ANOTHER FRESH OUTPUT LINE
999	006714	000703		BR	1\$	:AND DO IT OVER AND OVER AND OVER AGAIN
1000						

```

1002      .SB1TL  I/O SUB-SECTION '3'      AXV11-C A/D INPUT ECHO TO AXV11-C D/A OUTPUT
1003
1004 006716 104401 010546      IOTST3: TYPE      ,MS103      ;TELL OPERATOR THE NAME
1005 006722 104401 011154      TYPE      ,CCHAN      ;ASK OPER. FOR THE CHANNEL
1006 006726 104413      RDOCT
1007 006730 012637 006776      MOV      (SP)+,10$
1008 006734 042737 177760 006776      BIC      #177760,10$      ;REMOVE EXTRA BITS
1009 006742 104401 011214      TYPE      ,GCHAN      ;ASK OPER FOR THE GAIN SELECT VALUE
1010 006746 104413      RDOCT
1011 006750 012637 010076      MOV      (SP)+,OTHER      ;GET THE ANSWER
1012 006754 006337 010076      ASL      OTHER      ;MOVE INTO
1013 006760 006337 010076      ASL      OTHER      ;GAIN SELECT POSITION
1014 006764 042737 177763 010076      BIC      #177763,OTHER      ;REMOVE EXTRA BITS
1015
1016 006772 004537 007746      4$:      JSR      R5,CONVTR      ;CONVERT SELECTED CHANNEL AND GAIN
1017 006776 000000      10$:      0
1018
1019 007000 042737 170000 001360      BIC      #170000,TEMP      ;REMOVE EXTRA BITS
1020 007006 013777 001360 172310      MOV      TEMP,@DACA      ;LOAD DAC 'A'
1021 007014 013777 001360 172304      MOV      TEMP,@DACB      ;LOAD DAC 'B'
1022
1023 007022 000763      BR      4$      ;LOOP BACK AND REPEAT
1024
1025      .SBTTL  I/O SUB-SECTION '4'      AXV11-C D/A RAMPS
1026
1027 007024 104401 010611      IOTST4: TYPE      ,MS104      ;TELL OPERATOR THE NAME
1028 007030 012703 000000      MOV      #0,R3      ;LOAD DAC - F.S. VALUE
1029 007034 012704 007777      MOV      #7777,R4      ;LOAD DAC + F.S. VALUE
1030
1031 007040 012705 010000      1$:      MOV      #BIT12,R5      ;LOAD LOOP COUNT
1032 007044 010377 172254      2$:      MOV      R3,@DACA      ;LOAD DAC 'A'
1033 007050 010477 172252      MOV      R4,@DACB      ;LOAD DAC 'B'
1034 007054 005305      DEC      R5      ;FINISHED ALL BITS ?
1035 007056 001403      BEQ      3$      ;BR IF DONE
1036 007060 005304      DEC      R4      ;LOWER DAC 'B' VALUE
1037 007062 005203      INC      R3      ;RAISE DAC 'A' VALUE
1038 007064 000767      BR      2$      ;DO NEXT COUNT
1039
1040 007066 012705 010000      3$:      MOV      #BIT12,R5      ;LOAD LOOP COUNT
1041 007072 010377 172226      4$:      MOV      R3,@DACA      ;LOAD DAC 'A'
1042 007076 010477 172224      MOV      R4,@DACB      ;LOAD DAC 'B'
1043 007102 005305      DEC      R5      ;FINISHED ALL BITS ?
1044 007104 001755      BEQ      1$
1045 007106 005303      DEC      R3      ;LOWER DAC 'A' VALUE
1046 007110 005204      INC      R4      ;RAISE DAC 'B' VALUE
1047 007112 000767      BR      4$      ;DO NEXT COUNT

```



```

1049      .SBTTL  I/O SUB-SECTION '5'      AXV11-C D/A CALIBRATION
1050
1051 007114 104401 010664      IOTST5: TYPE      ,MSI05      ;TELL OPERATOR THE NAME
1052 007120 012703 000000      MOV      #0,R3      ;LOAD DAC - F.S. VALUE
1053 007124 012704 007777      MOV      #7777,R4      ;LOAD DAC + F.S. VALUE
1054 007130 012705 004000      MOV      #4000,R5      ;LOAD 0.0 F.S. VALUE
1055
1056 007134 010377 172164      1$:      MOV      R3,@DAC A      ;LOAD DAC 'A' TO - F.S.
1057 007140 010377 172162      MOV      R3,@DAC B      ;LOAD DAC 'B' TO - F.S.
1058 007144 104412      RDLIN
1059 007146 012600      MOV      (SP)+,R0      ;REMOVE CHARACTER
1060 007150 010477 172150      MOV      R4,@DAC A      ;LOAD DAC 'A' TO + F.S.
1061 007154 010477 172146      MOV      R4,@DAC B      ;LOAD DAC 'B' TO + F.S.
1062 007160 104412      RDLIN
1063 007162 012600      MOV      (SP)+,R0      ;REMOVE CHARACTER
1064 007164 010577 172134      MOV      R5,@DAC A      ;LOAD DAC 'A' TO MID POINT
1065 007170 010577 172132      MOV      R5,@DAC B      ;LOAD DAC 'B' TO MID POINT
1066 007174 104412      RDLIN
1067 007176 012600      MOV      (SP)+,R0      ;REMOVE CHARACTER
1068 007200 000755      BR      1$
1069
1070      .SBTTL  I/O SUB-SECTION '6'      AXV11-C D/A SQUARE WAVE
1071
1072 007202 104401 010731      IOTST6: TYPE      ,MSI06      ;TELL OPERATOR THE NAME
1073 007206 012703 000000      MOV      #0,R3      ;LOAD DAC - F.S.
1074 007212 012704 007777      MOV      #7777,R4      ;LOAD DAC + F.S.
1075
1076 007216 010377 172102      1$:      MOV      R3,@DAC A      ;LOAD DAC 'A' TO MIN LEVEL
1077 007222 010377 172100      MOV      R3,@DAC B      ;LOAD DAC 'B' TO MIN LEVEL
1078 007226 004737 001426      JSR      PC,STALL      ;DELAY
1079 007232 010477 172066      MOV      R4,@DAC A      ;LOAD DAC 'A' TO MAX LEVEL
1080 007236 010477 172064      MOV      R4,@DAC B      ;LOAD DAC 'B' TO MAX LEVEL
1081 007242 004737 001426      JSR      PC,STALL      ;DELAY
1082 007246 000763      BR      1$      ;LOOP BACK AND DO AGAIN
1083
1084      .SBTTL  I/O SUB-SECTION '7'      AXV11-C D/A OUTPUT TO A/D INPUT
1085
1086 007250 104401 011022      IOTST7: TYPE      ,MSI07      ;TELL OPERATOR THE SUB-SECTION NAME
1087 007254 005003      CLR      R3      ;INITILIZE THE DAC VALUE
1088 007256 104401 001171      1$:      TYPE      ,SCRLF      ;ENSURE FRESH OUTPUT LINE
1089 007262 012705 000010      MOV      #10,R5      ;LOAD LINE WIDTH COUNTER
1090
1091 007266 105277 172024      2$:      INCB      @STREG      ;START CONVERSION
1092 007272 105777 172020      3$:      TSTB      @STREG      ;WAIT FOR A/D DONE
1093 007276 100375      BPL      3$
1094 007300 010377 172020      MOV      R3,@DAC A      ;LOAD 'DAC A' OUTPUT VALUE
1095 007304 017746 172012      MOV      @ADBUFF,-(SP)      ;READ AND STORE A/D VALUE
1096 007310 104403      TYPOS
1097 007312 004      001      .BYTE      4,1
1098 007314 005203      INC      R3      ;UPDATE TO NEXT D/A VALUE
1099 007316 042703 170000      BIC      #170000,R3      ;ENSURE ONLY 12 BITS LONG
1100 007322 005305      DEC      R5      ;IS THE WIDTH FINISHED ?
1101 007324 001754      BEQ      1$      ;BR AND START FRESH OUTPUT LINE
1102 007326 104401 011124      TYPE      ,SPACE      ;ENSURE SOME ROOM
1103 007332 000755      BR      2$      ;AND DO ANOTHER CONVERSION

```

```

1105
1106
1107
1108
1109
1110
1111 007334
1112 007334 004737 002560
1113 007340 012737 007334 010304
1114 007346 000137 010306
1115
1116
1117 007352
1118 007352 004737 002560
1119 007356 004737 004136
1120 007362 012737 007352 010304
1121 007370 000137 010306
1122
1123
1124 007374
1125 007374 004737 004136
1126 007400 012737 007374 010304
1127 007406 000137 010306
1128
1129
1130 007412 032737 000001 001252
1131 007420 001402
1132 007422 005237 001374
1133 007426 032737 000002 001252
1134 007434 001402
1135 007436 005237 001376
1136 007442 032737 000004 001252
1137 007450 001402
1138 007452 005237 001366
1139 007456 032737 000010 001252
1140 007464 001402
1141 007466 005237 001370
1142 007472 032737 000020 001252
1143 007500 001402
1144 007502 005237 001402
1145 007506 032737 000040 001252
1146 007514 001402
1147 007516 005237 001372
1148 007522 000240
1149 007524 000240
1150 007526 000240
1151 007530 000240
1152 007532 000240
1153 007534 000137 007352

      .SBTTL
      .SBTTL END OF EXTERNAL TESTS SECTION
      .SBTTL
      .SBTTL LOGIC TEST SECTION

      BEGINL:
      1$: JSR PC,BEGL ;LOGIC TESTS
          MOV #1$,AGTST ;ADDRESS FOR EOP
          JMP $EOP ;TYPE END OF PASS

      .SBTTL AUTO TEST
      BEGINA:
      1$: JSR PC,BEGL ;LOGIC TESTS
          JSR PC,WRAP
          MOV #1$,AGTST ;ADDRESS FOR EOP
          JMP $EOP ;TYPE END OF PASS

      .SBTTL WRAPAROUND TEST
      BEGINW:
      1$: JSR PC,WRAP ;WRAPAROUND TESTS
          MOV #1$,AGTST
          JMP $EOP ;INCREMENTS $PASS

      .SBTTL DMT TEST STARTUP
      BEGIND:
      BIT #BIT0,$DEVM ;TEST IF KVV11-C CONNECTED TO RTC TRIGGER
      BEQ 1$ ;BR IF NOT
      INC KWAD ;SET KW CONNECTED TO AD RTC TRIG - FLAG
      BIT #BIT1,$DEVM ;TEST IF KVV11-C CONNECTED TO EXT TRIG AND 'F2'
      BEQ 2$ ;BR IF NOT
      INC KWEX ;SET KW CONNECTED TO AD EXT TRIG - FLAG
      BIT #BIT2,$DEVM ;TEST IF TEST FIXTURE CONNECTED
      BEQ 3$ ;BR IF NOT
      INC TC1 ;SET TEST FIXTURE PRESENT FLAG
      BIT #BIT3,$DEVM ;TEST IF AAV11-C CONNECTED TO TEST FIXTURE
      BEQ 4$ ;BR IF NOT
      INC TC2 ;SET AAV11-C ANALOG WRAPAROUND FLAG
      BIT #BIT4,$DEVM ;TEST IF BEVENT AND 'F1' CONNECTED
      BEQ 5$ ;BR IF NOT
      INC BTEX ;SET BEVENT AND 'F1' FLAG
      BIT #BIT5,$DEVM ;TEST IF MODULE IS AN 'ADV11-C'
      BEQ 6$ ;BR IF NOT
      INC ADV11C ;SET 'ADV11-C' FLAG

      6$: NOP
          NOP
          NOP
          NOP
          JMP BEGINA ;RUN THE 'AUTO-MODE' TESTS
  
