AXV11-C,
ADV11-C

AXV11-C/ADV11-C DIAG
CVAXAB0

AH-S895B-MC
FICHE 1 OF 1

IDENTIFICATION

Product Code: AC-S893B-MC

Product Name: CVAXABO AXV11-C/ADV11-C DIAG

Date: Feb. 1983

Maintainer: RAY SHOOP

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 KWV11-C CLOCK modules. The program does not test all the functions of the AAV11-C or KWV11-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 "$DEVM" (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 KWV11-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 folllowing 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 Typeouts

At end of pass, the following typeout 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 verify the level, he depresses the "RETURN". The program will the 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 load 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.

```
                                ;DEVELOPED USING SYSMAC.C4
                                .TITLE  MAINDEC-11-CVAXA-B
                                ;*COPYRIGHT (C) 1983
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY R.SHOOP
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
                                ;*
                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                                ;*MISCELLANEOUS DEFINITIONS
000011                          HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                          LF=     12              ;;CODE FOR LINE FEED
000015                          CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                          PS=     177776          ;;PROCESSOR STATUS WORD
                                .EQUIV  PS,PSW
177774                          STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                          R0=     %0              ;;GENERAL REGISTER
000001                          R1=     %1              ;;GENERAL REGISTER
000002                          R2=     %2              ;;GENERAL REGISTER
000003                          R3=     %3              ;;GENERAL REGISTER
000004                          R4=     %4              ;;GENERAL REGISTER
000005                          R5=     %5              ;;GENERAL REGISTER
000006                          R6=     %6              ;;GENERAL REGISTER
000007                          R7=     %7              ;;GENERAL REGISTER
000006                          SP=     %6              ;;STACK POINTER
000007                          PC=     %7              ;;PROGRAM COUNTER

                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3=    140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7

                                ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                          SW15=   100000
040000                          SW14=   40000
020000                          SW13=   20000
```

```
(1)             010000                  SW12=   10000
(1)             004000                  SW11=   4000
(1)             002000                  SW10=   2000
(1)             001000                  SW09=   1000
(1)             000400                  SW08=   400
(1)             000200                  SW07=   200
(1)             000100                  SW06=   100
(1)             000040                  SW05=   40
(1)             000020                  SW04=   20
(1)             000010                  SW03=   10
(1)             000004                  SW02=   4
(1)             000002                  SW01=   2
(1)             000001                  SW00=   1
(1)                                     .EQUIV  SW09,SW9
(1)                                     .EQUIV  SW08,SW8
(1)                                     .EQUIV  SW07,SW7
(1)                                     .EQUIV  SW06,SW6
(1)                                     .EQUIV  SW05,SW5
(1)                                     .EQUIV  SW04,SW4
(1)                                     .EQUIV  SW03,SW3
(1)                                     .EQUIV  SW02,SW2
(1)                                     .EQUIV  SW01,SW1
(1)                                     .EQUIV  SW00,SW0
(1)
(1)                                     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)             100000                  BIT15=  100000
(1)             040000                  BIT14=  40000
(1)             020000                  BIT13=  20000
(1)             010000                  BIT12=  10000
(1)             004000                  BIT11=  4000
(1)             002000                  BIT10=  2000
(1)             001000                  BIT09=  1000
(1)             000400                  BIT08=  400
(1)             000200                  BIT07=  200
(1)             000100                  BIT06=  100
(1)             000040                  BIT05=  40
(1)             000020                  BIT04=  20
(1)             000010                  BIT03=  10
(1)             000004                  BIT02=  4
(1)             000002                  BIT01=  2
(1)             000001                  BIT0 =  1
(1)                                     .EQUIV  BIT09,BIT9
(1)                                     .EQUIV  BIT08,BIT8
(1)                                     .EQUIV  BIT07,BIT7
(1)                                     .EQUIV  BIT06,BIT6
(1)                                     .EQUIV  BIT05,BIT5
(1)                                     .EQUIV  BIT04,BIT4
(1)                                     .EQUIV  BIT03,BIT3
(1)                                     .EQUIV  BIT02,BIT2
(1)                                     .EQUIV  BIT01,BIT1
(1)                                     .EQUIV  BIT00,BIT0
(1)
(1)                                     ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)             000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)             000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)             000014                  TBITVEC=14              ;;"T" BIT
```

```
(1)           000014                  TRTVEC= 14              ;;TRACE TRAP
(1)           000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)           000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)           000024                  PWRVEC= 24              ;;POWER FAIL
(1)           000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)           000034                  TRAPVEC=34              ;;"TRAP" TRAP
(1)           000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)           000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
(1)           000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
 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)
(1)           000000                          .=0
(1)                                   ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)                                   ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)                                   ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)           000174                          .=174
(1)  000174   000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
(1)  000176   000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
(1)                                   .SBTTL  STARTING ADDRESS(ES)
(1)  000200   000137  001522                  JMP     @#BEGIN0        ;;JUMP TO STARTING ADDRESS OF PROGRAM
 23  000204   000137  001530                  JMP     @#BEGIN2                ;RESTART ADDRESS
 24
 25           000100                          .=100
 26  000100   000104  000340  000002          104,340,2                       ;"B EVENT" HANDLER
 27
 28           000140                          .=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
```

```
                000014                  CHAN14= 14
                000015                  CHAN15= 15
                000016                  CHAN16= 16
                000017                  CHAN17= 17

                000000                  GAIN00= 00
                000004                  GAIN01= 04
                000010                  GAIN10= 10
                000014                  GAIN11= 14


                                .SBTTL  ACT11 HOOKS

                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
                000144                  $SVPC=.                 ;SAVE PC
                000046                  .=46
000046          010374                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                  .=52
000052          000000                  .WORD   0               ;;2)SET LOC.52 TO ZERO
                000144                  .=$SVPC                 ;; RESTORE PC
                001000          .=1000
                                .SBTTL  APT PARAMETER BLOCK

                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
                001000                  .$X=.   ;;SAVE CURRENT LOCATION
                000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024          000200                  200     ;;FOR APT START UP
                000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044          001000                  $APTHDR ;;POINT TO APT HEADER BLOCK
                001000                  .=.$X   ;;RESET LOCATION COUNTER
                                ;;******************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001174                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000550                  $TSTM:  .WORD   360.    ;;RUN TIM OF LONGEST TEST
001006  000132                  $PASTM: .WORD   90.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000550                  $UNITM: .WORD   360.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000031                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
    57                                  .SBTTL  COMMON TAGS
   (1)
   (2)                                  ;;*****************************************************************
   (1)                                  ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
   (1)                                  ;*USED IN THE PROGRAM.
   (1)
   (1)                 001100                   .=1100
   (1)  001100                          $CMTAG:                     ;;START OF COMMON TAGS
   (1)  001100  000000                          .WORD   0
   (1)  001102     000                  $TSTNM: .BYTE   0           ;;CONTAINS THE TEST NUMBER
   (1)  001103     000                  $ERFLG: .BYTE   0           ;;CONTAINS ERROR FLAG
   (1)  001104  000000                  $ICNT:  .WORD   0           ;;CONTAINS SUBTEST ITERATION COUNT
   (1)  001106  000000                  $LPADR: .WORD   0           ;;CONTAINS SCOPE LOOP ADDRESS
   (1)  001110  000000                  $LPERR: .WORD   0           ;;CONTAINS SCOPE RETURN FOR ERRORS
   (1)  001112  000000                  $ERTTL: .WORD   0           ;;CONTAINS TOTAL ERRORS DETECTED
   (1)  001114     000                  $ITEMB: .BYTE   0           ;;CONTAINS ITEM CONTROL BYTE
   (1)  001115     001                  $ERMAX: .BYTE   1           ;;CONTAINS MAX. ERRORS PER TEST
   (1)  001116  000000                  $ERRPC: .WORD   0           ;;CONTAINS PC OF LAST ERROR INSTRUCTION
   (1)  001120  000000                  $GDADR: .WORD   0           ;;CONTAINS ADDRESS OF 'GOOD' DATA
   (1)  001122  000000                  $BDADR: .WORD   0           ;;CONTAINS ADDRESS OF 'BAD' DATA
   (1)  001124  000000                  $GDDAT: .WORD   0           ;;CONTAINS 'GOOD' DATA
   (1)  001126  000000                  $BDDAT: .WORD   0           ;;CONTAINS 'BAD' DATA
   (1)  001130  000000                          .WORD   0           ;;RESERVED--NOT TO BE USED
   (1)  001132  000000                          .WORD   0
   (1)  001134     000                  $AUTOB: .BYTE   0           ;;AUTOMATIC MODE INDICATOR
   (1)  001135     000                  $INTAG: .BYTE   0           ;;INTERRUPT MODE INDICATOR
   (1)  001136  000000                          .WORD   0
   (1)  001140  177570                  SWR:    .WORD   DSWR        ;;ADDRESS OF SWITCH REGISTER
   (1)  001142  177570                  DISPLAY: .WORD  DDISP       ;;ADDRESS OF DISPLAY REGISTER
   (1)  001144  177560                  $TKS:   177560              ;;TTY KBD STATUS
   (1)  001146  177562                  $TKB:   177562              ;;TTY KBD BUFFER
   (1)  001150  177564                  $TPS:   177564              ;;TTY PRINTER STATUS REG. ADDRESS
   (1)  001152  177566                  $TPB:   177566              ;;TTY PRINTER BUFFER REG. ADDRESS
   (1)  001154     000                  $NULL:  .BYTE   0           ;;CONTAINS NULL CHARACTER FOR FILLS
   (1)  001155     002                  $FILLS: .BYTE   2           ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
   (1)  001156     012                  $FILLC: .BYTE   12          ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
   (1)  001157     000                  $TPFLG: .BYTE   0           ;;'TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
   (1)  001160  000000                  $TIMES: 0                   ;;MAX. NUMBER OF ITERATIONS
   (1)  001162  000000                  $ESCAPE:0                   ;;ESCAPE ON ERROR ADDRESS
   (1)  001164  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
   (1)  001170     077                  $QUES:  .ASCII  /?/         ;;QUESTION MARK
   (1)  001171     015                  $CRLF:  .ASCII  <15>        ;;CARRIAGE RETURN
   (1)  001172  000012                  $LF:    .ASCIZ  <12>        ;;LINE FEED
   (2)                                  ;;*****************************************************************
   (2)                                  .SBTTL  APT MAILBOX-ETABLE
   (2)
   (3)                                  ;;*****************************************************************
   (2)                                  .EVEN
   (2)  001174                          $MAIL:                      ;;APT MAILBOX
   (2)  001174  000000                  $MSGTY: .WORD   AMSGTY      ;;MESSAGE TYPE CODE
   (2)  001176  000000                  $FATAL: .WORD   AFATAL      ;;FATAL ERROR NUMBER
   (2)  001200  000000                  $TESTN: .WORD   ATESTN      ;;TEST NUMBER
   (2)  001202  000000                  $PASS:  .WORD   APASS       ;;PASS COUNT
   (2)  001204  000000                  $DEVCT: .WORD   ADEVCT      ;;DEVICE COUNT
   (2)  001206  000000                  $UNIT:  .WORD   AUNIT       ;;I/O UNIT NUMBER
   (2)  001210  000000                  $MSGAD: .WORD   AMSGAD      ;;MESSAGE ADDRESS
```

```
(2)  001212  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001214                         $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001214      000                $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001215      000                $ENVM:  .BYTE   AENVM   ;;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                $DFVM:  .WORD   ADEVM   ;;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)
(1)  001256                             $ERRTB:
 59
 60
 61
 70                                     ;ITEM   1
 71  001256  013247                             EM1                     ;STATUS REG. ERROR
 72  001260  013367                             DH1                     ;ERRPC STREG EXPECTED ACTUAL
 73  001262  013536                             DT1                     ;$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,,PREAD,$BDDAT
 93  001314  013576                             DF1
```

```
                                .SBTTL      MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
001316  170400                  STREG:  ABASE                   ;ADDRESS OF STATUS REGISTER
001320  170401                  ADST1:  ABASE+1                 ;UPPER BYTE OF STATUS REG.
001322  170402                  ADBUFF: ABASE+2                 ;ADDRESS OF A/D BUFFER
001324  170404                  DACA:   ABASE+4                 ;ADDRESS OF D TO A "A"
001326  170406                  DACB:   ABASE+6                 ;ADDRESS OF D TO A "B"
001330  000400                  VECTOR: AVECT1                  ;VECTOR ADDRESS
001332  000402                  VECTR1: AVECT1+2
001334  000404                  VECTR2: AVECT1+4                ;ERROR VECTOR ADDRESS
001336  000406                  VECTR3: AVECT1+6
001340  170420                  KWCSR:  170420                  ;CLOCK STATUS/CONTROL REGISTER
001342  170422                  KWBPR:  170422                  ;CLOCK PRESET/COUNTER REGISTER
001344  170440                  DAC0:   170440                  ;AAV11-C DAC "A" ADDRESS
001346  170442                  DAC1:   170442                  ;                "B"
001350  170444                  DAC2:   170444                  ;                "C"
001352  170446                  DAC3:   170446                  ;                "D"
001354  000020                  VWRAP:  20
001356  001000                  BARF:   BIT9                    ;DELAY FACTOR
001360  000000                  TEMP:   0                       ;WORK AREA
001362  000000                  CHANL:  0                       ;CHANNEL VALUE
001364  000000                  SPREAD: 0                       ;DEVIATION FROM THE NOMINAL
001366  000000                  TC1:    0       ;NON-ZERO, AXV11-C TEST FIXTURE IS INSTALLED
001370  000000                  TC2:    0       ;NON-ZERO, AAV11-C TO AXV11-C CABLE IN INSTALLED
001372  000000                  ADV11C: 0       ;NON-ZERO, MODULE IS ADV11-C (NO DAC'S ON BOARD)
001374  000000                  KWAD:   0       ;NON-ZERO, CLOCK CONNECTED TO RTC IN
001376  000000                  KWEX:   0       ;NON-ZERO, JUMPER F2 IS INSTALLED AND CLOCK CONNECTED TO EXT TRIG
001400  000000                  MAEX:   0       ;NON-ZERO, JUMPER F2 IS INSTALLED AND MANUAL TRIGGER IS CONNECTED
001402  000000                  BTEX:   0       ;NON-ZERO, JUMPER F1 IS INSTALLED

001404                          UNEXP:
(1) 001404  012737  001420  001162           MOV     #1$,$ESCAPE     ;;ESCAPE TO 1$ ON ERROR
    001412  005237  001103           INC     $ERFLG
    0014:6  104003           ERROR   3
    001420  005037  001162   1$:     CLR     $ESCAPE         ;RETURN ESCAPE TO NORMAL
    001424  000002           RTI                             ;UNEXPECTED INTERRUPT

                                ;SUBROUTINE TO DELAY AN AMOUNT OF CPU TIME

001426  013700  001356          STALL:  MOV     BARF,R0         ;GET DELAY FACTOR
001432  005300                  1$:     DEC     R0              ;DELAY
001434  001376                          BNE     1$
001436  000207                          RTS     PC              ;EXIT
```

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

```
                                        .SBTTL          INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
001522  005037  001360          BEGIN0: CLR     TEMP            ;CLEAR RESTART FLAG
001526  000402                  BEGIN2: BR      BEGST
001530  005237  001360                  INC     TEMP            ;SET RESTART FLAG
001534                          BEGST:
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001534  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001540  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001542  022706  001140                  CMP     #SWR,R6 ;;DONE?
001546  001374                          BNE     .-6             ;;LOOP BACK IF NO
001550  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
001554  012737  015416  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001562  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001570  012737  015676  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001576  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001604  012737  017562  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001612  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001620  012737  015240  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001626  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001634  013737  010342  010334          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
001642  005037  001160                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001646  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001652  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
001660  012737  001660  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001666  012737  001666  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001674  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001700  012737  001734  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001706  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001714  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001722  022777  177777  177210          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001730  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001732  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
001734  012716  001742          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001740  000002                          RTI
001742  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001750  012737  000174  001142          MOV     #DISPREG,DISPLAY
001756  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

001762  005037  001202                  CLR     $PASS           ;;CLEAR PASS COUNT
001766  132737  000200  001215          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
001774  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
001776  012737  001216  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
002004                          67$:
002004  012737  000300  000022          MOV     #300,@#IOTVEC+2 ;KXT11
002012  012737  000300  000032          MOV     #300,@#EMTVEC+2 ;  FIX
002020  012737  000300  000036          MOV     #300,@#TRAPVEC+2 ;   FOR LOWER
002026  012737  000300  000026          MOV     #300,@#PWRVEC+2 ;       PWS LEVELS
002034  012737  005046  016232          MOV     #5046,$TYPE     ;A WAY TO LOWER
002042  012737  012746  016234          MOV     #12746,$TYPE+2  ; PS FOR
002050  012737  016244  016236          MOV     #$TYPE+12,$TYPE+4
002056  012737  000002  016240          MOV     #RTI,$TYPE+6    ;   TTY OUTPUT
```

```
   171  002064  004737  013646                  JSR     PC,$TKINT       ;INIT THE CONSOLE VECTORS
```

```
                                        .SBTTL      DIALOGUE TO DETERMINE WHICH TEST TO RUN
                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002070  005227  177777                          INC     #-1                 ;;FIRST TIME?
002074  001053                                BNE     68$                 ;;BRANCH IF NO
002076  022737  010374  000042                CMP     #$ENDAD,@#42        ;;ACT-11?
002104  001447                                BEQ     68$                 ;;BRANCH IF YES
002106  104401  002154                        TYPE    ,69$                ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002112  005737  000042                        TST     @#42                ;;ARE WE RUNNING UNDER XXDP/ACT?
002116  001012                                BNE     70$                 ;;BRANCH IF YES
002120  123727  001214  000001                CMPB    $ENV,#1             ;;ARE WE RUNNING UNDER APT?
002126  001406                                BEQ     70$                 ;;BRANCH IF YES
002130  023727  001140  000176                CMP     SWR,#SWREG          ;;SOFTWARE SWITCH REG SELECTED?
002136  001005                                BNE     71$                 ;;BRANCH IF NO
002140  104407                                GTSWR                       ;;GET SOFT-SWR SETTINGS
002142  000403                                BR      71$
002144  112737  000001  001134  70$:          MOVB    #1,$AUTOB           ;;SET AUTO-MODE INDICATOR
002152                                  71$:
002152  000424                                BR      68$                 ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <CRLF># CVAXAB  AXV11-C/ADV11-C DIAGNOSTIC  #<CRLF>
002224                                  68$:
002224  004737  007540                        JSR     PC,FIXONE           ;INITIALIZE ADDRESSES
002230  005737  001360                77$:    TST     TEMP                ;ARE WE RESTARTING THE PROGRAM
002234  001062                                BNE     40$                 ;BR IF YES
002236  005737  001134                        TST     $AUTOB              ;IS IT CHAINED?
002242  001402                                BEQ     1$
002244  000137  007412                        JMP     BEGIND  ;RUN ONLY THE LOGIC TEST AND SELECTED WRAPAROUND IF APT/XXDP CHA
002250  004537  001456                1$:     JSR     R5,ASKTA            ;ASK OPERATOR ABOUT DIFFERENT CONFIG.
002254  011537                                MSKWAD                      ;IS KWV11-C CONNECTED TO CLOCK START
002256  001374                                KWAD                        ;
002260  000240                                NOP
002262  005037  001400                        CLR     MAEX                ;ENSURE CLEARED FLAG
002266  004537  001456                        JSR     R5,ASKTA            ;ASK IF KWV11-C CONNECTED TO EXT. START
002272  011621                                MSKWEX
002274  001376                                KWEX
002276  000403                                BR      2$
002300  000415                                BR      4$                  ;IF ANSWER WAS YES, BYPASS NEXT QUESTION
002302  005037  001402                        CLR     BTEX                ;ENSURE CLEARED FLAG
002306  004537  001456          2$:           JSR     R5,ASKTA            ;ASK IF MANUAL TRIGGER IS CONNECTED TO EXT. START
002312  011730                                MSMAEX
002314  001400                                MAEX
002316  000401                                BR      3$
002320  000405                                BR      4$
002322  004537  001456          3$:           JSR     R5,ASKTA            ;ASK IF B EVENT IS CONNECTED TO EXT TRIG
002326  012106                                MSBTEX
002330  001402                                BTEX
002332  000240                                NOP
002334  004537  001456          4$:           JSR     R5,ASKTA            ;ASK IF MODULE IS ADV11-C
002340  012201                                MSADV
002342  001372                                ADV11C
002344  000240                                NOP
002346  004537  001456          10$:          JSR     R5,ASKTA            ;ASK IF TEST FIXTURE #1 IS INSTALLED
002352  012230                                MSTC1
002354  001366                                TC1
002356  000240                                NOP
```

```
                                11$:    JSR     R5,ASKTA        ;ASK IF TEST CONNECTOR #2 IS INSTALLED
                                        MSTC2
                                        TC2
                                        NOP
                                12$:    NOP
                                20$:    NOP
                                30$:    TYPE,   MSG70           ;TELL THE OPERATOR THE TESTS AVAILABLE
                                40$:    TYPE    ,MSG71
                                ;ROUTINE TO ASK OPERATOR WHAT SUB-SECTION TO EXECUTE
                                TRYAG:  RDLIN
                                        BIS     #100,@$TKS
                                        CLR     -(SP)           ;CLEAR PSW
                                        MOV     #1$,-(SP)
                                        RTI
                                1$:     MOV     (SP)+,R0        ;READ ANSWER
                                        MOV     (R0),R0         ;GET THE 1ST CHARACTER
                                        BIC     #177600,R0      ;REMOVE EXTRA BITS
                                        MOV     #OKCHAR,R1      ;LOAD POINTER TO GOOD CHARACTER LIST
                                2$:     CMP     R0,(R1)+        ;CHECK IF VALID CHARACTER
                                        BNE     3$              ;BR IF NOT
                                        MOV     (R1),R1         ;GET THE ADDRESS
                                        JMP     @R1             ;DO THE SELECTED SUB-TEST
                                3$:     TST     (R1)+           ;BUMP THE POINTER
                                        BNE     2$              ;BR IF MORE CHARACTERS
                                6$:     TYPE    ,QUEST
                                        BR      TRYAG           ;WAIT FOR CHARACTER

                                ;TABLE OF VALID MENU CHARACTERS AND STARTING ADDRESS
                                OKCHAR: 141                     ;LOWER CASE "A"
                                        BEGINA
                                        154                     ;LOWER CASE "L"
                                        BEGINL
                                        167                     ;LOWER CASE "W"
                                        BEGINW
                                        'A
                                        BEGINA
                                        'L
                                        BEGINL
                                        'W
                                        BEGINW
                                        '1      ,IOTST1
                                        '2      ,IOTST2
                                        '3,     IOTST3
                                        '4,     IOTST4
                                        '5,     IOTST5
                                        '6,     IOTST6
                                        '7,     IOTST7
                                        0,0,0,0
```

```
                                BEGL:
                                ;;*****************************************************************
                                ;*TEST 1         ADDRESS THE 4 BUS ADDRESSES OF THE AXV11-C
                                ;;*****************************************************************
(2) 002560 012737 002560 001106 TST1:   MOV     #TST1,$LPADR
265 002566 012737 000001 001102         MOV     #$TN-1,$TSTNM   ;LOAD TEST NUMBER
266 002574 00?777 176516                TST     @STREG          ;ADDRESS A/D STATUS REGISTER
267 002600 00?777 176516                TST     @ADBUFF         ;ADDRESS A/D DATA BUFFER
268 002604 005777 176514                TST     @DACA           ;ADDRESS D TO A "A"
269 002610 005777 176512                TST     @DACB           ;ADDRESS D TO A "B"
                                ;;*****************************************************************
                                ;*TEST 2         FLOAT A ONE THRU MULTIPLEXER (BITS 11-8)
                                ;;*****************************************************************
(2) 002614 000004               TST2:   SCOPE
271 002616 012737 000400 001124         MOV     #BIT8,$GDDAT    ;LOAD FIRST BIT
272 002624 104415               2$:     CHKIT
273 002626 104001                       ERROR   1               ;FAILED TO LOAD + READ BIT
274 002630 006337 001124        1$:     ASL     $GDDAT          ;GET NEXT BIT
275 002634 023727 001124 010000         CMP     $GDDAT,#BIT12   ;FINISHED?
276 002642 001370                       BNE     2$              ;;NO,GO TO NEXT TEST
277
278                             ;;*****************************************************************
                                ;*TEST 3         LOAD AND READ BACK ERROR I.E. BIT14
                                ;;*****************************************************************
(2) 002644 000004               TST3:   SCOPE
279 002646 012737 040000 001124         MOV     #BIT14,$GDDAT
280 002654 104415                       CHKIT
281 002656 104001                       ERROR   1               ;FAILED TO LOAD + READ ERROR I.E.
282                             ;;*****************************************************************
                                ;*TEST 4         LOAD AND READ BACK INTERRUPT ENABLE BIT6
                                ;;*****************************************************************
(2) 002660 000004               TST4:   SCOPE
283 002662 012777 001404 176440         MOV     #UNEXP,@VECTOR  ;SETUP FOR UNEXPECTED INTERUPT
284 002670 012737 000100 001124         MOV     #BIT6,$GDDAT    ;LOAD EXPECTED DATA
285 002676 104415                       CHKIT
286 002700 104001                       ERROR   1               ;FAILED TO LOAD + READ INTERRUPT ENABLE
287
288                             ;;*****************************************************************
                                ;*TEST 5         LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BIT5
                                ;;*****************************************************************
(2) 002702 000004               TST5:   SCOPE
289 002704 012737 000040 001124         MOV     #BIT5,$GDDAT    ;LOAD EXPECTED DATA
290 002712 104415                       CHKIT
291 002714 104001                       ERROR   1               ;FAILED TO LOAD + READ CLOCK OVERFLOW START ENABLE
292                             ;;*****************************************************************
                                ;*TEST 6         LOAD AND READ BACK EXTERNAL START ENABLE BIT4
                                ;;*****************************************************************
(2) 002716 000004               TST6:   SCOPE
293 002720 012737 000020 001124         MOV     #BIT4,$GDDAT    ;LOAD EXPECTED DATA
294 002726 104415                       CHKIT
295 002730 104001                       ERROR   1               ;FAILED TO LOAD + READ EXT. START ENABLE
```