```



```

1155 .SBTTL ROUTINE TO INITILIZE THE BUS AND VECTOR ADDRESSES
1156 007540 012737 000006 000004 FIXONE: MOV #6, @ERRVEC ;SET UP ERRVEC
1157 007546 013737 001250 001316 MOV $BASE, STREG ;RELOAD INITIAL ADDRESSES
1158 007554 013737 001250 001320 MOV $BASE, ADST1
1159 007562 013737 001250 001322 MOV $BASE, ADBUFF
1160 007570 013737 001250 001324 MOV $BASE, DACA ;PRIME DAC 'A' ADDRESS
1161 007576 013737 001250 001326 MOV $BASE, DACB ;
1162 007604 005237 001320 INC ADST1
1163 007610 062737 000002 001322 ADD #2, ADBUFF
1164 007616 062737 000004 001324 ADD #4, DACA
1165 007624 062737 000006 001326 ADD #6, DACB
1166 007632 013737 001244 001330 MOV $VECT1, VECTOR
1167 007640 042737 170000 001330 BIC #170000, VECTOR
1168 007646 013737 001330 001332 MOV VECTOR, VECTR1
1169 007654 062737 000002 001332 ADD #2, VECTR1
1170 007662 013737 001330 001334 MOV VECTOR, VECTR2
1171 007670 062737 000004 001334 ADD #4, VECTR2
1172 007676 013737 001330 001336 MOV VECTOR, VECTR3
1173 007704 062737 000006 001336 ADD #6, VECTR3
1174 ;;LOAD .+2 AND HALT TRAP CATCH;;
1175 007712 012700 000216 MOV #216, R0 ;FILL .+2
1176 007716 012701 000214 MOV #214, R1 ;LOAD HALT
1177 007722 010021 1$: MOV R0, (R1)+
1178 007724 005021 CLR (R1)+
1179 007726 010100 MOV R1, R0
1180 007730 005720 TST (R0)+
1181 007732 020027 001002 CMP R0, #1002
1182 007736 001371 BNE 1$
1183 007740 000207 RTS PC ;TEST NEXT A/D
1184
1185

```

```

1187      ::ROUTINE TO AVERAGE 8 CONVERSIONS;;
1188      CONVTR: CLR      OTHER      ;REMOVE EXTRA BITS
1189      CONVTR: MOV      (R5)+,R0    ;GET CHANNEL VALUE
1190      MOV      R0,CHANL
1191      SWAB     R0
1192      BIS      OTHER,R0            ;ADD GAIN SELECT IF NEEDED
1193      CLR      TEMP
1194      MOV      R0,@STREG            ;LOAD CHANNEL INTO MIX BITS
1195      MOV      #10000,R0
1196      2$: DEC     R0
1197      BNE     2$
1198      MOV      #RETURN,@VECTOR ;LOAD VECTOR
1199      MOV      #10,R0           ;SET UP COUNTER
1200      1$: BISB   #101,@STREG    ;SET INTRPT. EN., START CONV.
1201      WAIT
1202      MOV      @ADBUFF,77$      ;READ CONVERTED VALUE
1203      BIC      #170000,77$     ;REMOVE HIGH BITS
1204      ADD      77$,TEMP        ;READ BUFFER
1205      DEC     R0
1206      BNE     1$              ;DO 8 TIMES
1207      ASR     TEMP            ;AVERAGE VALUE
1208      ASR     TEMP
1209      ASR     TEMP
1210      ASR     TEMP
1211      ADC     TEMP
1212      RTS     R5              ;RETURN
1213      77$: 0
1214      OTHER: 0

1215      ;COMPARE $GDDAT AND $BDDAT;;
1216      COMPAR: MOV      (R5)+,$GDDAT ;GET GOOD DATA
1217      MOV      @R5+$,SPREAD ;GET SPREAD
1218      MOV      TEMP,$BDDAT ;GET BAD(ACTUAL) DATA
1219      MOV      $GDDAT,R0
1220      SUB     $BDDAT,R0
1221      BPL     7$
1222      NEG     R0
1223      7$: CMP     R0,SPREAD ;COMPARE IT TO SPREAD
1224      BGT     10$ ;GO TO ERROR PRINTOUT
1225      TST     (R5)+ ;BUMP RETURN POINTER AROUND ERROR CALL
1226      10$: RTS     R5

```



```
1228      ;;SUBROUTINE TO TYPE INTRPT. TST MSG.;;
1229      DUMW:  TST      $PASS
1230             BNE      20$
1231             MOV      #20$, $LPERR
1232             MOV      #20$, $LPADR
1233             TYPE     ,METST
1234             MOV      R0,-(SP)
1235             (1)
1236             (1) 010174 104403      TYPOS
1237             (1) 010176      002      .BYTE 2
1238             (1) 010177      000      .BYTE 0
1239             1235 010200 104401 011370  TYPE ,ONAD
1240             1236 010204 013746 001316  MOV  STREG,-(SP)
1241             (1)
1242             (1) 010210 104403      TYPOS
1243             (1) 010212      006      .BYTE 6
1244             (1) 010213      001      .BYTE 1
1245             1237 010214 000207 20$:  RTS  PC
1246             1238
1247             1239 010216 005737 001202  DUMC:  TST      $PASS
1248             1240 010222 001010      BNE      30$
1249             1241 010224 012737 010244 001110  MOV      #30$, $LPERR
1250             1242 010232 012737 010244 001106  MOV      #30$, $LPADR
1251             1243 010240 104401 011133  TYPE     ,DONE
1252             1244 010244 000207 30$:  RTS  PC
1253             1245
1254             ;;SUBROUTINE TO RESET & SET INTRPT. EN.;
1255             1247 010246 000005      RST:  RESET
1256             1248 010250 052777 000100 170666  BIS      #100,$TKS
1257             1249 010256 005046      CLR      -(SP)
1258             1250 010260 012746 010266  MOV      #1$,-(SP)
1259             1251 010264 000002      RTI
1260             1252 010266 000207 1$:  RTS  PC
1261             1253
1262             1254
1263             1255 010270 000002      V2:  2
1264             1256 010272 000012      V12: 12
1265             1257
1266             1258 010274 052777 000100 170642  AGATST: BIS      #100,$TKS
1267             1259 010302 000137      JMP      @PC)+
1268             1260 010304 001522      AGTST:  BEGIN0
```

1262

.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 AGATST
```

\$EOP:

```
NOP  
CLR $TSTNM      ;;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)+,a(PC)+ ;;RESTORE COUNTER
```

\$ENDCT: .WORD

1

```
$EOPCT  
TYPE $ENDMG      ;;TYPE 'END PASS #'  
MOV $PASS,-(SP)  ;;SAVE $PASS FOR TYPEOUT  
TYPDS            ;;GO TYPE--DECIMAL ASCII WITH SIGN
```

```
$GET42: MOV $ENULL  
BEQ a#42,R0      ;;TYPE A NULL CHARACTER  
RESET            ;;GET MONITOR ADDRESS
```

```
$ENDAD: JSR PC,(R0) ;;BRANCH IF NO MONITOR  
NOP              ;;CLEAR THE WORLD  
NOP              ;;GO TO MONITOR  
NOP              ;;SAVE ROOM
```

```
$DOAGN: NOP        ;;FOR  
NOP        ;;ACT11
```

```
$RTNAD: JMP a(PC)+  ;;RETURN  
$ENULL: .WORD AGATST
```

```
$ENDMG: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
```

```
$.ASCIZ <15><12>/END PASS #/
```

```
(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1) 010306  
(2) 010306 000240  
(1) 010310 005037 001102  
(1) 010314 005037 001160  
(1) 010320 005237 001202  
(1) 010324 042737 100000 001202  
(1) 010332 005327  
(1) 010334 000001  
(1) 010336 003022  
(1) 010340 012737  
(1) 010342 000001  
(1) 010344 010334  
(1) 010346 104401 010413  
(2) 010352 013746 001202  
(2) 010356 104405  
(1) 010360 104401 010410  
(1) 010364 013700 000042  
(1) 010370 001405  
(1) 010372 000005  
(1) 010374 004710  
(1) 010376 000240  
(1) 010400 000240  
(1) 010402 000240  
(1) 010404  
(1) 010404 000137  
(1) 010406 010274  
(1) 010410 377 377 000  
(1) 010413 015 042412 042116  
(1) 010420 050040 051501 020123  
(1) 010426 000043
```



```

1264      .SBTTL      ASCII MESSAGES
1265 010430 020200 042522 047520 MS101: .ASCIZ <200>\ REPORTING CONVERTED A TO D CHANNEL VALUES \<200>
      010436 052122 047111 020107
      010444 047503 053116 051105
      010452 042524 020104 020101
      010460 047524 042040 041440
      010466 040510 047116 046105
      010474 053040 046101 042525
      010502 020123 000200
1266 010506 020200 041523 047101 MS102: .ASCIZ <200>\ SCANNING CHANNELS AND GAINS \<200>
      010514 044516 043516 041440
      010522 040510 047116 046105
      010530 020123 047101 020104
      010536 040507 047111 020123
      010544 000200
1267 010546 020200 027501 020104 MS103: .ASCIZ <200>\ A/D INPUT ECHOED TO D/A OUTPUTS\<200>
      010554 047111 052520 020124
      010562 041505 047510 042105
      010570 052040 020117 027504
      010576 020101 052517 050124
      010604 052125 100123 000
1268 010611 200 047440 052125 MS104: .ASCIZ <200>\ OUTPUT A RAMP ON DAC 'A' AND 'B' OUTPUT\<200>
      010616 052520 020124 020101
      010624 040522 050115 047440
      010632 020116 040504 020103
      010640 040442 020042 047101
      010646 020104 041042 020042
      010654 052517 050124 052125
      010662 000200
1269 010664 020200 040503 044514 MS105: .ASCIZ <200>\ CALIBRATE THE AXV11-C D/A OUTPUTS\<200>
      010672 051102 052101 020105
      010700 044124 020105 054101
      010706 030526 026461 020103
      010714 027504 020101 052517
      010722 050124 052125 100123
      010730 000
1270 010731 200 047440 052125 MS106: .ASCIZ <200>\ OUTPUT SQUARE WAVES ON AXV11-C DAC 'A' AND 'B' OUTPUT\<200>
      010736 052520 020124 050523
      010744 040525 042522 053440
      010752 053101 051505 047440
      010760 020116 054101 030526
      010766 026461 020103 040504
      010774 020103 040442 020042
      011002 047101 020104 041042
      011010 020042 052517 050124
      011016 052125 000200
1271 011022 020200 054101 030526 MS107: .ASCIZ <200>\ AXV11-C D/A OUTPUT ECHOED TO A/D INPUT\<200>
      011030 026461 020103 027504
      011036 020101 052517 050124
      011044 052125 042440 044103
      011052 042517 020104 047524
      011060 040440 042057 044440
      011066 050116 052125 000200
1272 011074 136 103 040 CMSG: .BYTE 136,103,40,40,0 ;CONTROL C ECHO
      011077 040 000
1273 011101 136 101 040 AMSG: .BYTE 136,101,40,40,0 ;CONTROL A ECHO

```

1274	011104	040	000			
	011106	136	107	015	GMSG: .BYTE	136,107,15,12,123,127,122,105,107,72,0 ;CONTROL G ECHO
	011111	012	123	127		
	011114	122	105	107		
	011117	072	000			
1275	011121	103	000110		CH: .ASCIZ	/CH/
1276	011124	040	040	040	SPACE: .BYTE	40,40,40,40,0
	011127	040	000			
1277	011131	077	000		QUEST: .BYTE	77,0
1278	011133	040	020040	042040	DONE: .ASCIZ	/ DONE/<15><12>
	011140	047117	006505	000012		
1279	011146	000056			ADOT: .ASCIZ	\.
1280	011150	000060			AZERO: .ASCIZ	\0\
1281	011152	000057			SLASH: .ASCIZ	#/#
1282	011154	005015	051525	047111	CCHAN: .ASCIZ	<15><12>/USING OCTAL CHANNEL (0-17) ? /
	011162	020107	041517	040524		
	011170	020114	044103	047101		
	011176	042516	020114	030050		
	011204	030455	024467	037440		
	011212	000040				
1283	011214	005015	051525	047111	GCHAN: .ASCIZ	<15><12>/USING GAIN SELECT VALUE OF (0-3) ? /
	011222	020107	040507	047111		
	011230	051440	046105	041505		
	011236	020124	040526	052514		
	011244	020105	043117	024040		
	011252	026460	024463	037440		
	011260	000040				
1284	011262	005015	047105	044504	ECHAN: .ASCIZ	<15><12>/ENDING WITH OCTAL CHANNEL (0-17) ? /
	011270	043516	053440	052111		
	011276	020110	041517	040524		
	011304	020114	044103	047101		
	011312	042516	020114	030050		
	011320	030455	024467	037440		
	011326	000040				
1285	011330	005015	042504	051120	CRWR: .ASCIZ	<15><12>/DEPRESS 'RETURN' WHEN READY/<15><12>
	011336	051505	020123	051042		
	011344	052105	051125	021116		
	011352	053440	042510	020116		
	011360	042522	042101	006531		
	011366	000012				
1286	011370	047440	020116	054101	ONAD: .ASCIZ	\ ON AXV/ADV11-C AT BUS ADDRESS \
	011376	027526	042101	030526		
	011404	026461	020103	052101		
	011412	041040	051525	040440		
	011420	042104	042522	051523		
	011426	020040	000			
1287	011431	015	052012	050131	MSG71: .ASCIZ	<15><12>/TYPE LETTER AND DEPRESS 'RETURN' /
	011436	020105	042514	052124		
	011444	051105	040440	042116		
	011452	042040	050105	042522		
	011460	051523	021040	042522		
	011466	052524	047122	020042		
	011474	000				
1288	011475	015	050012	044522	HEAD5: .ASCII	<15><12>/PRINT VALUES--/
	011502	052116	053040	046101		
	011510	042525	026523	055		

1289	011515	015	020012	047105	METST: .ASCIIZ <15><12>/ ENTERING TEST /
	011522	042524	044522	043516	
	011530	052040	051505	020124	
	011536	000			
1290	011537	015	012		MSKWAD: .BYTE 15,12
1291	011541	111	020123	053513	.ASCIIZ \IS KVV11-C CONNECTED TO 'RTC IN' (J1-PIN 21) ? \
	011546	030526	026461	020103	
	011554	047503	047116	041505	
	011562	042524	020104	047524	
	011570	021040	052122	020103	
	011576	047111	020042	045050	
	011604	026461	044520	020116	
	011612	030462	020051	020077	
	011620	000			
1292	011621	015	012		MSKWEX: .BYTE 15,12
1293	011623	111	020123	053513	.ASCIIZ \IS KVV11-C CONNECTED TO 'EXT TRIG' (J1-PIN 19 AND 'F2' INSTALLED) ? \
	011630	030526	026461	020103	
	011636	047503	047116	041505	
	011644	042524	020104	047524	
	011652	021040	054105	020124	
	011660	051124	043511	020042	
	011666	045050	026461	044520	
	011674	020116	034461	040440	
	011702	042116	021040	031106	
	011710	020042	047111	052123	
	011716	046101	042514	024504	
	011724	037440	000040		
1294	011730	015	012		MSMAEX: .BYTE 15,12
1295	011732	051511	040440	046440	.ASCIIZ \IS A MANUAL TRIGGER CONNECTED TO 'EXT TRIG' (J1-PIN 19 AND 'F2' INSTALL
	011740	047101	040525	020114	
	011746	051124	043511	042507	
	011754	020122	047503	047116	
	011762	041505	042524	020104	
	011770	047524	021040	054105	
	011776	020124	051124	043511	
	012004	020042	045050	026461	
	012012	044520	020116	034461	
	012020	040440	042116	021040	
	012026	031106	020042	047111	
	012034	052123	046101	042514	
	012042	024504	037440	000040	
1296	012050	015	012		MSGNEX: .BYTE 15,12
1297	012052	042507	042516	040522	.ASCIIZ \GENERATE ONE TRIGGER SIGNAL\
	012060	042524	047440	042516	
	012066	052040	044522	043507	
	012074	051105	051440	043511	
	012102	040516	000114		
1298	012106	015	012		MSBTEx: .BYTE 15,12
1299	012110	051511	021040	020102	.ASCIIZ \IS 'B EVENT' CONNECTED TO 'EXT TRIG' ('F1' INSTALLED) ? \
	012116	053105	047105	021124	
	012124	041440	047117	042516	
	012132	052103	042105	052040	
	012140	020117	042442	052130	
	012146	052040	044522	021107	
	012154	024040	043042	021061	
	012162	044440	051516	040524	

	012170	046114	042105	020051	
	012176	020077	000		
1300	012201	200	051511	052040	MSADV: .ASCIIZ <200>\IS THIS AN ADV11-C ? \
	012206	044510	020123	047101	
	012214	040440	053104	030461	
	012222	041455	037440	000040	
1301	012230	015	012		MSTC1: .BYTE 15,12
1302	012232	051511	052040	042510	.ASCIIZ \IS THE AXV/ADV11-C TEST FIXTURE INSTALLED ? \
	012240	040440	053130	040457	
	012246	053104	030461	041455	
	012254	052040	051505	020124	
	012262	044506	052130	051125	
	012270	020105	047111	052123	
	012276	046101	042514	020104	
	012304	020077	000		
1303	012307	015	012		MSTC2: .BYTE 15,12
1304	012311	111	020123	044124	.ASCIIZ \IS THE AAV11-C TO AXV/ADV11-C TEST CABLE INSTALLED ? \
	012316	020105	040501	030526	
	012324	026461	020103	047524	
	012332	040440	053130	040457	
	012340	053104	030461	041455	
	012346	052040	051505	020124	
	012354	040503	046102	020105	
	012362	047111	052123	046101	
	012370	042514	020104	020077	
	012376	000			
1305	012377	015	012		MSG70: .BYTE 15,12
1306	012401	015	040412	020072	.ASCII <15><12>/A: AUTOMATED RUNNING OF LOGIC AND ANALOG WRAPAROUND TESTS/
	012406	052501	047524	040515	
	012414	042524	020104	052522	
	012422	047116	047111	020107	
	012430	043117	046040	043517	
	012436	041511	040440	042116	
	012444	040440	040516	047514	
	012452	020107	051127	050101	
	012460	051101	052517	042116	
	012466	052040	051505	051524	
1307	012474	005015	035114	046040	.ASCII <15><12>/L: LOGIC TESTS ONLY/
	012502	043517	041511	052040	
	012510	051505	051524	047440	
	012516	046116	131		
1308	012521	015	053412	020072	.ASCII <15><12>/W: WRAPAROUND OF ANALOG TESTS ONLY/
	012526	051127	050101	051101	
	012534	052517	042116	047440	
	012542	020106	047101	046101	
	012550	043517	052040	051505	
	012556	051524	047440	046116	
	012564	131			
1309	012565	015	030412	020072	.ASCII <15><12>/1: PRINT VALUES OF SELECTED CHANNEL/
	012572	051120	047111	020124	
	012600	040526	052514	051505	
	012606	047440	020106	042523	
	012614	042514	052103	042105	
	012622	041440	040510	047116	
	012630	046105			
1310	012632	005015	035062	050040	.ASCII <15><12>/2: PRINT VALUES OF SCANNED CHANNEL AND GAIN/

	012640	044522	052116	053040	
	012646	046101	042525	020123	
	012654	043117	051440	040503	
	012662	047116	042105	041440	
	012670	040510	047116	046105	
	012676	040440	042116	043440	
	012704	044501	116		
1311	012707	015	031412	020072	.ASCII <15><12>/3: AXV11-C A TO D INPUT ECHOED TO D TO A OUTPUT/
	012714	054101	030526	026461	
	012722	020103	020101	047524	
	012730	042040	044440	050116	
	012736	052125	042440	044103	
	012744	042517	020104	047524	
	012752	042040	052040	020117	
	012760	020101	052517	050124	
	012766	052125			
1312	012770	005015	035064	040440	.ASCII <15><12>/4: AXV11-C D TO A RAMP/
	012776	053130	030461	041455	
	013004	042040	052040	020117	
	013012	020101	040522	050115	
1313	013020	005015	035065	040440	.ASCII <15><12>/5: AXV11-C D TO A CALIBRATION/
	013026	053130	030461	041455	
	013034	042040	052040	020117	
	013042	020101	040503	044514	
	013050	051102	052101	047511	
	013056	116			
1314	013057	015	033012	020072	.ASCII <15><12>/6: AXV11-C D TO A SQUARE WAVES/
	013064	054101	030526	026461	
	013072	020103	020104	047524	
	013100	040440	051440	052521	
	013106	051101	020105	040527	
	013114	042526	123		
1315	013117	015	033412	020072	.ASCII <15><12>/7: AXV11-C D TO A OUTPUT TO A TO D INPUT/
	013124	054101	030526	026461	
	013132	020103	020104	047524	
	013140	040440	047440	052125	
	013146	052520	020124	047524	
	013154	040440	052040	020117	
	013162	020104	047111	052520	
	013170	124			
1316	013171	015	020012	000040	.ASCIIZ <15><12>/ /
1317	013176	005015	051511	045440	HEAD2: .ASCIIZ <15><12>\IS KVV11-C CONNECTED TO AXV/ADV11-C ? \
	013204	053127	030461	041455	
	013212	041440	047117	042516	
	013220	052103	042105	052040	
	013226	020117	054101	027526	
	013234	042101	030526	026461	
	013242	020103	020077	000	
1318	013247	123	040524	052524	EM1: .ASCIIZ /STATUS REG. ERROR/
	013254	020123	042522	027107	
	013262	042440	051122	051117	
	013270	000			
1319	013271	106	044501	042514	EM2: .ASCIIZ /FAILED TO INTERRUPT/
	013276	020104	047524	044440	
	013304	052116	051105	052522	
	013312	052120	000		

```

1320 013315 125 042516 050130 EM3: .ASCIIZ /UNEXPECTED INTERRUPT/
      013322 041505 042524 020104
      013330 047111 042524 051122
      013336 050125 000124
1321 013342 051105 047522 020122 EM4: .ASCIIZ #ERROR ON A/D CHANNEL#
      013350 047117 040440 042057
      013356 041440 040510 047116
      013364 046105 000
1322 013367 105 051122 041520 DH1: .ASCIIZ /ERRPC STREG EXPECTED ACTUAL/
      013374 020040 051440 051124
      013402 043505 020040 042440
      013410 050130 041505 042524
      013416 020104 041501 052524
      013424 046101 000
1323 013427 105 051122 041520 DH2: .ASCIIZ /ERRPC STREG CHANNEL NOMINAL SPREAD ACTUAL/
      013434 020040 051440 051124
      013442 043505 020040 041440
      013450 040510 047116 046105
      013456 047040 046517 047111
      013464 046101 051440 051120
      013472 040505 020104 040440
      013500 052103 040525 000114
1324 013506 051105 050122 020103 DH3: .ASCIIZ /ERRPC STREG ACTUAL/
      013514 020040 052123 042522
      013522 020107 020040 040440
      013530 052103 040525 000114
1325 .EVEN
1326
1327 013536 001116 001316 001124 DT1: $ERRPC, STREG, $GDDAT, $BDDAT,0
      013544 001126 000000
1328 013550 001116 001316 001362 DT2: $ERRPC,STREG,CHANL,$GDDAT,SPREAD,$BDDAT,0
      013556 001124 001364 001126
      013564 000000
1329 013566 001116 001316 001126 DT3: $ERRPC,STREG,$BDDAT,0
      013574 000000
1330 013576 000000 DF1: 0

```



```

.SBTTL  TTY INPUT ROUTINE

```

```

:*****
:ENABL  LSB
$TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0          ;;INPUT POINTER
$TKQOUT: .WORD 0         ;;OUTPUT POINTER
$TKQSRT: .BLKB 32.       ;;TTY KEYBOARD QUEUE
$TKQEND=.

```

```

; *TK INITIALIZE ROUTINE
; *THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
; *SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT

```

```

:
: *CALL:
: *      JSR      PC,$TKINT
: *      RETURN
:

```

```

:★      JSR      PC,$TKINT
:★      RETURN

```

```

$TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
        MOV      $TKQSR, $TKQIN ;;MOVE THE STARTING ADDRESS OF THE
        MOV      $TKQIN, $TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV      $TKSRV, @TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
        MOV      #200, @TKVEC+2 ;;'BR' LEVEL 4
        TST      @TKB        ;;CLEAR DONE FLAG
        MOV      #100, @TKS   ;;ENABLE TTY KEYBOARD INTERRUPT
        RTS      PC          ;;RETURN TO CALLER

```