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

```
                                                        ;  OR BITO FAILED TO CLEAR
003132  017700  176164                MOV     @ADBUFF,R0      ;CLEAR DONE FLAG FOR ITERATIONS

                              ;;*****************************************************************
                              ;*TEST 15       TEST INIT CLEARS DONE FLAG
                              ;;*****************************************************************
                              TST15:  SCOPE
003136  000004
003140  012737  000300  001160          MOV     #300,$TIMES     ;;DO 300 ITERATIONS
003146  005037  001124                CLR     $GDDAT          ;CLEAR EXPECTED
003152  005277  176140                INC     @STREG          ;START CONVERSION
003156  105777  176134          2$:   TSTB    @STREG
003162  100375                      BPL     2$
003164  000005                      RESET
003166  104414                      CHECK
003170  104001                      ERROR   1               ;DONE FLAG FAILED TO CLEAR
003172  052777  000100  175744      BIS     #100,@$TKS      ;SET INTRPT. EN. BIT

                              ;;*****************************************************************
                              ;*TEST 16       TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
                              ;;*****************************************************************
                              TST16:  SCOPE
003200  000004
003202  005277  176110                INC     @STREG          ;SET A/D START CONVERSION BIT
003206  105777  176104          1$:   TSTB    @STREG          ;WAIT FOR FLAG
003212  100375                      BPL     1$
003214  017700  176102                MOV     @ADBUFF,R0      ;READ CONVERTED VALUE
003220  104414                      CHECK
003222  104001                      ERROR   1               ;DONE FLAG FAILED TO CLEAR
```

```
                                ;;*****************************************************************
                                ;*TEST 17       GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
                                ;;*****************************************************************
003224  000004                  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''.
003226  012700  000017                  MOV     #17,R0          ;GET TEST NO.
003232  004737  010144                  JSR     PC,DUMW         ;PRINT MESSAGE
003236  005046                          CLR     -(SP)           ;RESET PRIORITY
003240  012746  003246                  MOV     #3$,-(SP)
003244  000002                          RTI
003246  012777  003322  176054  3$:     MOV     #1$,@VECTOR     ;INTERRUPT VECTOR ADDRESS
003254  012777  000200  176050          MOV     #200,@VECTR1    ;SET UP NEW PSW
003262  012777  000101  176026          MOV     #BIT6!BIT0,@STREG ;SET INTERRUPT ENABLE BIT + START CONVERSION
003270  105777  176022          2$:     TSTB    @STREG          ;WAIT FOR DONE
003274  100375                          BPL     2$              ;FLAG TO SET
003276  017737  176014  001126          MOV     @STREG,$BDDAT   ;READ STATUS REGISTER
003304  012737  000300  001124          MOV     #BIT7!BIT6,$GDDAT ;GOOD DATA
003312  104002                          ERROR   2               ;FAILED TO INTERRUPT ON DONE
003314  004737  010216                  JSR     PC,DUMC         ;TYPE COMPLETED
003320  000414                          BR      TST20           ;;BRANCH TO NEXT TEST
003322  022626                  1$:     CMP     (SP)+,(SP)+     ;RESET STACK POINTER
003324  012777  001404  175776          MOV     #UNEXP,@VECTOR  ;SET UP FOR UNEXPECTED INTERRUPT
003332  005046                          CLR     -(SP)           ;CLEAR PSW
003334  012746  003342                  MOV     #4$,-(SP)
003340  000002                          RTI
003342  004737  010216          4$:     JSR     PC,DUMC         ;TYPE COMPLETED
003346  005777  175750                  TST     @ADBUFF         ;CLEAR DONE BIT
                                ;;*****************************************************************
                                ;*TEST 20       TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET
                                ;;*****************************************************************
003352  000004                  TST20:  SCOPE
                                        ;* 'ENTERING TEST 20'' TYPED OUT TO TELL YOU THE NEXT
                                        ;*TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
                                        ;*THERE IS DANGER THAT THE 'Q BUSS'' COULD GET 'HUNG'' WHILE
                                        ;*EXECUTING TEST ''20''.
003354  012700  000020                  MOV     #20,R0          ;GET TEST NO.
003360  004737  010144                  JSR     PC,DUMW         ;PRINT MESSAGE
003364  012777  003424  175742          MOV     #1$,@VECTR2     ;SETUP VECTOR ADDRESS
003372  012777  140000  175716          MOV     #BIT15!BIT14,@STREG     ;CAUSE AN INTERRUPT
003400  017737  175712  001126          MOV     @STREG,$BDDAT   ;BAD DATA
003406  012737  140000  001124          MOV     #BIT15!BIT14,$GDDA      ;GOOD DATA
003414  104002                          ERROR   2
003416  004737  010216                  JSR     PC,DUMC         ;TYPE COMPLETED
003422  000753                          BR      TST20
003424  022626                  1$:     CMP     (SP)+,(SP)+     ;POP STACK
003426  004737  010216                  JSR     PC,DUMC
003432  005077  175660                  CLR     @STREG
```

```
                                ;;*****************************************************************
                                ;*TEST 21        TEST ERROR FLAG SETS IF 2ND CONVERSION IS STARTED WHILE A/D DONE IS SET
                                ;;*****************************************************************
003436  000004                  TST21:  SCOPE
003440  012777  000001  175650          MOV     #BIT0,@STREG    ;START CONVERSION
003446  105777  175644          1$:     TSTB    @STREG          ;WAIT FOR
003452  100375                          BPL     1$
003454  012737  100200  001124          MOV     #BIT15!BIT7,$GDDAT ;LOAD EXPECTED VALUE
003462  012777  000001  175626          MOV     #BIT0,@STREG    ;START 2ND CONVERSION
003470  104414                          CHECK
003472  104001                          ERROR   1               ;ERROR FLAG NOT SET WHEN 2ND
                                        ; CONVERSION WAS STARTED BEFORE READING BUFFER FROM FIRST
003474  017700  175622                  MOV     @ADBUFF,R0      ;CLEAR DONE FLAG
003500  005077  175612                  CLR     @STREG          ;CLEAR A/D CONTROL

                                ;;*****************************************************************
                                ;*TEST 22        TEST CLOCK OVERFLOW STARTS A/D  (IF KWV11-C IS AVAILABLE)
                                ;;*****************************************************************
003504  000004                  TST22:  SCOPE
003506  005737  001374                  TST     KWAD            ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
003512  001424                          BEQ     TST23           ;;BR IF NO CLOCK THERE
003514  012737  000240  001124          MOV     #BIT7!BIT5,$GDDAT       ;LOAD EXPECTED A/D STATUS
003522  113777  001124  175566          MOVB    $GDDAT,@STREG   ;ENABLE THE A/D STATUS REGISTER
003530  012777  177776  175604          MOV     #177776,@KWBPR  ;LOAD KWV11-C CLOCK PRESET REGISTER
003536  012777  000011  175574          MOV     #11,@KWCSR      ;START CLOCK
003544  004737  001426                  JSR     PC,STALL        ;DELAY FOR A CLOCK TICK
003550  104414                          CHECK                   ;CHECK A/D STATUS AGAINST EXPECTED
003552  104001                          ERROR   1               ;A/D DONE FAILED TO SET WITH CLOCK STARTS
003554  005777  175542                  TST     @ADBUFF         ;CLEAR A/D DONE
003560  005077  175532                  CLR     @STREG          ;CLEAR A/D CONTROL

                                ;;*****************************************************************
                                ;*TEST 23        TEST EXTERNAL TRIGGER STARTS A/D (IF KW11-C IS CONNECTED TO EXT START TA
                                ;;*****************************************************************
003564  000004                  TST23:  SCOPE
003566  005737  001376                  TST     KWEX            ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
003572  001424                          BEQ     TST24           ;;BR IF NO CLOCK THERE
003574  012737  000220  001124          MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPECTED A/D STATUS
003602  113777  001124  175506          MOVB    $GDDAT,@STREG   ;ENABLE THE A/D STATUS REGISTER
003610  012777  177776  175524          MOV     #177776,@KWBPR  ;LOAD KWV11-C CLOCK PRESET REGISTER
003616  012777  000011  175514          MOV     #11,@KWCSR      ;START CLOCK
003624  004737  001426                  JSR     PC,STALL        ;DELAY FOR CLOCK TICKS
003630  104414                          CHECK                   ;CHECK A/D STATUS AGAINST EXPECTED
003632  104001                          ERROR   1               ;A/D DONE FAILED TO SET WITH EXTERNAL STARTS
003634  005777  175462                  TST     @ADBUFF         ;CLEAR A/D DONE
003640  005077  175452                  CLR     @STREG          ;CLEAR A/D CONTROL
```

```
                                ;;*****************************************************************
                                ;*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)+,R0        ;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!BIT5,$GDDAT ;LOAD EXPECTED
                                        MOV     #-2,@KWBPR      ;LOAD CLOCK PRESET
                                        MOVB    #BIT5,@STREG    ;ENABLE CLOCK START
                                        MOV     @ADBUFF,R0      ;ENSURE CLEARED A/D DONE
                                        MOV     #11,@KWCSR      ;START CLOCK
                                1$:     TSTB    @KWCSR          ;WAIT FOR CLOCK READY
                                        BPL     1$
                                        BISB    #BIT0,@STREG    ;CLOCK OVERFLOW SHOULD HAVE STARTED A/D
                                                                ;TRY TO ST,  IT AGAIN AND GET AN ERROR
                                        MOV     @STREG,$BDDAT   ;READ A/D STATUS
                                        BMI     2$              ;;BR IF ERROR BIT SET
                                        ERROR   1               ;ERROR FLAG NOT SET WHEN 2ND CONVERT STARTED
                                                                ; WHILE FIRST IS IN PROGRESS
                                2$:     MOV     @ADBUFF,R0      ;READ AND CLEAR A/D DONE
                                        CLR     @STREG          ;CLEAR STATUS REG
```

```
                                ;;*************************************************************
                                ;*TEST 26       TEST "B EVENT" STARTS A/D (IF JUMPER " 1" IS PRESENT)
                                ;;*************************************************************
004034  000004                  TST26:  SCOPE
004036  005737  001402                  TST     BTEX            ;TEST IF OPERATOR SAID "F1" IS INSTALLED
004042  001416                          BEQ     TST27           ;;BR IF NOT THERE
004044  012737  000220  001124          MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPECTED A/D STATUS
004052  013777  001124  175236          MOV     $GDDAT,@STREG   ;ENABLE THE A/D STATUS REGISTER
004060  004737  001426                  JSR     PC,STALL        ;DELAY AN AMOUNT OF TIME
004064  104414                          CHECK                   ;CHECK A/D STATUS AGAINST EXPECTED
004066  104001                          ERROR   1               ;A/D DONE FAILED TO SET WITH "B EVENT"
004070  005077  175222                  CLR     @STREG          ;CLEAR A/D CONTROL
004074  005777  175222                  TST     @ADBUFF         ;CLEAR A/D DONE


                                ;;*************************************************************
                                ;*TEST 27       END OF ADV11-C LOGIC TESTS
                                ;;*************************************************************
004100  000004                  TST27:  SCOPE
004102  000207                          RTS     PC              ;RETURN TO TEST SECTION


                                        .SBTTL
                                        .SBTTL  END OF LOGIC TESTS - SECTION


                                ;;SUBROUTINE FOR LOGIC TESTS;;
004104  013777  001124  175204  TESTIT: MOV     $GDDAT,@STREG   ;LOAD EXPECTED VALUE
004112  017737  175200  001126  TEST:   MOV     @STREG,$BDDAT   ;READ ST. REG.
004120  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;COMPARE RESULTS
004126  001002                          BNE     RETERR          ;;ERROR RETURN
004130  062716  000002                  ADD     #2,(SP)         ;BUMP RETURN ADDRESS TO GET AROUND ERROR
004134  000002                  RETERR: RTI


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