```

MOV    $TKQSR, $TKQIN    ;; MOVE THE STARTING ADDRESS OF THE
MOV    $TKQIN, $TKQOUT    ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
MOV    $TKSRV, @TKVEC    ;; INITIALIZE THE KEYBOARD VECTOR
MOV    #200, @TKVEC+2    ;; 'BR' LEVEL 4
TST    @TKB              ;; CLEAR DONE FLAG
MOV    #100, @TKS        ;; ENABLE TTY KEYBOARD INTERRUPT
RTS    PC                ;; RETURN TO CALLER

```

```

MOV    $TKSRV,@TKVEC    ;;INITIALIZE THE KEYBOARD VECTOR
MOV    #200,@TKVEC+2    ;;'BR' LEVEL 4
TST    @TKB             ;;CLEAR DONE FLAG
MOV    #100,@$TKS       ;;ENABLE TTY KEYBOARD INTERRUPT
RTS    PC               ;;RETURN TO CALLER

```

```

TST    @STKB      ;;CLEAR DONE FLAG
MOV    #100,@STKS  ;;ENABLE TTY KEYBOARD INTERRUPT
RTS     PC         ;;RETURN TO CALLER

```

RTS PC ::RETURN TO CALLER

```

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (BEGIN2)

```

```

; THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT.
; BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
; IT IN THE QUEUE.
; IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
; UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (BEGIN2)

```

```

;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (BEGIN2)

```

UPON RETURN EXIT IS MADE TO THE CONTROL-C RESTART ADDRESS (BEGINE,

```

$TKSRV: MOVB    @STKB, -(SP)      ;; PICKUP THE CHARACTER
        BIC     #^C177, (SP)      ;; STRIP THE JUNK
        CMP     (SP), #XON        ;; IS IT A RANDOM XON?
        BNE     30$              ;; BRANCH IF NO
        TST     (SP)+            ;; CLEAN RANDOM XON OFF STACK
        RTI                     ;; RETURN
30$.

```

```

CMP      (SP),#9XON    ;IS IT A RANDOM XON?       :RAN001
BNE      30$           ;;BRANCH IF NO               :RAN001
TST      (SP)+          ;;CLEAN RANDOM XON OFF STACK :RAN001
RTI                      ;;RETURN                   :RAN001

```

30\$.:

```

305.      RTI
          (SP)+
          ;;CLEAR RANDOM XON OFF STACK
          ;;RETURN
          RAN001
          RAN001

```

```

BNE      IS          ;;BRANCH IF NO
TYPE     ,SCNTLC     ;;TYPE A CONTROL-C (^C)
JSR      PC,$TKINT   ;;INIT THE KEYBOARD
TST      (SP)+        ;;CLEAN UP STACK

```

```
JSR      PC, $K1INT      :: INIT THE KEYBOARD
IST      (SP)+            :: CLEAN UP STACK
```

```

1$:      JMP      BEGIN2      ;;CONTROL C RESTART
        CMP      (SP),#7      ;;IS IT A CONTROL G?

```

```

BNE      2$          ;;BRANCH IF NO
CMP      #SWREG,SWR  ;;IS SOFT-SWR SELECTED?
BEQ      6$          ;;GO TO SWR CHANGE

```

```
25:      CMP      #32.,$TKCNT      ;;IS THE QUEUE FULL?
      BNE      3$               ;;BRANCH IF NO
      TYPE     ,SBELL           ;;RING THE TTY BELL
```

```

BNE      SS      ;;BRANCH IF NO
TYPE     ,SBELL   ;;RING THE TTY BELL

```

```
(1) 014016 005726      TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
(1) 014020 000451      BR       5$      ;;EXIT
(1) 014022 021627 000023 3$:    CMP      (SP),#23      ;;IS IT A CONTROL-S?
(1) 014026 001021      BNE      32$      ;;BRANCH IF NO
(1) 014030 005077 165110      CLR      @STKS      ;;DISABLE TTY KEYBOARD INTERRUPTS
(1) 014034 005726      TST      (SP)+      ;;CLEAN CHAR OFF STACK
(1) 014036 105777 165102 31$:  TSTB     @STKS      ;;WAIT FOR A CHAR
(1) 014042 100375      BPL      31$      ;;LOOP UNTIL ITS THERE
(1) 014044 117746 165076      MOVB     @STKB,-(SP)      ;;GET THE CHARACTER
(1) 014050 042716 177600      BIC      #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
(1) 014054 022627 000021      CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
(1) 014060 001366      BNE      31$      ;;BRANCH IF NO
(1) 014062 012777 000100 165054 MOV      #100,@STKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
(1) 014070 000002      RTI      ;;RETURN
(1) 014072 005237 013600 32$:  INC      $TKCNT      ;;COUNT THIS CHARACTER
(1) 014076 021627 000140      CMP      (SP),#140      ;;IS IT UPPER CASE?
(1) 014102 002405      BLT      4$      ;;BRANCH IF YES
(1) 014104 021627 000175      CMP      (SP),#175      ;;IS IT A SPECIAL CHAR?
(1) 014110 003002      BGT      4$      ;;BRANCH IF YES
(1) 014112 042716 000040      BIC      #40,(SP)      ;;MAKE IT UPPER CASE
(1) 014116 112677 177460 4$:  MOVB     (SP)+,@STKQIN      ;;AND PUT IT IN QUEUE
(1) 014122 005237 013602      INC      $TKQIN      ;;UPDATE THE POINTER
(1) 014126 023727 013602 013646 CMP      $TKQIN,$$TKQEND      ;;GO OFF THE END?
(1) 014134 001003      BNE      5$      ;;BRANCH IF NO
(1) 014136 012737 013606 013602 MOV      $$TKQSRT,$TKQIN      ;;RESET THE POINTER
(1) 014144 000002 5$:    RTI      ;;RETURN
```

```
(1)
(2)
(1)
(1)
(1)
(1)
(1) 014146 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
(1) 014154 001124      BNE      15$      ;;EXIT IF NOT
(1) 014156 105777 164762      TSTB     @STKS      ;;IS A CHAR WAITING?
(1) 014162 100121      BPL      15$      ;;IF NOT, EXIT
(1) 014164 117746 164756      MOVB     @STKB,-(SP)      ;;YES
(1) 014170 042716 177600      BIC      #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
(1) 014174 021627 000007      CMP      (SP),#7      ;;IS IT A CONTROL-G?
(1) 014200 001300      BNE      2$      ;;IF NOT, PUT IT IN THE TTY QUEUE
(1)
(1)
(2)
(1)
(1)
(1)
(1) 014202 123727 001134 000001 6$:  CMPB     $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?
(1) 014210 001674      BEQ      2$      ;;BRANCH IF YES
(1) 014212 005726      TST      (SP)+      ;;CLEAR CONTROL-G OFF STACK
(1) 014214 004737 013646      JSR      PC,$TKINT      ;;FLUSH THE TTY INPUT QUEUE
(1) 014220 005077 164720      CLR      @STKS      ;;DISABLE TTY KEYBOARD INTERRUPTS
(1) 014224 112737 000001 001135      MOVB     #1,$INTAG      ;;SET INTERRUPT MODE INDICATOR
(1)
(1) 014232 104401 015106      TYPE     ,SCNTLG      ;;ECHO THE CONTROL-G (^G)
(1) 014236 104401 015113      $GTSWR: TYPE     ,SMSWR      ;;TYPE CURRENT CONTENTS
(2) 014242 013746 000176      MOV      SWREG,-(SP)      ;;SAVE SWREG FOR TYPEOUT
```


(2)	014246	104402			TYPUC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)	014250	104401	015124		TYPE	,\$MNEW	::PROMPT FOR NEW SWR
(1)	014254	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
(1)	014256	005046			CLR	-(SP)	::THE NEW SWR
(1)	014260	105777	164660	7\$:	TSTB	@\$TKS	::CHAR THERE?
(1)	014264	100375			BPL	7\$	::IF NOT TRY AGAIN
(1)							
(1)	014266	117746	164654		MOVB	@\$TKB, -(SP)	::PICK UP CHAR
(1)	014272	042716	177600		BIC	#^C177, (SP)	::MAKE IT 7-BIT ASCII
(1)							
(1)	014276	021627	000003		CMP	(SP), #3	::IS IT A CONTROL-C?
(1)	014302	001015			BNE	9\$	::BRANCH IF NOT
(1)	014304	104401	015074		TYPE	,\$CNTLC	::YES, ECHO CONTROL-C (^C)
(1)	014310	062706	000006		ADD	#6, SP	::CLEAN UP STACK
(1)	014314	123727	001135	000001	CMPB	\$INTAG, #1	::REENABLE TTY KEYBOARD INTERRUPTS?
(1)	014322	001003			BNE	8\$	::BRANCH IF NO
(1)	014324	012777	000100	164612	MOV	#100, @\$TKS	::ALLOW TTY KEYBOARD INTERRUPTS
(1)	014332	000137	001530	8\$:	JMP	BEGIN2	::CONTROL-C RESTART
(1)							
(1)							
(1)	014336	021627	000025	9\$:	CMP	(SP), #25	::IS IT A CONTROL-U?
(1)	014342	001005			BNE	10\$	::BRANCH IF NOT
(1)	014344	104401	015101		TYPE	,\$CNTLU	::YES, ECHO CONTROL-U (^U)
(1)	014350	062706	000006	20\$:	ADD	#6, SP	::IGNORE PREVIOUS INPUT
(1)	014354	000737			BR	19\$	::LET'S TRY IT AGAIN
(1)							
(1)							
(1)	014356	021627	000015	10\$:	CMP	(SP), #15	::IS IT A <CR>?
(1)	014362	001022			BNE	16\$	::BRANCH IF NO
(1)	014364	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
(1)	014370	001403			BEQ	11\$	::BRANCH IF YES
(1)	014372	016677	000002	164540	MOV	2(SP), @SWR	::SAVE NEW SWR
(1)	014400	062706	000006	11\$:	ADD	#6, SP	::CLEAN UP STACK
(1)	014404	104401	001171	14\$:	TYPE	,\$CRLF	::ECHO <CR> AND <LF>
(1)	014410	123727	001135	000001	CMPB	\$INTAG, #1	::RE-ENABLE TTY KBD INTERRUPTS?
(1)	014416	001003			BNE	15\$	::BRANCH IF NOT
(1)	014420	012777	000100	164516	MOV	#100, @\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
(1)	014426	000002		15\$:	RTI		::RETURN
(1)	014430	004737	016444	16\$:	JSR	PC, \$TYPEC	::ECHO CHAR
(1)	014434	021627	000060		CMP	(SP), #60	::CHAR < 0?
(1)	014440	002420			BLT	18\$	::BRANCH IF YES
(1)	014442	021627	000067		CMP	(SP), #67	::CHAR > 7?
(1)	014446	003015			BGT	18\$	::BRANCH IF YES
(1)	014450	042726	000060		BIC	#60, (SP)+	::STRIP-OFF ASCII
(1)	014454	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
(1)	014460	001403			BEQ	17\$	::BRANCH IF YES
(1)	014462	006316			ASL	(SP)	::NO, SHIFT PRESENT
(1)	014464	006316			ASL	(SP)	::CHAR OVER TO MAKE
(1)	014466	006316			ASL	(SP)	::ROOM FOR NEW ONE.
(1)	014470	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
(1)	014474	056616	177776		BIS	-2(SP), (SP)	::SET IN NEW CHAR
(1)	014500	000667			BR	7\$	::GET THE NEXT ONE
(1)	014502	104401	001170	18\$:	TYPE	,\$QUES	::TYPE ?<CR><LF>
(1)	014506	000720			BR	20\$	::SIMULATE CONTROL-U
(1)					.DSABL	LSB	
(1)							

```

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 014510 011646
(1) 014512 016666 000004 000002
(1) 014520 005066 000004
(2) 014524 005046
(2) 014526 012746 014534
(2) 014532 000002
(2) 014534
(1) 014534 005737 013600
(1) 014540 001775
(1) 014542 005337 013600
(1) 014546 117766 177032 000004
(1) 014554 005237 013604
(1) 014560 023727 013604 013646
(1) 014566 001003
(1) 014570 012737 013606 013604
(1) 014576 000002
(2)
(1)
(1)
(1)
(1)
(1)
(1) 014600 010346
(1) 014602 005046
(1) 014604 012703 015034
(1) 014610 022703 015074
(1) 014614 101456
(1) 014616 104411
(1) 014620 112613
(1) 014622 122713 000177
(1) 014626 001022
(1) 014630 005716
(1) 014632 001007
(1) 014634 112737 000134 015032
(1) 014642 104401 015032
(1) 014646 012716 177777
(1) 014652 005303
(1) 014654 020327 015034
(1) 014660 103434
(1) 014662 111337 015032
(1) 014666 104401 015032
(1) 014672 000746
(1) 014674 005716
(1) 014676 001406
(1) 014700 112737 000134 015032
(1) 014706 104401 015032

::*****
::THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
::CALL:
::*
RDCHR
::*
RETURN HERE
::*
::GET A CHARACTER FROM THE QUEUE
::CHARACTER IS ON THE STACK
::WITH PARITY BIT STRIPPED OFF
::
$RDCHR: MOV (SP),-(SP)
MOV 4(SP),2(SP)
CLR 4(SP)
CLR -(SP)
MOV #64$,-(SP)
RTI
::PUSH DOWN THE PC AND
::THE PS
::GET READY FOR A CHARACTER
::PUT NEW PS ON STACK
::PUT NEW PC ON STACK
::POP NEW PC AND PS
64$:
1$: TST $TKCNT
BEQ 1$
DEC $TKCNT
MOVB @ $TKQOUT,4(SP)
INC $TKQOUT
CMP $TKQOUT,$TKQEND
BNE 2$
MOV # $TKQSRT,$TKQOUT
RTI
::WAIT ON A CHARACTER
::DECREMENT THE COUNTER
::GET ONE CHARACTER
::UPDATE THE POINTER
::DID IT GO OFF OF THE END?
::BRANCH IF NO
::RESET THE POINTER
::RETURN
::*****
::THIS ROUTINE WILL INPUT A STRING FROM THE TTY
::CALL:
::*
RDLIN
::*
RETURN HERE
::*
::INPUT A STRING FROM THE TTY
::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
::TERMINATOR WILL BE A BYTE OF ALL 0'S
$RDLIN: MOV R3,-(SP)
CLR -(SP)
1$: MOV # $TTYIN,R3
2$: CMP # $TTYIN+32.,R3
BLOS 4$
RDCHR
MOVB (SP)+,(R3)
10$: CMPB #177,(R3)
BNE 5$
TST (SP)
BNE 6$
MOVB #' \,9$
TYPE ,9$
MOV #-1,(SP)
6$: DEC R3
CMP R3,# $TTYIN
BLO 4$
MOVB (R3),9$
TYPE ,9$
BR 2$
5$: TST (SP)
BEQ 7$
MOVB #' \,9$
TYPE ,9$
::SAVE R3
::CLEAR THE RUBOUT KEY
::GET ADDRESS
::BUFFER FULL?
::BR IF YES
::GO READ ONE CHARACTER FROM THE TTY
::GET CHARACTER
::IS IT A RUBOUT
::BR IF NO
::IS THIS THE FIRST RUBOUT?
::BR IF NO
::TYPE A BACK SLASH
::SET THE RUBOUT KEY
::BACKUP BY ONE
::STACK EMPTY?
::BR IF YES
::SETUP TO TYPEOUT THE DELETED CHAR.
::GO TYPE
::GO READ ANOTHER CHAR.
::RUBOUT KEY SET?
::BR IF NO
::TYPE A BACK SLASH

```


(1)	014712	005016				CLR	(SP)	::CLEAR THE RUBOUT KEY
(1)	014714	122713	000025		7\$:	CMPB	#25,(R3)	::IS CHARACTER A CTRL U?
(1)	014720	001003				BNE	8\$	::BR IF NO
(1)	014722	104401	015101			TYPE	,\$CNTLU	::TYPE A CONTROL 'U'
(1)	014726	000726				BR	1\$	::GO START OVER
(1)	014730	122713	000022		8\$:	CMPB	#22,(R3)	::IS CHARACTER A '^R'?
(1)	014734	001011				BNE	3\$	::BRANCH IF NO
(1)	014736	105013				CLRB	(R3)	::CLEAR THE CHARACTER
(1)	014740	104401	001171			TYPE	,\$CRLF	::TYPE A 'CR' & 'LF'
(1)	014744	104401	015034			TYPE	,\$TTYIN	::TYPE THE INPUT STRING
(1)	014750	000717				BR	2\$	::GO PICKUP ANOTHER CHACTER
(1)	014752	104401	001170		4\$:	TYPE	,\$QUES	::TYPE A '?'
(1)	014756	000712				BR	1\$	::CLEAR THE BUFFER AND LOOP
(1)	014760	111337	015032		3\$:	MOVB	(R3),9\$	::ECHO THE CHARACTER
(1)	014764	104401	015032			TYPE	,\$	
(1)	014770	122723	000015			CMPB	#15,(R3)+	::CHECK FOR RETURN
(1)	014774	001305				BNE	2\$	::LOOP IF NOT RETURN
(1)	014776	105063	177777			CLRB	-1(R3)	::CLEAR RETURN (THE 15)
(1)	015002	104401	001172			TYPE	,\$LF	::TYPE A LINE FEED
(1)	015006	005726				TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
(1)	015010	012603				MOV	(SP)+,R3	::RESTORE R3
(1)	015012	011646				MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
(1)	015014	016666	000004	000002		MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
(1)	015022	012766	015034	000004		MOV	,\$TTYIN,4(SP)	
(1)	015030	000002				RTI		::RETURN
(1)	015032	000			9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
(1)	015033	000				.BYTE	0	::TERMINATOR
(1)	015034	000040				.BLKB	32.	::RESERVE 32. BYTES FOR TTY INPUT
(1)	015074	041536	005015	000		,\$CNTLC:	,\$ASCIIZ	:/^C/<15><12>
(1)	015101	136	006525	000012		,\$CNTLU:	,\$ASCIIZ	:/^U/<15><12>
(1)	015106	043536	005015	000		,\$CNTLG:	,\$ASCIIZ	:/^G/<15><12>
(1)	015113	015	051412	051127		,\$MSWR:	,\$ASCIIZ	<15><12>/SWR = /
(1)	015120	036440	000040					
(1)	015124	020040	042516	020127		,\$MNEW:	,\$ASCIIZ	/ NEW = /
(1)	015132	020075	000					
(1)	015136					.EVEN		

1336

.SBTTL POWER DOWN AND UP ROUTINES

```

(1)
(2)
(1)
(1) 015240 012737 015400 000024 $PWRDN: MOV $ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 015246 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
(3) 015254 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 015256 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 015260 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 015262 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 015264 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
(3) 015266 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(3) 015270 017746 163644 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
(1) 015274 010637 015404 MOV SP,$SAVR6 ;;SAVE SP
(1) 015300 012737 015312 000024 MOV #PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 015306 000000 HALT
(1) 015310 000776 BR -2 ;;HANG UP
(1)
(2)
(1)
(1) 015312 012737 015400 000024 $PWRUP: MOV $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1) 015320 013706 015404 MOV $SAVR6,SP ;;GET SP
(1) 015324 005037 015404 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
(1) 015330 005237 015404 1$: INC $SAVR6 ;;WAIT FOR THE INC
(1) 015334 001375 BNE 1$ ;;OF WORD
(3) 015336 012677 163576 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
(3) 015342 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
(3) 015344 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
(3) 015346 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 015350 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 015352 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 015354 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 015356 012737 015240 000024 MOV $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 015364 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
(1) 015372 104401 TYPE $PWRMG: .WORD $POWER ;;REPORT THE POWER FAILURE
(1) 015374 015406 RTI ;;POWER FAIL MESSAGE POINTER
(1) 015376 000002 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
(1) 015400 000000 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 015402 000776 $SAVR6: 0 ;;PUT THE SP HERE
(1) 015404 000000 $POWER: .ASCIIZ <15><12>'POWER'
(1) 015406 005015 047520 042527 .EVEN
(1) 015414 000122

```

```

1338 .SBTTL SCOPE HANDLER ROUTINE
(1)
(2) *****
(1) *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1) *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) *SW14=1 LOOP ON TEST
(1) *SW11=1 INHIBIT ITERATIONS
(1) *SW09=1 LOOP ON ERROR
(1) *SW08=1 LOOP ON TEST IN SWR<7:0>
(1) *CALL
(1) * SCOPE ;;SCOPE=IOT
(1)
(1) $SCOPE:
(1) 015416 104410 040000 163512 1$: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 015420 032777 040000 163512 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
(1) 015426 001114 040000 163512 1$: BNE $OVER ;;YES IF SW14=1
(1) *****START OF CODE FOR THE XOR TESTER*****
(1) 015430 000416 040000 163512 1$: XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1) 015432 013746 000004 000004 1$: MOV @#ERRVEC,-(SP) ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1) 015436 012737 015456 000004 1$: MOV #5$,@#ERRVEC ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 015444 005737 177060 000004 1$: TST @#177060 ;;SET FOR TIMEOUT
(1) 015450 012637 000004 000004 1$: MOV (SP)+,@#ERRVEC ;;TIME OUT ON XOR?
(1) 015454 000463 000004 000004 1$: BR $SVLAD ;;RESTORE THE ERROR VECTOR
(1) 015456 022626 000004 000004 5$: CMP (SP)+,(SP)+ ;;GO TO THE NEXT TEST
(1) 015460 012637 000004 000004 5$: MOV (SP)+,@#ERRVEC ;;CLEAR THE STACK AFTER A TIME OUT
(1) 015464 000423 000004 000004 5$: BR 7$ ;;RESTORE THE ERROR VECTOR
(1) 015466 032777 000400 163444 6$: *****END OF CODE FOR THE XOR TESTER*****
(1) 015474 001404 163436 001102 2$: BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
(1) 015476 127737 163436 001102 2$: BEQ 2$ ;;BR IF NO
(1) 015504 001465 001103 001103 2$: CMPB @SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
(1) 015506 105737 001103 001103 2$: BEQ $OVER ;;BR IF YES
(1) 015512 001421 001115 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
(1) 015514 123737 001115 001103 2$: BEQ 3$ ;;BR IF NO
(1) 015522 101015 001000 163406 2$: CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 015524 032777 001000 163406 2$: BHI 3$ ;;BR IF NO
(1) 015532 001404 001110 001106 7$: BIT #BIT09,@SWR ;;LOOP ON ERROR?
(1) 015534 013737 001110 001106 7$: BEQ 4$ ;;BR IF NO
(1) 015542 000446 001103 001160 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 015544 105037 001103 001160 7$: BR $OVER
(1) 015550 005037 001160 001160 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
(1) 015554 000415 004000 163354 3$: CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 015556 032777 004000 163354 3$: BR 1$ ;;ESCAPE TO THE NEXT TEST
(1) 015564 001011 001202 001104 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1) 015566 005737 001202 001104 3$: BNE 1$ ;;BR IF YES
(1) 015572 001406 001104 001160 3$: TST $PASS ;;IF FIRST PASS OF PROGRAM
(1) 015574 005237 001104 001160 3$: BEQ 1$ ;;INHIBIT ITERATIONS
(1) 015600 023737 001160 001104 3$: INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 015606 002024 001102 001200 3$: CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 015610 012737 000001 001104 3$: BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 015616 013737 015674 001160 3$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1) 015624 105237 001102 001200 3$: MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 015630 113737 001102 001200 3$: $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
(1) 015630 113737 001102 001200 3$: MOVB $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX

```


[illegible]

```

(1) 016062 5$: CMP #SENDAD,@#42 ::ACT-11 AUTO-ACCEPT?
(1) 016062 022737 010374 000042 BNE 6$ ::BRANCH IF NO
(1) 016070 001001 HALT ::YES
(1) 016072 000000
(1) 016074 6$: RTI ::RETURN
(1) 016074 000002 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE

1340
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 016076 $ERRTYP:
(1) 016076 104401 001171 TYPE , $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 016102 010046 MOV RO,-(SP) ::SAVE RO
(1) 016104 005000 CLR RO ::PICKUP THE ITEM INDEX
(1) 016106 153700 001114 BISB @#$ITEMB,RO
(1) 016112 001004 BNE 1$ ::IF ITEM NUMBER IS ZERO, JUST
(1)
(2) 016114 013746 001116 MOV $ERRPC,-(SP) ::TYPE THE PC OF THE ERROR
(2)
(2) 016120 104402 TYPOC ::SAVE $ERRPC FOR TYPEOUT
(1) 016122 000426 BR 6$ ::ERROR ADDRESS
(1) 016124 005300 1$: DEC RO ::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 016126 006300 ASL RO ::GET OUT
(1) 016130 006300 ASL RO ::ADJUST THE INDEX SO THAT IT WILL
(1) 016132 006300 ASL RO ::WORK FOR THE ERROR TABLE
(1) 016134 062700 001256 ADD # $ERRTB,RO ::FORM TABLE POINTER
(1) 016140 012037 016150 MOV (RO)+,2$ ::PICKUP 'ERROR MESSAGE' POINTER
(1) 016144 001404 BEQ 3$ ::SKIP TYPEOUT IF NO POINTER
(1) 016146 104401 TYPE ::TYPE THE 'ERROR MESSAGE'
(1) 016150 000000 2$: .WORD 0 ::'ERROR MESSAGE' POINTER GOES HERE
(1) 016152 104401 001171 TYPE , $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 016156 012037 016166 3$: MOV (RO)+,4$ ::PICKUP 'DATA HEADER' POINTER
(1) 016162 001404 BEQ 5$ ::SKIP TYPEOUT IF 0
(1) 016164 104401 TYPE ::TYPE THE 'DATA HEADER'
(1) 016166 000000 4$: .WORD 0 ::'DATA HEADER' POINTER GOES HERE
(1) 016170 104401 001171 TYPE , $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 016174 011000 5$: MOV (RO),RO ::PICKUP 'DATA TABLE' POINTER
(1) 016176 001004 BNE 7$ ::GO TYPE THE DATA
(1) 016200 012600 6$: MOV (SP)+,RO ::RESTORE RO
(1) 016202 104401 001171 TYPE , $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 016206 000207 RTS PC ::RETURN
(1) 016210 7$:
(2) 016210 013046 MOV @ (RO)+,-(SP) ::SAVE @ (RO)+ FOR TYPEOUT
(2) 016212 104402 TYPOC ::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 016214 005710 TST (RO) ::IS THERE ANOTHER NUMBER?
(1) 016216 001770 BEQ 6$ ::BR IF NO
(1) 016220 104401 016226 TYPE 8$ ::TYPE TWO(2) SPACES
(1) 016224 000771 BR 7$ ::LOOP
(1) 016226 020040 000 8$: .ASCIZ / / ::TWO(2) SPACES
(1) 016232 .EVEN

```


1342

```

.SBTL  TYPE ROUTINE

```

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

```

:*****
:*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
:*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
:*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
:*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
:*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.

```

```

; *CALL:
; *1) USING A TRAP INSTRUCTION
; *      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
; *OR
; *      TYPE
; *      MESADR
; *

```

Address	Op Code	Op Name	Comments
(1) 016232	105737	001157	\$TYPE: TSTB \$TPFLG :: IS THERE A TERMINAL?
(1) 016236	100002		BPL 1\$:: BR IF YES
(1) 016240	000000		HALT :: HALT HERE IF NO TERMINAL
(1) 016242	000430		BR 3\$:: LEAVE
(1) 016244	010046	1\$:	MOV R0,-(SP) :: SAVE R0
(1) 016246	017600	000002	MOV @2(SP),R0 :: GET ADDRESS OF ASCIZ STRING
(1) 016252	122737	000001 001214	CMPB #APTENV,\$ENV :: RUNNING IN APT MODE
(1) 016260	001011		BNE 62\$:: NO,GO CHECK FOR APT CONSOLE
(1) 016262	132737	000100 001215	BITB #APTPOOL,\$ENV :: SPOOL MESSAGE TO APT
(1) 016270	001405		BEQ 62\$:: NO,GO CHECK FOR CONSOLE
(1) 016272	010037	016302	MOV R0,61\$:: SETUP MESSAGE ADDRESS FOR APT
(1) 016276	004737	016574	JSR PC,\$ATY3 :: SPOOL MESSAGE TO APT
(1) 016302	000000		61\$: .WORD 0 :: MESSAGE ADDRESS
(1) 016304	132737	000040 001215	62\$: BITB #APTCSUP,\$ENV :: APT CONSOLE SUPPRESSED
(1) 016312	001003		BNE 60\$:: YES,SKIP TYPE OUT
(1) 016314	112046	2\$:	MOVB (R0)+,-(SP)- :: PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 016316	001005		BNE 4\$:: BR IF IT ISN'T THE TERMINATOR
(1) 016320	005726		TST (SP)+ :: IF TERMINATOR POP IT OFF THE STACK
(1) 016322	012600	60\$:	MOV (SP)+,R0 :: RESTORE R0
(1) 016324	062716	000002	3\$: ADD #2,(SP) :: ADJUST RETURN PC
(1) 016330	000002		RTI :: RETURN
(1) 016332	122716	000011	4\$: CMPB #HT,(SP) :: BRANCH IF <HT>
(1) 016336	001430		BEQ 8\$
(1) 016340	122716	000200	CMPB #CRLF,(SP) :: BRANCH IF NOT <CRLF>
(1) 016344	001006		BNE 5\$
(1) 016346	005726		TST (SP)+ :: POP <CR><LF> EQUIV
(1) 016350	104401		TYPE :: TYPE A CR AND LF
(1) 016352	001171		\$CRLF
(1) 016354	105037	016562	CLRB \$CHARCNT :: CLEAR CHARACTER COUNT
(1) 016360	000755		BR 2\$:: GET NEXT CHARACTER
(1) 016362	004737	016444	5\$: JSR PC,\$TYPEC :: GO TYPE THIS CHARACTER
(1) 016366	123726	001156	6\$: CMPB \$FILLC,(SP)+ :: IS IT TIME FOR FILLER CHARS.?
(1) 016372	001350		BNE 2\$:: IF NO GO GET NEXT CHAR.
(1) 016374	013746	001154	MOV \$NULL,-(SP) :: GET # OF FILLER CHARS. NEEDED
(1) 016400	105366	000001	7\$: DECB 1(SP) :: AND THE NULL CHAR.
(1) 016404	002770		BLT 6\$:: DOES A NULL NEED TO BE TYPED?
(1) 016406	004737	016444	JSR PC,\$TYPEC :: BR IF NO--GO POP THE NULL OFF OF STACK
(1) 016412	105337	016562	DECB \$CHARCNT :: GO TYPE A NULL
			:: DO NOT COUNT AS A COUNT

```

(1) 016416 000770 BR 7$ ::LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 016420 112716 000040 8$: MOVB #' (SP) ::REPLACE TAB WITH SPACE
(1) 016424 004737 016444 9$: JSR PC,$TYPEC ::TYPE A SPACE
(1) 016430 132737 000007 016562 BITB #7,$CHARCNT ::BRANCH IF NOT AT
(1) 016436 001372 BNE 9$ ::TAB STOP
(1) 016440 005726 TST (SP)+ ::POP SPACE OFF STACK
(1) 016442 000724 BR 2$ ::GET NEXT CHARACTER
(1) 016444 $TYPEC:
(1) 016444 105777 162474 TSTB @STKS ::CHAR IN KYBD BUFFER? ;MJD001
(1) 016450 100022 BPL 10$ ::BR IF NOT ;MJD001
(1) 016452 017746 162470 MOV @STKB,-(SP) ::GET CHAR ;MJD001
(1) 016456 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS ;MJD001
(1) 016462 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF ;MJD001
(1) 016466 001012 BNE 102$ ::BR IF NOT ;MJD001
(1) 016470 101$:
(1) 016470 105777 162450 TSTB @STKS ::WAIT FOR CHAR ;MJD001
(1) 016474 100375 BPL 101$ ;MJD001
(1) 016476 117716 162444 MOVB @STKB,(SP) ::GET CHAR ;MJD001
(1) 016502 042716 177600 BIC #177600,(SP) ::STRIP IT ;MJD001
(1) 016506 122716 000021 CMPB #$XON,(SP) ::WAS IT XON? ;MJD001
(1) 016512 001366 BNE 101$ ::BR IF NOT ;MJD001
(1) 016514 102$:
(1) 016514 005726 TST (SP)+ ::FIX STACK ;MJD001
(1) 016516 10$:
(1) 016516 105777 162426 TSTB @STPS ::WAIT UNTIL PRINTER IS READY ;MJD001
(1) 016522 100375 BPL 10$ ;MJD001
(1) 016524 116677 000002 162420 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 016532 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
(1) 016540 001003 BNE 1$ ::BRANCH IF NO
(1) 016542 105037 016562 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT
(1) 016546 000406 BR $TYPEX ::EXIT
(1) 016550 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
(1) 016556 001402 BEQ $TYPEX ::BRANCH IF YES
(1) 016560 105227 INCB (PC)+ ::COUNT THE CHARACTER
(1) 016562 000000 $CHARCNT: .WORD 0 ::CHARACTER COUNT STORAGE
(1) 016564 000207 $TYPEX: RTS PC
(1)
1343 .SBTTL APT COMMUNICATIONS ROUTINE
(1)
(2) ::*****
(1) 016566 112737 000001 017032 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
(1) 016574 112737 000001 017030 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
(1) 016602 000403 BR $ATYC
(1) 016604 112737 000001 017032 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
(1) 016612 $ATYC:
(3) 016612 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
(3) 016614 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
(1) 016616 105737 017030 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
(1) 016622 001450 BEQ 5$ ::IF NOT: BR
(1) 016624 122737 000001 001214 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
(1) 016632 001031 BNE 3$ ::IF NOT: BR
(1) 016634 132737 000100 001215 BITB #APTPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
(1) 016642 001425 BEQ 3$ ::IF NOT: BR

```


(1)	016644	017600	000004		MOV	@4(SP),R0	::GET MESSAGE ADDR.
(1)	016650	062766	000002	000004	ADD	#2,4(SP)	::BUMP RETURN ADDR.
(1)	016656	005737	001174	1\$:	TST	\$MSGTYPE	::SEE IF DONE W/ LAST XMISSION?
(1)	016662	001375			BNE	1\$	::IF NOT: WAIT
(1)	016664	010037	001210		MOV	R0,\$MSGAD	::PUT ADDR IN MAILBOX
(1)	016670	105720		2\$:	TSTB	(R0)+	::FIND END OF MESSAGE
(1)	016672	001376			BNE	2\$	
(1)	016674	163700	001210		SUB	\$MSGAD,R0	::SUB START OF MESSAGE
(1)	016700	006200			ASR	R0	::GET MESSAGE LGTH IN WORDS
(1)	016702	010037	001212		MOV	R0,\$MSGGLT	::PUT LENGTH IN MAILBOX
(1)	016706	012737	000004	001174	MOV	#4,\$MSGTYPE	::TELL APT TO TAKE MSG.
(1)	016714	000413			BR	5\$	
(1)	016716	017637	000004	016742	MOV	@4(SP),4\$	::PUT MSG ADDR IN JSR LINKAGE
(1)	016724	062766	000002	000004	ADD	#2,4(SP)	::BUMP RETURN ADDRESS
(3)	016732	013746	177776		MOV	177776,-(SP)	::PUSH 177776 ON STACK
(1)	016736	004737	016232		JSR	PC,\$TYPE	::CALL TYPE MACRO
(1)	016742	000000		4\$:	.WORD	0	
(1)	016744			5\$:			
(1)	016744	105737	017032	10\$:	TSTB	\$FFLG	::SHOULD REPORT FATAL ERROR?
(1)	016750	001416			BEQ	12\$	::IF NOT: BR
(1)	016752	005737	001214		TST	\$ENV	::RUNNING UNDER APT?
(1)	016756	001413			BEQ	12\$	::IF NOT: BR
(1)	016760	005737	001174	11\$:	TST	\$MSGTYPE	::FINISHED LAST MESSAGE?
(1)	016764	001375			BNE	11\$	::IF NOT: WAIT
(1)	016766	017637	000004	001176	MOV	@4(SP),\$FATAL	::GET ERROR #
(1)	016774	062766	000002	000004	ADD	#2,4(SP)	::BUMP RETURN ADDR.
(1)	017002	005237	001174		INC	\$MSGTYPE	::TELL APT TO TAKE ERROR
(1)	017006	105037	017032	12\$:	CLRB	\$FFLG	::CLEAR FATAL FLAG
(1)	017012	105037	017031		CLRB	\$LFLG	::CLEAR LOG FLAG
(1)	017016	105037	017030		CLRB	\$MFLG	::CLEAR MESSAGE FLAG
(3)	017022	012601			MOV	(SP)+,R1	::POP STACK INTO R1
(3)	017024	012600			MOV	(SP)+,R0	::POP STACK INTO R0
(1)	017026	000207			RTS	PC	::RETURN
(1)	017030	000		\$MFLG:	.BYTE	0	::MESSG. FLAG
(1)	017031	000		\$LFLG:	.BYTE	0	::LOG FLAG
(1)	017032	000		\$FFLG:	.BYTE	0	::FATAL FLAG
(1)		017034			.EVEN		
(1)		000200			APTSIZE=200		
(1)		000001			APTENV=001		
(1)		000100			APTSPool=100		
(1)		000040			APTC SUP=040		

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

(1)					::*****
(2)					:*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)					:*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)					:*\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)					:*CALL:
(1)					:* MOV NUM,-(SP) ::NUMBER TO BE TYPED
(1)					:* TYPOS ::CALL FOR TYPEOUT
(1)					:* .BYTE N ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)					:* .BYTE M ::M=1 OR 0
(1)					:* ::1=TYPE LEADING ZEROS
(1)					:* ::0=SUPPRESS LEADING ZEROS
(1)					:*
(1)					:*\$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)					:*\$TYPOS OR \$TYPOC
(1)					:*CALL:
(1)					:* MOV NUM,-(SP) ::NUMBER TO BE TYPED
(1)					:* TYPON ::CALL FOR TYPEOUT
(1)					:*
(1)					:*\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)					:*CALL:
(1)					:* MOV NUM,-(SP) ::NUMBER TO BE TYPED
(1)					:* TYPOC ::CALL FOR TYPEOUT
(1)					:*
(1)	017034	017646	000000		\$TYPOS: MOV a(SP),-(SP) ::PICKUP THE MODE
(1)	017040	116637	000001	017257	MOVB 1(SP),\$OFILL ::LOAD ZERO FILL SWITCH
(1)	017046	112637	017261		MOVB (SP)+,\$OMODE+1 ::NUMBER OF DIGITS TO TYPE
(1)	017052	062716	000002		ADD #2,(SP) ::ADJUST RETURN ADDRESS
(1)	017056	000406			BR \$TYPON
(1)	017060	112737	000001	017257	\$TYPOC: MOVB #1,\$OFILL ::SET THE ZERO FILL SWITCH
(1)	017066	112737	000006	017261	MOVB #6,\$OMODE+1 ::SET FOR SIX(6) DIGITS
(1)	017074	112737	000005	017256	\$TYPON: MOVB #5,\$OCNT ::SET THE ITERATION COUNT
(1)	017102	010346			MOV R3,-(SP) ::SAVE R3
(1)	017104	010446			MOV R4,-(SP) ::SAVE R4
(1)	017106	010546			MOV R5,-(SP) ::SAVE R5
(1)	017110	113704	017261		MOVB \$OMODE+1,R4 ::GET THE NUMBER OF DIGITS TO TYPE
(1)	017114	005404			NEG R4
(1)	017116	062704	000006		ADD #6,R4 ::SUBTRACT IT FOR MAX. ALLOWED
(1)	017122	110437	017260		MOVB R4,\$OMODE ::SAVE IT FOR USE
(1)	017126	113704	017257		MOVB \$OFILL,R4 ::GET THE ZERO FILL SWITCH
(1)	017132	016605	000012		MOV 12(SP),R5 ::PICKUP THE INPUT NUMBER
(1)	017136	005003			CLR R3 ::CLEAR THE OUTPUT WORD
(1)	017140	006105		1\$:	ROL R5 ::ROTATE MSB INTO "C"
(1)	017142	000404			BR 3\$::GO DO MSB
(1)	017144	006105		2\$:	ROL R5 ::FORM THIS DIGIT
(1)	017146	006105			ROL R5
(1)	017150	006105			ROL R5
(1)	017152	010503			MOV R5,R3
(1)	017154	006103		3\$:	ROL R3 ::GET LSB OF THIS DIGIT
(1)	017156	105337	017260		DECB \$OMODE ::TYPE THIS DIGIT?
(1)	017162	100016			BPL 7\$::BR IF NO
(1)	017164	042703	177770		BIC #177770,R3 ::GET RID OF JUNK
(1)	017170	001002			BNE 4\$::TEST FOR 0
(1)	017172	005704			TST R4 ::SUPPRESS THIS 0?
(1)	017174	001403			BEQ 5\$::BR IF YES