```
 497  004136                                  WRAP:
 (4)                                          ;;*****************************************************************
 (3)                                          ;*TEST 30       SETUP TO RUN ANALOG WRAPAROUND TEST
 (3)                                          ;;*****************************************************************
 (2)  004136  012737  000030  001102  TST30:  MOV     #$TN,$TSTNM
 (1)  004144  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
 498                                  ;LOAD AXV11-C DAC TO MAX OUTPUT VOLTAGE
 499  004152  012777  007777  175144          MOV     #7777,@DACA     ;LOAD DAC "A"
 500  004160  012777  007777  175140          MOV     #7777,@DACB     ;LOAD DAC "B"
 501  004166  012737  004210  001110          MOV     #1$,$LPERR      ;LOAD ERROR ADDRESS
 502  004174  012737  004210  001106          MOV     #1$,$LPADR      ;LOAD LOOP ADDRESS
 503                                  ;DELAY SUFFICIENT TIME TO LET THE DAC'S SETTLE
 504  004202  012700  000002                  MOV     #2,R0           ;LOAD DELAY TIMER
 505  004206  005001                          CLR     R1              ;CLEAR DELAY COUNT
 506  004210  005301                  1$:     DEC     R1              ;DELAY
 507  004212  001376                          BNE     1$
 508  004214  005300                          DEC     R0              ;DELAY
 509  004216  001374                          BNE     1$
 510
 511                                          ;;*****************************************************************
 (3)                                          ;*TEST 31       COMPARE CHANNEL 0 (F.S.) AGAINST 1 (1/2 FS), 2 (1/4 FS), 3 (1/8)
 (3)                                          ;;*****************************************************************
 (2)  004220  000004                  TST31:  SCOPE
 (1)  004222  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
 512  004230  005737  001366          1$:     TST     TC1             ;TEST IF TEST FIXTURE IS INSTALLED
 513  004234  001440                          BEQ     TST32           ;;BR IF NOT
 514  004236  004537  007742                  JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 515  004242  000000                          CHAN00                  ;CHANNEL 0
 516  004244  004537  010100                  JSR     R5,COMPAR       ;COMPARE RESULTS
 517  004250  007777                          7777
 518  004252  001354                          VWRAP
 519  004254  104004                          ERROR   4               ;ERROR AN A/D CHANNEL 0 - VALUE DID NOT
 520                                                                  ;   EQUAL EXPECTED VALUE
 521  004256  004537  007742                  JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 522  004262  000001                          CHAN01                  ;CHANNEL 1
 523  004264  004537  010100                  JSR     R5,COMPAR       ;COMPARE RESULTS
 524  004270  006000                          6000                    ;EXPECTED VALUE
 525  004272  001354                          VWRAP                   ;USING A KNOWN SPREAD
 526  004274  104004                          ERROR   4               ;ERROR ON A/D CHANNEL 1 - VALUE DID NOT
 527                                                                  ;  EQUAL EXPECTED
 528  004276  004537  007742                  JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 529  004302  000002                          CHAN02                  ;CHANNEL 2
 530  004304  004537  010100                  JSR     R5,COMPAR       ;COMPARE RESULTS
 531  004310  005000                          5000                    ;AGAINST THIS VALUE FOR CHANNEL 2
 532  004312  001354                          VWRAP                   ;USING A KNOWN SPREAD
 533  004314  104004                          ERROR   4               ;ERROR ON A/D CHANNEL 2 - VALUE DID NOT
 534                                                                  ;  EQUAL EXPECTED
 535  004316  004537  007742                  JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 536  004322  000003                          CHAN03                  ;CHANNEL 03
 537  004324  004537  010100                  JSR     R5,COMPAR       ;COMPARE RESULTS
 538  004330  004400                          4400                    ;AGAINST THIS VALUE FOR CHANNEL 3
 539  004332  001354                          VWRAP                   ;USING A KNOWN SPREAD
 540  004334  104004                          ERROR   4               ;ERROR ON A/D CHANNEL 3 - VALUE DID NOT
 541                                                                  ;  EQUAL EXPECTED
```

```
543
(3)                                     ;;*****************************************************************
(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
```

```
                                ;;*****************************************************************
                                ;*TEST 33       COMPARE CHANNEL 1 (1/2 F.S.) AGAINST OTHER 1/2 F.S. CHANNELS (5 AND 11)
                                ;;*****************************************************************
004436  000004                TST33:  SCOPE
004440  012737  000001  001160        MOV     #1,$TIMES       ;;DO 1 ITERATION
004446  005737  001366                TST     TC1             ;TEST IF TEST FIXTURE IS INSTALLED
004452  001431                        BEQ     TST34           ;;BR IF NOT
004454  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004460  000001                        CHAN01                  ;CHANNEL 1
004462  013737  001360  004510        MOV     TEMP,4$         ;SAVE CHANNEL 1 CONVERTED VALUE
004470  013737  001360  004530        MOV     TEMP,10$        ;SAVE IT AGAIN

004476  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004502  000005                        CHAN05                  ;CHANNEL 5
004504  004537  010100                JSR     R5,COMPAR       ;COMPARE RESULTS
004510  000000                4$:     0                       ;AGAINST THIS VALUE FOR CHANNEL 1
004512  010270                        V2                      ;USING A SPREAD OF 2 COUNTS
004514  104004                        ERROR   4               ;ERROR ON A/D CHANNEL 5 - VALUE DID NOT
                                                              ;  EQUAL VALUE OF CHANNEL 0

004516  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004522  000011                        CHAN11                  ;CHANNEL 11
004524  004537  010100                JSR     R5,COMPAR       ;COMPARE RESULTS
004530  000000                10$:    0                       ;AGAINST THIS VALUE FOR CHANNEL 1
004532  010270                        V2                      ;USING A SPREAD OF 2 COUNTS
004534  104004                        ERROR   4               ;ERROR ON A/D CHANNEL 11 - VALUE DID NOT
                                                              ;  EQUAL VALUE OF CHANNEL 1
```

```
                                ;;*****************************************************************
                                ;*TEST 34       COMPARE CHANNEL 2 (1/4 F.S.) AGAINST OTHER 1/4 F.S. CHANNELS (6 AND 12)
                                ;;*****************************************************************
004536  000004                TST34:  SCOPE
004540  012737  000001  001160        MOV     #1,$TIMES       ;;DO 1 ITERATION
004546  005737  001366                TST     TC1             ;TEST IF TEST FIXTURE IS INSTALLED
004552  001431                        BEQ     TST35           ;;BR IF NOT
004554  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004560  000002                        CHAN02                  ;CHANNEL 2
004562  013737  001360  004610        MOV     TEMP,4$         ;SAVE CHANNEL 2 CONVERTED VALUE
004570  013737  001360  004630        MOV     TEMP,10$        ;SAVE IT AGAIN

004576  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004602  000006                        CHAN06                  ;CHANNEL 6
004604  004537  010100                JSR     R5,COMPAR       ;COMPARE RESULTS
004610  000000                4$:     0                       ;AGAINST THIS VALUE FOR CHANNEL 2D
004612  010270                        V2                      ;USING A SPREAD OF 2 COUNTS
004614  104004                        ERROR   4               ;ERROR ON A/D CHANNEL 6 - VALUE DID NOT
                                                              ;  EQUAL VALUE OF CHANNEL 2

004616  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004622  000012                        CHAN12                  ;CHANNEL 12
004624  004537  010100                JSR     R5,COMPAR       ;COMPARE RESULTS
004630  000000                10$:    0                       ;AGAINST THIS VALUE FOR CHANNEL 2
004632  010270                        V2                      ;USING A SPREAD OF 2 COUNTS
004634  104004                        ERROR   4               ;ERROR ON A/D CHANNEL 12 - VALUE DID NOT
                                                              ;  EQUAL VALUE OF CHANNEL 2

                                ;;*****************************************************************
                                ;*TEST 35       COMPARE CHANNEL 3 (1/8 F.S.) AGAINST CHANNEL 7 (1/8 F.S.)
                                ;;*****************************************************************
004636  000004                TST35:  SCOPE
004640  012737  000001  001160        MOV     #1,$TIMES       ;;DO 1 ITERATION
004646  005737  001366                TST     TC1             ;TEST IF TEST FIXTURE IS INSTALLED
004652  001416                        BEQ     TST36           ;;BR IF NOT
004654  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004660  000003                        CHAN03                  ;CHANNEL 3
004662  013737  001360  004702        MOV     TEMP,4$         ;SAVE CHANNEL 3 CONVERTED VALUE

004670  004537  007742                JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
004674  000007                        CHAN07                  ;CHANNEL 7
004676  004537  010100                JSR     R5,COMPAR       ;COMPARE RESULTS
004702  000000                4$:     0                       ;AGAINST THIS VALUE FOR CHANNEL 3
004704  010270                        V2                      ;USING A SPREAD OF 2 COUNTS
004706  104004                        ERROR   4               ;ERROR ON A/D CHANNEL 7 - VALUE DID NOT
                                                              ;  EQUAL VALUE OF CHANNEL 3
```

```
                                ;;*****************************************************************
                                ;*TEST 36       RELATIVE GAIN TEST USING CHANNEL 3 (1/8 F.S.)
                                ;;*****************************************************************
                                TST36:  SCOPE
004710  000004                  
004712  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
004720  005737  001366                  TST     TC1             ;TEST IF AXV11 OR ADV11 CONNECTOR INSTALLED
004724  001454                          BEQ     TST37           ;;BR IF NO CONNECTOR
004726  012737  000000  010076          MOV     #GAIN00,OTHER   ;SELECT GAIN OF 00
004734  004537  007746                  JSR     R5,CONVTR       ;GET THE VALUE OF CHANNEL 03
004740  000003                          CHAN03
004742  004537  010100                  JSR     R5,COMPAR       ;TEST GAIN
004746  004400                          4400                    ;EXPECTED VALUE
004750  001354                          VWRAP                   ; USING KNOWN SPREAD
004752  104004                          ERROR   4               ;GAIN SELECT OF 00 FAILED TO EQUAL EXPECTED VALUE

004754  012737  000004  010076          MOV     #GAIN01,OTHER   ;SELECT GAIN OF 01
004762  004537  007746                  JSR     R5,CONVTR       ;GET THE VALUE OF CHANNEL 03
004766  000003                          CHAN03
004770  004537  010100                  JSR     R5,COMPAR       ;TEST GAIN 01
004774  005000                          5000                    ;EXPECTED VALUE
004776  001354                          VWRAP                   ; USING KNOWN SPREAD
005000  104004                          ERROR   4               ;GAIN SELECT OF 01 FAILED TO INCREASE
                                                                ;  CONVERTED VALUE CORRECTLY
005002  012737  000010  010076          MOV     #GAIN10,OTHER   ;SET GAIN SELECT = 10
005010  004537  007746                  JSR     R5,CONVTR       ;GET VALUE OF CHANNEL 03
005014  000003                          CHAN03
005016  004537  010100                  JSR     R5,COMPAR       ;TEST GAIN 10 VALUE AGAINST 01
005022  006000                          6000                    ;EXPECTED VALUE
005024  001354                          VWRAP                   ; USING KNOWN SPREAD
005026  104004                          ERROR   4               ;GAIN SELECT OF 10 FAILED TO INCREASE
                                                                ;  CONVERTED VALUE CORRECTLY
005030  012737  000014  010076          MOV     #GAIN11,OTHER   ;SET GAIN SELECT = 11
005036  004537  007746                  JSR     R5,CONVTR       ;GET VALUE OF CHANNEL 03
005042  000003                          CHAN03
005044  004537  010100                  JSR     R5,COMPAR       ;TEST GAIN 11 VALUE AGAINST 10
005050  007777                          7777                    ;EXPECTED VALUE
005052  001354                          VWRAP                   ;  USING KNOWN SPREAD
005054  104004                          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
005056  000004                  
005060  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
005066  012777  004000  174232          MOV     #4000,@DACB     ;SET DAC "B" TO MIDRANGE
005074  005737  001372                  TST     ADV11C          ;TEST IF ADV11-C
005100  001410                          BEQ     TST40           ;;BR IF NOT ADV11-C
005102  004537  007742                  JSR     R5,CONVRT       ;GET THE CONVERTED VALUE FOR CH13
005106  000013                          CHAN13
005110  004537  010100                  JSR     R5,COMPAR       ;TEST CH13 AGAINST EXPECTED
005114  007777                          7777                    ;+ F.S.
005116  001354                          VWRAP
005120  104004                          ERROR   4               ;CH13 WAS NOT PULLED UP TO +F.S.
```

```
                                        .SBTTL
                                        .SBTTL  END OF ADV11-C ANALOG WRAPAROUND SECTION
                                        .SBTTL
                                        .SBTTL  START OF AXV11-C ANALOG WRAPAROUND SECTION
                                        .SBTTL

                                ;;******************************************************************
                                ;*TEST 40        AXV11-C ANALOG WRAPAROUND TEST (DAC "A" TO A/D CHAN 0)
                                ;;******************************************************************
(2) 005122  000004              TST40:  SCOPE
(1) 005124  012737  000001  001160      MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        ;AXV11-C DAC "A" CONNECTED TO AXV11-C A/D CHANNEL 0
                                        ;AXV11-C TEST FIXTURE IS REQUIRED