```

(1) 017176 005204 4$: INC R4 ::DON'T SUPPRESS ANYMORE 0'S
(1) 017200 052703 000060 BIS #'0,R3 ::MAKE THIS DIGIT ASCII
(1) 017204 052703 000040 5$: BIS #' ,R3 ::MAKE ASCII IF NOT ALREADY
(1) 017210 110337 017254 MOV R3,8$ ::SAVE FOR TYPING
(1) 017214 104401 017254 TYPE 8$ ::GO TYPE THIS DIGIT
(1) 017220 105337 017256 7$: DECB $OCNT ::COUNT BY 1
(1) 017224 003347 BGT 2$ ::BR IF MORE TO DO
(1) 017226 002402 BLT 6$ ::BR IF DONE
(1) 017230 005204 INC R4 ::INSURE LAST DIGIT ISN'T A BLANK
(1) 017232 000744 BR 2$ ::GO DO THE LAST DIGIT
(1) 017234 012605 6$: MOV (SP)+,R5 ::RESTORE R5
(1) 017236 012604 MOV (SP)+,R4 ::RESTORE R4
(1) 017240 012603 MOV (SP)+,R3 ::RESTORE R3
(1) 017242 016666 000002 000004 MOV 2(SP),4(SP) ::SET THE STACK FOR RETURNING
(1) 017250 012616 MOV (SP)+,(SP)
(1) 017252 000002 RTI ::RETURN
(1) 017254 000 8$: .BYTE 0 ::STORAGE FOR ASCII DIGIT
(1) 017255 000 .BYTE 0 ::TERMINATOR FOR TYPE ROUTINE
(1) 017256 000 $OCNT: .BYTE 0 ::OCTAL DIGIT COUNTER
(1) 017257 000 $OFILL: .BYTE 0 ::ZERO FILL SWITCH
(1) 017260 000000 $OMODE: .WORD 0 ::NUMBER OF DIGITS TO TYPE
1346 .SBTTL BINARY TO ASCII AND TYPE ROUTINE
(1)
(2)
(1) ::*****
(1) ::THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
(1) ::BINARY-ASCII NUMBER AND TYPE IT.
(1) ::CALL:
(1) * MOV NUMBER,-(SP) ::NUMBER TO BE TYPED
(1) * TYPBN ::TYPE IT
(1)
(1) $TYPBN: MOV R1,-(SP) ::SAVE R1 ON THE STACK
(1) 017262 010146 000006 MOV 6(SP),R1 ::GET THE INPUT NUMBER
(1) 017264 016601 000006 SEC ::SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1) 017270 000261 000060 017334 1$: MOVB #'0,$BIN ::SET CHARACTER TO AN ASCII '0'.
(1) 017272 112737 000060 017334 ROL R1 ::GET THIS BIT
(1) 017300 006101 BEQ 2$ ::DONE?
(1) 017302 001406 ADCB $BIN ::NO--SET THE CHARACTER EQUAL TO THIS BIT
(1) 017304 105537 017334 TYPE $BIN ::GO TYPE THIS BIT
(1) 017310 104401 017334 CLC ::CLEAR 'C' SO CAN KEEP TRACK OF BITS
(1) 017314 000241 BR 1$ ::GO DO THE NEXT BIT
(1) 017316 000765 2$: MOV (SP)+,R1 ::POP THE STACK INTO R1
(1) 017320 012601 000002 000004 MOV 2(SP),4(SP) ::ADJUST THE STACK
(1) 017322 016666 000002 000004 MOV (SP)+,(SP)
(1) 017330 012616 RTI ::RETURN TO USER
(1) 017332 000002 $BIN: .BYTE 0,0 ::STORAGE FOR ASCII CHAR. AND TERMINATOR
(1) 017334 000 000 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1347
(1)
(2)
(1) ::*****
(1) ::THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1) ::SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1) ::NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1) ::BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1) ::REPLACED WITH SPACES.
(1) ::CALL:
(1) * MOV NUM,-(SP) ::PUT THE BINARY NUMBER ON THE STACK
(1) * TYPDS ::GO TO THE ROUTINE

```

Address	Op Code	Op Name	Comments
017336	010046	MOV R0,-(SP)	::PUSH R0 ON STACK
017336	010146	MOV R1,-(SP)	::PUSH R1 ON STACK
017340	010246	MOV R2,-(SP)	::PUSH R2 ON STACK
017342	010346	MOV R3,-(SP)	::PUSH R3 ON STACK
017344	010546	MOV R5,-(SP)	::PUSH R5 ON STACK
017350	012746	MOV #20200,-(SP)	::SET BLANK SWITCH AND SIGN
017354	016605	MOV 20(SP),R5	::GET THE INPUT NUMBER
017360	100004	BPL 1\$	::BR IF INPUT IS POS.
017362	005405	NEG R5	::MAKE THE BINARY NUMBER POS.
017364	112766	MOVB #'-,1(SP)	::MAKE THE ASCII NUMBER NEG.
017372	005000	CLR R0	::ZERO THE CONSTANTS INDEX
017374	012703	MOV #SDBLK,R3	::SETUP THE OUTPUT POINTER
017400	112723	MOVB #' ,(R3)+	::SET THE FIRST CHARACTER TO A BLANK
017404	005002	CLR R2	::CLEAR THE BCD NUMBER
017406	016001	MOV \$DTBL(R0),R1	::GET THE CONSTANT
017412	160105	SUB R1,R5	::FORM THIS BCD DIGIT
017414	002402	BLT 4\$	::BR IF DONE
017416	005202	INC R2	::INCREASE THE BCD DIGIT BY 1
017420	000774	BR 3\$	
017422	060105	ADD R1,R5	::ADD BACK THE CONSTANT
017424	005702	TST R2	::CHECK IF BCD DIGIT=0
017426	001002	BNE 5\$	::FALL THROUGH IF 0
017430	105716	TSTB (SP)	::STILL DOING LEADING 0'S?
017432	100407	BMI 7\$	::BR IF YES
017434	106316	ASLB (SP)	::MSD?
017436	103003	BCC 6\$	::BR IF NO
017440	116663	MOVB 1(SP),-1(R3)	::YES--SET THE SIGN
017446	052702	BIS #'0,R2	::MAKE THE BCD DIGIT ASCII
017452	052702	BIS #' ,R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
017456	110223	MOVB R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
017460	005720	TST (R0)+	::JUST INCREMENTING
017462	020027	CMP R0,#10	::CHECK THE TABLE INDEX
017466	002746	BLT 2\$	::GO DO THE NEXT DIGIT
017470	003002	BGT 8\$	::GO TO EXIT
017472	010502	MOV R5,R2	::GET THE LSD
017474	000764	BR 6\$	::GO CHANGE TO ASCII
017476	105726	TSTB (SP)+	::WAS THE LSD THE FIRST NON-ZERO?
017500	100003	BPL 9\$	::BR IF NO
017502	116663	MOVB -1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
017510	105013	CLRB (R3)	::SET THE TERMINATOR
017512	012605	MOV (SP)+,R5	::POP STACK INTO R5
017514	012603	MOV (SP)+,R3	::POP STACK INTO R3
017516	012602	MOV (SP)+,R2	::POP STACK INTO R2
017520	012601	MOV (SP)+,R1	::POP STACK INTO R1
017522	012600	MOV (SP)+,R0	::POP STACK INTO R0
017524	104401	TYPE ,SDBLK	::NOW TYPE THE NUMBER
017530	016666	MOV 2(SP),4(SP)	::ADJUST THE STACK
017536	012616	MOV (SP)+,(SP)	
017540	000002	RTI	::RETURN TO USER
017542	023420	\$DTBL: 10000.	
017544	001750	1000.	
017546	000144	100.	
017550	000012	10.	
017552	000004	\$DBLK: .BLKW 4	