005132  005737  001366                  TST     TC1             ;TEST IF AXV11-C TEST FIXTURE IS PRESENT
005136  001445                          BEQ     TST41           ;;BR IF NO TEST FIXTURE
005140  005737  001372                  TST     ADV11C          ;TEST IF THE MODULE IS A ADV11-C
005144  001042                          BNE     TST41           ;;BR IF NO DAC'S PRESENT
005146  012737  000000  005206          MOV     #0,2$           ;PRIME THE DAC OUTPUT VALUE
005154  013777  005206  174142          MOV     2$,@DACA        ;PRIME THE DAC OUTPUT STAGE
005162  012777  000000  174126          MOV     #0,@STREG       ;INITIILIZE THE A/D STATUS REG
005170  017700  174126                  MOV     @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG
005174  004537  007742          1$:     JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 0
005200  000000                          CHAN00
005202  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED D/A VALUE
005206  000000          2$:     0                               ;EXPECTED
005210  001354                          VWRAP                   ;SPREAD ALLOWED
005212  000413                          BR      3$              ;CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
005214  062737  000010  005206          ADD     #10,2$          ;UPDATE THE D/A OUTPUT VALUE
005222  013777  005206  174074          MOV     2$,@DACA        ;UPDATE THE D/A OUTPUT VOLTAGE
005230  022737  010000  005206          CMP     #10000,2$       ;TEST IF LAST STEP
005236  001356                          BNE     1$
005240  000401                          BR      4$              ;;BR TO NEXT TEST
005242  104004          3$:     ERROR   4                       ;CONVERTED A/D VALUE DID NOT EQUAL EXPECTED VALUE
005244  012777  007777  174052  4$:     MOV     #7777,@DACA     ;LOAD DAC "A" TO +F.S.

```

```
                                        ;;*****************************************************************
                                        ;*TEST 41        AXV11-C ANALOG WRAPAROUND TEST (DAC "B" TO A/D CHAN 13)
                                        ;;*****************************************************************
005252  000004                  TST41:  SCOPE
005254  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        ;AXV11-C DAC "B" CONNECTED TO AXV11-C A/D CHANNEL 13
                                        ;AXV11-C TEST CABLE IS REQUIRED

005262  005737  001366                  TST     TC1             ;TEST IF AXV11-C TEST FIXTURE IS PRESENT
005266  001445                          BEQ     TST42           ;;BR IF NO TEST FIXTURE
005270  005737  001372                  TST     ADV11C          ;TEST IF MODULE IS AN ADV11-C
005274  001042                          BNE     TST42           ;;BR IF NO DAC'A PRESENT
005276  012737  000000  005336          MOV     #0,2$           ;PRIME THE DAC OUTPUT VALUE
005304  013777  005336  174014          MOV     2$,@DACB        ;PRIME THE DAC OUTPUT STAGE
005312  012777  000000  173776          MOV     #0,@STREG       ;INITIILIZE THE A/D STATUS REG
005320  017700  173776                  MOV     @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG
005324  004537  007742          1$:     JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 13
005330  000013                          CHAN13
005332  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED D/A VALUE
005336  000000                  2$:     0                       ;EXPECTED
005340  001354                          VWRAP                   ;SPREAD ALLOWED
005342  000413                          BR      3$              ;CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
005344  062737  000010  005336          ADD     #10,2$          ;UPDATE THE D/A OUTPUT VALUE
005352  013777  005336  173746          MOV     2$,@DACB        ;UPDATE THE D/A OUTPUT VOLTAGE
005360  022737  010000  005336          CMP     #10000,2$       ;TEST IF LAST STEP
005366  001356                          BNE     1$
005370  000401                          BR      4$              ;;BR TO NEXT TEST
005372  104004                  3$:     ERROR   4               ;CONVERTED D/A VALUE DID NOT EQUAL EXPECTED
005374  012777  007777  173724  4$:     MOV     #7777,@DACB     ;SET DAC "B" TO + F.S.

                                        .SBTTL
                                        .SBTTL  END OF AXV11-C ANALOG WRAPAROUND SECTION
```

```
                                        .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.
                                ;;******************************************************************
005402  000004                  TST42:  SCOPE
005404  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        ;AAV11-C TEST CONNECTOR IS NOT REQUIRED (IN FACT WILL ERROR IF PRESENT)

005412  005737  001370                  TST     TC2             ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
005416  001045                          BNE     TST43           ;;BR IF TEST CONNECTOR
005420  012777  000000  173670          MOV     #0,@STREG       ;INITIILIZE THE A/D STATUS REG
005426  017700  173670                  MOV     @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG
005432  004537  007742                  JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 14
005436  000014                          CHAN14
005440  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED VALUE
005444  004000                          4000                    ;EXPECTED
005446  010270                          V2                      ;SPREAD ALLOWED
005450  104004                          ERROR   4               ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

005452  004537  007742                  JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 15
005456  000015                          CHAN15
005460  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED VALUE
005464  004000                          4000
005466  010270                          V2                      ;SPREAD ALLOWED
005470  104004                          ERROR   4               ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

005472  004537  007742                  JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 16
005476  000016                          CHAN16
005500  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED VALUE
005504  004000                          4000
005506  010270                          V2                      ;SPREAD ALLOWED
005510  104004                          ERROR   4               ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

005512  004537  007742                  JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 17
005516  000017                          CHAN17
005520  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED VALUE
005524  004000                          4000
005526  010270                          V2                      ;SPREAD ALLOWED
005530  104004                          ERROR   4               ;CONVERTED VLAUE DID NOT EQUAL EXPECTED VALUE
```

```
                                        .SBTTL
                                        .SBTTL  START OF AAV11-C TO AXV11-C ANALOG WRAPAROUND SECTION
                                        .SBTTL

                                ;;**********************************************************************
                                ;*TEST 43        AAV11-C ANALOG WRAPAROUND TEST (DAC "A" TO A/D CHAN 14)
                                ;;**********************************************************************
(2)  005532  000004             TST43:  SCOPE
(1)  005534  012737  000001  001160     MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        ;AAV11-C TEST CONNECTOR IS REQUIRED

                                        TST     TC2             ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
                                        BEQ     TST44           ;;BR IF NO TEST CONNECTOR
                                        MOV     #0,2$           ;PRIME THE DAC OUTPUT VALUE
                                        MOV     #7777,@DAC0     ;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
                                        NOP
                                        NOP

                                1$:     JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 14
                                        CHAN14
                                        JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED D/A VALUE
                                2$:     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$,@DAC0        ;UPDATE THE D/A OUTPUT VOLTAGE
                                        CMP     #10000,2$       ;TEST IF LAST STEP
                                        BNE     1$
                                        BR      TST44           ;;BR TO NEXT TEST
                                7$:     0
                                10$:    ERROR   4               ;CONVERTED D/A VALUE DID NOT EQUAL EXPECTED
```

```
                                ;;*****************************************************************
                                ;*TEST 44       AAV11-C ANALOG WRAPAROUND TEST (DAC "B" TO A/D CHAN 15)
                                ;*****************************************************************
005674  000004                  TST44:  SCOPE
005676  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        ;AAV11-C TEST CONNECTOR IS REQUIRED

005704  005737  001370                  TST     TC2             ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
005710  001450                          BEQ     TST45           ;;BR IF NO TEST CONNECTOR
005712  012737  000000  005752          MOV     #0,2$           ;PRIME THE DAC OUTPUT VALUE
005720  012777  007777  173420          MOV     #7777,@DAC1     ;PRIME THE DAC OUTPUT STAGE
005726  012777  000000  173362          MOV     #0,@STREG       ;INITIILIZE THE A/D STATUS REG
005734  017700  173362                  MOV     @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG

005740  004537  007742          1$:     JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 15
005744  000015                          CHAN15
005746  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED D/A VALUE
005752  000000                  2$:     0
005754  001354                          VWRAP                   ;SPREAD ALLOWED
005756  000424                          BR      10$             ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
005760  062737  000010  005752          ADD     #10,2$          ;UPDATE THE D/A OUTPUT VALUE
005766  013737  005752  006026          MOV     2$,7$           ;COPY VALUE
005774  005137  006026                  COM     7$              ;INVERT DATA
006000  042737  170000  006026          BIC     #170000,7$      ;REMOVE EXTRA BITS
006006  013777  006026  173332          MOV     7$,@DAC1        ;UPDATE THE D/A OUTPUT VOLTAGE
006014  022737  010000  005752          CMP     #10000,2$       ;TEST IF LAST STEP
006022  001346                          BNE     1$
006024  000402                          BR      TST45           ;;BR TO NEXT TEST
006026  000000                  7$:     0
006030  104004                  10$:    ERROR   4               ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
```

```
                                ;;*****************************************************************
                                ;*TEST 45       AAV11-C ANALOG WRAPAROUND TEST (DAC "C" TO A/D CHAN 16)
                                ;*****************************************************************
006032  000004                  TST45:  SCOPE
006034  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        ;AAV11-C TEST CONNECTOR IS REQUIRED

006042  005737  001370                  TST     TC2             ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
006046  001450                          BEQ     TST46           ;;BR IF NO TEST CONNECTOR
006050  012737  000000  006110          MOV     #0,2$           ;PRIME THE DAC OUTPUT VALUE
006056  012777  007777  173264          MOV     #7777,@DAC2     ;PRIME THE DAC OUTPUT STAGE
006064  012777  000000  173224          MOV     #0,@STREG       ;INITIILIZE THE A/D STATUS REG
006072  017700  173224                  MOV     @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG

006076  004537  007742          1$:     JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 16
006102  000016                          CHAN16
006104  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED D/A VALUE
006110  000000                  2$:     0
006112  001354                          VWRAP                   ;SPREAD ALLOWED
006114  000424                          BR      10$             ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
006116  062737  000010  006110          ADD     #10,2$          ;UPDATE THE D/A OUTPUT VALUE
006124  013737  006110  006164          MOV     2$,7$           ;COPY VALUE
006132  005137  006164                  COM     7$              ;INVERT DATA
006136  042737  170000  006164          BIC     #170000,7$      ;REMOVE EXTRA BITS
006144  013777  006164  173176          MOV     7$,@DAC2        ;UPDATE THE D/A OUTPUT VOLTAGE
006152  022737  010000  006110          CMP     #10000,2$       ;TEST IF LAST STEP
006160  001346                          BNE     1$
006162  000402                          BR      TST46           ;;BR TO NEXT TEST
006164  000000                  7$:     0
006166  104004                  10$:    ERROR   4               ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
```

```
                                ;;*****************************************************************
                                ;*TEST 46        AAV11-C ANALOG WRAPAROUND TEST (DAC "D" TO A/D CHAN 17)
                                ;;*****************************************************************
006170  000004                  TST46:  SCOPE
006172  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        ;AAV11-C TEST CONNECTOR IS REQUIRED
006200  005737  001370                  TST     TC2             ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
006204  001450                          BEQ     TST47           ;;BR IF NO TEST CONNECTOR
006206  012737  000000  006246          MOV     #0,2$           ;PRIME THE DAC OUTPUT VALUE
006214  012777  007777  173130          MOV     #7777,@DAC3     ;PRIME THE DAC OUTPUT STAGE
006222  012777  000000  173066          MOV     #0,@STREG       ;INITIILIZE THE A/D STATUS REG
006230  017700  173066                  MOV     @ADBUFF,R0      ;READ A/D VALUE AND CLEAR A/D DONE FLAG

006234  004537  007742          1$:     JSR     R5,CONVRT       ;GET THE VALUE OF CHANNEL 17
006240  000017                          CHAN17
006242  004537  010100                  JSR     R5,COMPAR       ;COMPARE AGAINST EXPECTED D/A VALUE
006246  000000                  2$:     0
006250  001354                          VWRAP                   ;SPREAD ALLOWED
006252  000424                          BR      10$             ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
006254  062737  000010  006246          ADD     #10,2$          ;UPDATE THE D/A OUTPUT VALUE
006262  013737  006246  006322          MOV     2$,7$           ;COPY DATA
006270  005137  006322                  COM     7$              ;INVERT DATA
006274  042737  170000  006322          BIC     #170000,7$      ;REMOVE EXTRA BITS
006302  013777  006322  173042          MOV     7$,@DAC3        ;UPDATE THE D/A OUTPUT VOLTAGE
006310  022737  010000  006246          CMP     #10000,2$       ;TEST IF LAST S
006316  001346                          BNE     1$
006320  000402                          BR      TST47           ;;BR TO NEXT TEST
006322  000000                  7$:     0
006324  104004                  10$:    ERROR   4               ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
                                ;;*****************************************************************
                                ;*TEST 47        END OF AAV11-C TO AXV11-C ANALOG WRAPAROUND
                                ;;*****************************************************************
006326  000004                  TST47:  SCOPE
006330  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
006336  000207                          RTS     PC              ;EXIT AND RETURN TO CALLING ROUTINE
```

```
                                        .SBTTL  I/O SUB-SECTION "1"     REPORT THE CONVERTED A/D VALUES

006340  005077  172752        IOTST1: CLR     @STREG          ;CLEAR STATUS REGISTER
006344  104401  010430                TYPE    ,MSIO1          ;TYPE OUT HEADING
006350  005046                        CLR     -(SP)           ;CLEAR PSW
006352  012746  006360                MOV     #77$,-(SP)
006356  000002                        RTI
006360  104401  011154        77$:    TYPE    ,CCHAN          ;ASK OPERATOR FOR CHANNEL
006364  104413                        RDOCT
006366  012637  006454                MOV     (SP)+,10$       ;GET ANSWER
006372  042737  177760  006454        BIC     #177760,10$     ;REMOVE EXTRA BITS
006400  104401  011214                TYPE    ,GCHAN          ;ASK OPERATOR FOR GAIN
006404  104413                        RDOCT
006406  012637  010076                MOV     (SP)+,OTHER     ;GET ANSWER
006412  006137  010076                ROL     OTHER           ;MOVE TO BITS
006416  006137  010076                ROL     OTHER           ;2 + 3
006422  042737  177763  010076        BIC     #177763,OTHER   ;REMOVE ANY UNWANTED BITS
006430  104401  011121        1$:     TYPE    ,CH
006434  013746  006454                MOV     10$,-(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
006444  012702  000010        2$:     MOV     #10,R2          ;TYPEOUT COUNTER
006450  004537  007746        3$:     JSR     R5,CONVTR       ;GET AN AVERAGED VALUE FOR THIS CHANNEL
006454  000000                10$:    0
006456  104401  011124        4$:     TYPE    ,SPACE
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
006472  012701  010000                MOV     #10000,R1
006476  005301                5$:     DEC     R1
006500  001376                        BNE     5$
006502  005302                        DEC     R2              ;DECREMENT THE COUNTER
006504  001361                        BNE     3$              ;NO CARRIAGE RETURN
006506  104401  001171                TYPE    ,$CRLF          ;CARRIAGE RETURN
006512  000746                        BR      1$              ;REPEAT CONVERSION
```

```
                                                .SBTTL  I/O SUB-SECTION "2"     SCANNING CHANNELS AND GAIN SELECT - SECTION

006514  104401  010506          IOTST2: TYPE    ,MSIO2          ;TELL OPERATOR THE SECTION NAME

006520  005002                          CLR     R2              ;INITILIZE THE CHANNEL SCANNER
006522  005003                          CLR     R3              ;INITILIZE THE GAIN SELECT VALUE

006524  104401  001171          1$:     TYPE    ,$CRLF          ;MAKE A FRESH OUTPUT LINE
006530  012704  000007                  MOV     #7,R4           ;LOAD LINE WIDTH COUNTER

006534  104401  011121                  TYPE    ,CH             ;SHOW "CH" TEXT

006540  010246                          MOV     R2,-(SP)        ;LOAD THE CHANNEL CODE
006542  104403                          TYPOS
006544     002     001                  .BYTE   2,1

006546  104401  011146                  TYPE    ,ADOT           ;SEPERATE CH FROM GS

006552  112737  000060  011150          MOVB    #'0,AZERO       ;LOAD ASCII 0
006560  132703  000010                  BITB    #10,R3          ;TEST IF GS1 = 1
006564  001402                          BEQ     2$              ;BR IF NOT SET
006566  105237  011150                  INCB    AZERO           ;MAKE IT A ONE
006572  104401  011150          2$:     TYPE    ,AZERO          ;REPORT GS1 STATUS

006576  112737  000060  011150          MOVB    #'0,AZERO       ;LOAD ASCII 0
006604  132703  000004                  BITB    #4,R3           ;TEST IF GS0 = 1
006610  001402                          BEQ     3$              ;BR IF NOT SET
006612  105237  011150                  INCB    AZERO           ;MAKE IT A ONE
006616  104401  011150          3$:     TYPE    ,AZERO          ;REPORT GS0 STATUS

006622  010200                          MOV     R2,R0           ;GET CURRENT CHANNEL VALUE
006624  000300                          SWAB    R0              ;MOVE TO MUX POSITION
006626  050300                          BIS     R3,R0           ;ADD THE GAIN SELECT BITS
006630  010077  172462                  MOV     R0,@STREG       ;SELECT MUX AND GAIN BITS
006634  105277  172456          4$:     INCB    @STREG          ;START CONVERSION
006640  105777  172452          5$:     TSTB    @STREG          ;WAIT FOR A/D DONE
006644  100375                          BPL     5$

006646  104401  011124                  TYPE    ,SPACE          ;ENSURE SOME OUTPUT ROOM
006652  017746  172444                  MOV     @ADBUFF,-(SP)   ;READ CONVERTED VALUE AND SAVE FOR TYPOUT
006656  104403                          TYPOS
006660     004     001                  .BYTE   4,1

006662  105304                          DECB    R4              ;FINISHED A LINE ACROSS THE PAGE
006664  001363                          BNE     4$              ;BR AND CONVERT WITH CURRENT GAIN AND CHANNEL

006666  005202                          INC     R2              ;BUMP CHANNEL VALUE
006670  062703  000004                  ADD     #4,R3           ;BUMP GAIN SELECT VALUE
006674  042703  177763                  BIC     #177763,R3      ;REMOVE EXTRA BITS
006700  122702  000020                  CMPB    #20,R2          ;TEST IS LAST CHANNEL
006704  001307                          BNE     1$              ;BR IF NOT
006706  005002                          CLR     R2              ;INITILIZE THE CHANNEL
006710  104401  001171                  TYPE    ,$CRLF          ;INSERT ANOTHER FRESH OUTPUT LINE
006714  000703                          BR      1$              ;AND DO IT OVER AND OVER AND OVER AGAIN
```

```
                                                .SBTTL  I/O SUB-SECTION "3"     AXV11-C A/D INPUT ECHO TO AXV11-C D/A OUTPUT

006716  104401  010546          IOTST3: TYPE    ,MSI03          ;TELL OPERATOR THE NAME
006722  104401  011154                  TYPE    ,CCHAN          ;ASK OPER. FOR THE CHANNEL
006726  104413                          RDOCT
006730  012637  006776                  MOV     (SP)+,10$
006734  042737  177760  006776          BIC     #177760,10$     ;REMOVE EXTRA BITS
006742  104401  011214                  TYPE    ,GCHAN          ;ASK OPER FOR THE GAIN SELECT VALUE
006746  104413                          RDOCT
006750  012637  010076                  MOV     (SP)+,OTHER     ;GET THE ANSWER
006754  006337  010076                  ASL     OTHER           ;MOVE INTO
006760  006337  010076                  ASL     OTHER           ;GAIN SELECT POSITION
006764  042737  177763  010076          BIC     #177763,OTHER   ;REMOVE EXTRA BITS

006772  004537  007746          4$:     JSR     R5,CONVTR       ;CONVERT SELECTED CHANNEL AND GAIN
006776  000000                  10$:    0

007000  042737  170000  001360          BIC     #170000,TEMP    ;REMOVE EXTRA BITS
007006  013777  001360  172310          MOV     TEMP,@DACA      ;LOAD DAC "A"
007014  013777  001360  172304          MOV     TEMP,@DACB      ;LOAD DAC "B"

007022  000763                          BR      4$              ;LOOP BACK AND REPEAT

                                                .SBTTL  I/O SUB-SECTION "4"     AXV11-C D/A RAMPS

007024  104401  010611          IOTST4: TYPE    ,MSI04          ;TELL OPERATOR THE NAME
007030  012703  000000                  MOV     #0,R3           ;LOAD DAC - F.S. VALUE
007034  012704  007777                  MOV     #7777,R4        ;LOAD DAC + F.S. VALUE