```

1349      .SBTTL  TRAP DECODER
(1)
(2)      ;*****
(1)      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)      ;*GO TO THAT ROUTINE.
(1)
(1)      $TRAP:  MOV     R0,-(SP)          ;;SAVE R0
(1)      017562  010046          000002      MOV     2(SP),R0          ;;GET TRAP ADDRESS
(1)      017564  016600          000002      TST      -(R0)          ;;BACKUP BY 2
(1)      017570  005740          000002      MOV     (R0),R0          ;;GET RIGHT BYTE OF TRAP
(1)      017572  111000          000002      ASL     R0              ;;POSITION FOR INDEXING
(1)      017574  006300          017616      MOV     $TRPAD(R0),R0      ;;INDEX TO TABLE
(1)      017576  016000          017616      RTS      R0              ;;GO TO ROUTINE
(1)      017602  000200          017616
(1)
(1)      ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1)      $TRAP2: MOV     (SP),-(SP)        ;;MOVE THE PC DOWN
(1)      017604  011646          000004  000002      MOV     4(SP),2(SP)      ;;MOVE THE PSW DOWN
(1)      017606  016666          000004  000002      RTI              ;;RESTORE THE PSW
(1)      017614  000002          000004  000002
(1)
(3)      .SBTTL  TRAP TABLE
(3)
(3)      ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)      ;*BY THE "TRAP" INSTRUCTION.
(3)
(3)      :      ROUTINE
(3)      :      -----
(3)      $TRPAD: .WORD  $TRAP2
(3)      017616  017604          017616      ;;CALL=TYPE          TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3)      017620  016232          017616      ;;CALL=TYPOC         TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)      017622  017060          017616      ;;CALL=TYPOS         TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)      017624  017034          017616      ;;CALL=TYPON         TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)      017626  017074          017616      ;;CALL=TYPDS         TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
(3)      017630  017336          017616      ;;CALL=TYPBN         TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
(3)      017632  017262          017616
(1)
(3)      017634  014236          017616      ;;CALL=GTSWR         TRAP+7(104407)  GET SOFT-SWR SETTING
(1)
(3)      017636  014146          017616      ;;CALL=CKSWR         TRAP+10(104410)  TEST FOR CHANGE IN SOFT-SWR
(3)      017640  014510          017616      ;;CALL=RDCHR         TRAP+11(104411)  TTY TYPEIN CHARACTER ROUTINE
(3)      017642  014600          017616      ;;CALL=RDLIN         TRAP+12(104412)  TTY TYPEIN STRING ROUTINE
(3)      017644  015136          017616      ;;CALL=RDOCT         TRAP+13(104413)  READ AN OCTAL NUMBER FROM TTY
(3)      017646  004112          017616      ;;CALL=CHECK         TRAP+14(104414)
(3)      017650  004104          017616      ;;CALL=CHKIT         TRAP+15(104415)
1350
1351      .EVEN  NOP
1352
1353      .END
1354

```

;JUST TO FIND THE LAST LOCATION OF THE PROGRAM

[illegible]

ATESTN=	000000	57																		
AUNIT =	000000	57																		
AUSWR =	000000	57																		
AVECT1=	000400	18#	57	101	102	103	104													
AVECT2=	000000	57																		
AZERO	011150	964*	967*	968	970*	973*	974	1280#												
BARF	001356	112#	132																	
BEGINA	007352	238	244	1117#	1153															
BEGIND	007412	180	1130#																	
BEGINL	007334	240	246	1111#																
BEGINW	007374	242	248	1124#																
BEGINO	001522	22	159#	1260																
BEGIN2	001530	23	161#	1332																
BGL	002560	263#	1112	1118																
BGST	001534	160	162#																	
BIT0 =	000001	15#	361	390	394	456	1130													
BIT00 =	000001	15#																		
BIT01 =	000002	15#																		
BIT02 =	000004	15#																		
BIT03 =	000010	15#																		
BIT04 =	000020	15#																		
BIT05 =	000040	15#																		
BIT06 =	000100	15#																		
BIT07 =	000200	15#																		
BIT08 =	000400	15#	1338																	
BIT09 =	001000	15#	1338	1339																
BIT1 =	000002	15#	1133																	
BIT10 =	002000	15#	1339																	
BIT11 =	004000	15#	1338																	
BIT12 =	010000	15#	275	1031	1040															
BIT13 =	020000	15#	1339																	
BIT14 =	040000	15#	279	379	381	1338														
BIT15 =	100000	15#	307	320	330	379	381	393	449											
BIT2 =	000004	15#	298	1136																
BIT3 =	000010	15#	302	1139																
BIT4 =	000020	15#	293	417	433	468	1142													
BIT5 =	000040	15#	289	404	449	451	1													

[illegible]

PIRQ	=	177772	15#			
PIRQVE	=	000240	15#			
PR0	=	000000	15#			
PR1	=	000040	15#			
PR2	=	000100	15#			
PR3	=	000140	15#			
PR4	=	000200	15#			
PR5	=	000240	15#			
PR6	=	000300	15#			
PR7	=	000340	15#			
PS	=	177776	15#			
PSW	=	177776	15#			
PWRVEC	=	000024	15#	162*	166*	1336*
QUEST		011131	233	1277#		
RDCHR	=	104411	1332	1349#		

[illegible]

[illegible]

V12	010272	1256#												
V2	010270	555	563	580	588	605	613	628	759	766	773	780	1255#	
WRAP	004136	497#	1119	1125										
\$APTHD	001000	56#												
\$ASTAT=	***** U	1343												
\$ATYC	016612	1343#												
\$ATY1	016566	1343#												
\$ATY3	016574	1342	1343#											
\$ATY4	016604	1339	1343#											
\$AUTOB	001134	57#	174*	178	1332									
\$BASE	001250	57#	1157	1158	1159	1160	1161							
\$BDADR	001122	57#												
\$BDDAT	001126	57#	315*	364*	380*	458*	487*	488	1218*	1220	1327	1328	1329	
\$BELL	001164	57#	1332	1339										
\$BIN	017334	1346#*												
\$CDW1	001254	57#												
\$CHARC	016562	1342#*												
\$CKSWR	014146	1332#	1349											
\$CMTAG	001100	57#	162											
\$CM3 =	000000	57#												
\$CNTLC	015074	1332#												
\$CNTLG	015106	1332#												
\$CNTLU	015101	1332#												
\$CPUOP	001222	57#												
\$CRLF	001171	57#	145	943	953	998	1088	1332	1339	1340	1342			
\$DBLK	017552	1347#												
\$DEVCT	001204	57#												
\$DEVM	001252	57#	1130	1133	1136	1139	1142	1145						
\$DOAGN	010404	1262#												
\$DTBL	017542	1347#												
\$ENDAD	010374	54	174	1262#	1339									
\$ENDCT	010342	162	1262#											
\$ENDMG	010413	1262#												
\$ENULL	010410	1262#												
\$ENV	001214	57#	174	1339	1342	1343								
\$ENVM	001215	57#	162	1342	1343									
\$EOP	010306	1114	1121	1127	1262#									
\$EOPCT	010334	162*	1262#											
\$ERFLG	001103	57#	125*	1338*	1339*									
\$ERMAX	001115	57#	162*	1338*										
\$ERROR	015676	162	1339#											
\$ERRPC	001116	57#	1327	1328	1329	1339*	1340							
\$ERRTB	001256	57#	1340											
\$ERRTY	016076	1339	1340#											
\$ERTTL	001112	57#	1339*											
\$ESCAP	001162	57#	124*	127*	162*	1338*	1339							
\$ETABL	001214	57#												
\$ETEND	001256	56	57#											
\$FATAL	001176	57#	1343*											
\$FFLG	017032	1343#*												
\$FILLC	001156	57#	1342											
\$FILLS	001155	57#	1342											
\$GDADR	001120	57#												
\$GDDAT	001124	57#	271*	274*	275	279*	284*	289*	293*	298*	302*	307*	311*	328*
		337*	365*	381*	393*	404*	405	417*	418	433*	434	449*	468*	469
		486	488	1216*	1219	1327	1328							

[illegible]

[illegible]

COMMENT	15#														
DUMWRN	62#	355	377												
ENDCOM	15#														
ERROR	15#	126	273	281	286	291	295	300	304	309	317	324	332	343	352
	366	382	396	410	423	442	460	472	519	526	533	540	556	564	581
	589	606	614	629	642	650	658	666	678	709	737	760	767	774	781
	816	845	874	902											
ESCAPE	15#	124													
GETPRI	15#														
GETSWR	15#	174#													
MULT	15#														
NEWST	15#	264	270	278	282	288	292	297	301	306	310	319	325	336	346
	354	376	389	401	414	428	446	465	477	497	511	543	568	593	617
	633	669	686	714	748	789	820	849	878	903					
POP	15#	1334	1336	1343	1347										
PUSH	15#	1334	1336	1343	1347										
REPORT	15#														
SCOPE	15#	270	278	282	288	292	297	301	306	310	319	325	336	346	354
	376	389	401	414	428	446	465	477	511	543	568	593	617	633	669
	686	714	748	789	820	849	878	903							
SETPRI	15#	1332													
SETTRA	1349#	1350	1351												
SETUP	15#	162													
SKIP	15#	276	316	368	403	416	430	432	448	459	467	489	513	545	570
	595	619	635	672	691	693	708	719	721	736	752	793	814	824	843
	853	872	881	900											
SLASH	15#														
SPACE	15#														
STARS	15#	54	56	57	264	270	278	282	288	292	297	301	306	310	319
	325	336	346	354	376	389	401	414	428	446	465	477	497	511	543
	568	593	617	633	669	686	714	748	789	820	849	878	903	1262	1332
	1334	1336	1338	1339	1340	1342	1343	1345	1346	1347	1349				
SWRSU	15#	162#													
TRMTRP	1349#														
TYPBIN	15#														
TYPDEC	15#	1262													
TYPNAM	15#	174													
TYPNUM	15#														
TYPOCS	15#	932	937	1234	1236										
TYPOCT	15#	1332	1340												
TYPTXT	15#														
\$\$CMRE	57#														
\$\$CMTM	57#														
\$\$ESCA	15#														
\$\$NEWT	15#	264	270	278	282	288	292	297	301	306	310	319	325	336	346
	354	376	389	401	414	428	446	465	477	497	511	543	568	593	617
	633	669	686	714	748	789	820	849	878	903					
\$\$SET	1349#	1350	1351												
\$\$SETM	162#														
\$\$SKIP	15#	316	368	403	416	430	432	448	467	513	545	570	595	619	635
	672	691	693	719	721	752	793	814	824	843	853	872	881	900	
.EQUAT	8#	15													
.HEADE	8#	14													
.SETUP	10#	58													
.SWRHI	10#	16													
.SWRLO	16#														

.SACT1	11#	54
.SAPT8	11#	57#
.SAPTH	11#	56
.SAPTY	11#	1343
.SCATC	8#	22
.SCMTA	8#	57
.SEOP	8#	1262
.SERRO	8#	1339
.SERRT	10#	1340
.SPARM	9#	
.SPOWE	9#	1336
.SRAND	11#	
.SRDOC	11#	1334
.SREAD	9#	1332
.SSAVE	9#	
.SSCOP	9#	1338
.SSPAC	10#	
.SSWDO	10#	
.STRAP	10#	1349
.STYPB	9#	1346
.STYPD	11#	1347
.STYPE	10#	1342
.STYPO	9#	1345

. ABS. 017654 000 OVR RW REL LCL D

ERRORS DETECTED: 0

CVAXAB, CVAXAB/CRF=CVAXAB
RUN-TIME: 21 9 1 SECONDS
RUN-TIME RATIO: 72/32=2.2
CORE USED: 26K (51 PAGES)