007040  012705  010000          1$:     MOV     #BIT12,R5       ;LOAD LOOP COUNT
007044  010377  172254          2$:     MOV     R3,@DACA        ;LOAD DAC "A"
007050  010477  172252                  MOV     R4,@DACB        ;LOAD DAC "B"
007054  005305                          DEC     R5              ;FINISHED ALL BITS ?
007056  001403                          BEQ     3$              ;BR IF DONE
007060  005304                          DEC     R4              ;LOWER DAC "B" VALUE
007062  005203                          INC     R3              ;RAISE DAC "A" VALUE
007064  000767                          BR      2$              ;DO NEXT COUNT

007066  012705  010000          3$:     MOV     #BIT12,R5       ;LOAD LOOP COUNT
007072  010377  172226          4$:     MOV     R3,@DACA        ;LOAD DAC "A"
007076  010477  172224                  MOV     R4,@DACB        ;LOAD DAC "B"
007102  005305                          DEC     R5              ;FINISHED ALL BITS ?
007104  001755                          BEQ     1$
007106  005303                          DEC     R3              ;LOWER DAC "A" VALUE
007110  005204                          INC     R4              ;RAISE DAC "B" VALUE
007112  000767                          BR      4$              ;DO NEXT COUNT
```

```
                                        .SBTTL  I/O SUB-SECTION "5"     AXV11-C D/A CALIBRATION

007114  104401  010664        IOTST5: TYPE    ,MSIO5          ;TELL OPERATOR THE NAME
007120  012703  000000                MOV     #0,R3           ;LOAD DAC - F.S. VALUE
007124  012704  007777                MOV     #7777,R4        ;LOAD DAC + F.S. VALUE
007130  012705  004000                MOV     #4000,R5        ;LOAD 0.0 F.S. VALUE

007134  010377  172164        1$:     MOV     R3,@DACA        ;LOAD DAC "A" TO - F.S.
007140  010377  172162                MOV     R3,@DACB        ;LOAD DAC "B" TO - F.S.
007144  104412                        RDLIN
007146  012600                        MOV     (SP)+,R0        ;REMOVE CHARACTER
007150  010477  172150                MOV     R4,@DACA        ;LOAD DAC "A" TO + F.S.
007154  010477  172146                MOV     R4,@DACB        ;LOAD DAC "B" TO + F.S.
007160  104412                        RDLIN
007162  012600                        MOV     (SP)+,R0        ;REMOVE CHARACTER
007164  010577  172134                MOV     R5,@DACA        ;LOAD DAC "A" TO MID POINT
007170  010577  172132                MOV     R5,@DACB        ;LOAD DAC "B" TO MID POINT
007174  104412                        RDLIN
007176  012600                        MOV     (SP)+,R0        ;REMOVE CHARACTER
007200  000755                        BR      1$

                                        .SBTTL  I/O SUB-SECTION "6"     AXV11-C D/A SQUARE WAVE

007202  104401  010731        IOTST6: TYPE    ,MSIO6          ;TELL OPERATOR THE NAME
007206  012703  000000                MOV     #0,R3           ;LOAD DAC - F.S.
007212  012704  007777                MOV     #7777,R4        ;LOAD DAC + F.S.

007216  010377  172102        1$:     MOV     R3,@DACA        ;LOAD DAC "A" TO MIN LEVEL
007222  010377  172100                MOV     R3,@DACB        ;LOAD DAC "B" TO MIN LEVEL
007226  004737  001426                JSR     PC,STALL        ;DELAY
007232  010477  172066                MOV     R4,@DACA        ;LOAD DAC "A" TO MAX LEVEL
007236  010477  172064                MOV     R4,@DACB        ;LOAD DAC "B" TO MAX LEVEL
007242  004737  001426                JSR     PC,STALL        ;DELAY
007246  000763                        BR      1$              ;LOOP BACK AND DO AGAIN

                                        .SBTTL  I/O SUB-SECTION "7"     AXV11-C D/A OUTPUT TO A/D INPUT

007250  104401  011022        IOTST7: TYPE    ,MSIO7          ;TELL OPERATOR THE SUB-SECTION NAME
007254  005003                        CLR     R3              ;INITILIZE THE DAC VALUE
007256  104401  001171        1$:     TYPE    ,$CRLF          ;ENSURE FRESH OUTPUT LINE
007262  012705  000010                MOV     #10,R5          ;LOAD LINE WIDTH COUNTER

007266  105277  172024        2$:     INCB    @STREG          ;START CONVERSION
007272  105777  172020        3$:     TSTB    @STREG          ;WAIT FOR A/D DONE
007276  100375                        BPL     3$
007300  010377  172020                MOV     R3,@DACA        ;LOAD "DAC A" OUTPUT VALUE
007304  017746  172012                MOV     @ADBUFF,-(SP)   ;READ AND STORE A/D VALUE
007310  104403                        TYPOS
007312     004     001                .BYTE   4,1
007314  005203                        INC     R3              ;UPDATE TO NEXT D/A VALUE
007316  042703  170000                BIC     #170000,R3      ;ENSURE ONLY 12 BITS LONG
007322  005305                        DEC     R5              ;IS THE WIDTH FINISHED ?
007324  001754                        BEQ     1$              ;BR AND START FRESH OUTPUT LINE
007326  104401  011124                TYPE    ,SPACE          ;ENSURE SOME ROOM
007332  000755                        BR      2$              ;AND DO ANOTHER CONVERSION
```

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

007334                          BEGINL:
007334  004737  002560          1$:     JSR     PC,BEGL                 ;LOGIC TESTS
007340  012737  007334  010304          MOV     #1$,AGTST               ;ADDRESS FOR EOP
007346  000137  010306                  JMP     $EOP                    ;TYPE END OF PASS

                                .SBTTL          AUTO TEST
007352                          BEGINA:
007352  004737  002560          1$:     JSR     PC,BEGL                 ;LOGIC TESTS
007356  004737  004136                  JSR     PC,WRAP
007362  012737  007352  010304          MOV     #1$,AGTST               ;ADDRESS FOR EOP
007370  000137  010306                  JMP     $EOP                    ;TYPE END OF PASS

                                .SBTTL          WRAPAROUND TEST
007374                          BEGINW:
007374  004737  004136          1$:     JSR     PC,WRAP                 ;WRAPAROUND TESTS
007400  012737  007374  010304          MOV     #1$,AGTST
007406  000137  010306                  JMP     $EOP                    ;INCREMENTS $PASS

                                .SBTTL          DMT TEST STARTUP
007412  032737  000001  001252  BEGIND: BIT     #BIT0,$DEVM     ;TEST IF KWV11-C CONNECTED TO RTC TRIGGER
007420  001402                          BEQ     1$              ;BR IF NOT
007422  005237  001374                  INC     KWAD            ;SET KW CONNECTED TO AD RTC TRIG - FLAG
007426  032737  000002  001252  1$:     BIT     #BIT1,$DEVM     ;TEST IF KWV11-C CONNECTED TO EXT TRIG AND "F2"
007434  001402                          BEQ     2$              ;BR IF NOT
007436  005237  001376                  INC     KWEX            ;SET KW CONNECTED TO AD EXT TRIG - FLAG
007442  032737  000004  001252  2$:     BIT     #BIT2,$DEVM     ;TEST IF TEST FIXTURE CONNECTED
007450  001402                          BEQ     3$              ;BR IF NOT
007452  005237  001366                  INC     TC1             ;SET TEST FIXTURE PRESENT FLAG
007456  032737  000010  001252  3$:     BIT     #BIT3,$DEVM     ;TEST IF AAV11-C CONNECTED TO TEST FIXTURE
007464  001402                          BEQ     4$              ;BR IF NOT
007466  005237  001370                  INC     TC2             ;SET AAV11-C ANALOG WRAPAROUND FLAG
007472  032737  000020  001252  4$:     BIT     #BIT4,$DEVM     ;TEST IF BEVENT AND "F1" CONNECTED
007500  001402                          BEQ     5$              ;BR IF NOT
007502  005237  001402                  INC     BTEX            ;SET BEVENT AND "F1" FLAG
007506  032737  000040  001252  5$:     BIT     #BIT5,$DEVM     ;TEST IF MODULE IS AN "ADV11-C"
007514  001402                          BEQ     6$              ;BR IF NOT
007516  005237  001372                  INC     ADV11C          ;SET "ADV11-C" FLAG
007522  000240                  6$:     NOP
007524  000240                          NOP
007526  000240                          NOP
007530  000240                          NOP
007532  000240                          NOP
007534  000137  007352                  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              ;              "B"
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  007742  005037  010076                CONVRT: CLR     OTHER           ;REMOVE EXTRA BITS
1189  007746  012500                        CONVTR: MOV     (R5)+,R0        ;GET CHANNEL VALUE
1190  007750  010037  001362                        MOV     R0,CHANL
1191  007754  000300                                SWAB    R0
1192  007756  053700  010076                        BIS     OTHER,R0        ;ADD GAIN SELECT IF NEEDED
1193  007762  005037  001360                        CLR     TEMP
1194  007766  010077  171324                        MOV     R0,@STREG       ;LOAD CHANNEL INTO MIX BITS
1195  007772  012700  010000                        MOV     #10000,R0
1196  007776  005300                        2$:     DEC     R0
1197  010000  001376                                BNE     2$
1198  010002  012777  001440  171320                MOV     #RETURN,@VECTOR ;LOAD VECTOR
1199  010010  012700  000010                        MOV     #10,R0          ;SET UP COUNTER
1200  010014  152777  000101  171274  1$:     BISB    #101,@STREG     ;SET INTRPT. EN., START CONV.
1201  010022  000001                                WAIT                    ;WAIT FOR CONVERSION
1202  010024  017737  171272  010074                MOV     @ADBUFF,77$     ;READ CONVERTED VALUE
1203  010032  042737  170000  010074                BIC     #170000,77$     ;REMOVE HIGH BITS
1204  010040  063737  010074  001360                ADD     77$,TEMP        ;READ BUFFER
1205  010046  005300                                DEC     R0
1206  010050  001361                                BNE     1$              ;DO 8 TIMES
1207  010052  006237  001360                        ASR     TEMP            ;AVERAGE VALUE
1208  010056  006237  001360                        ASR     TEMP
1209  010062  006237  001360                        ASR     TEMP
1210  010066  005537  001360                        ADC     TEMP
1211  010072  000205                                RTS     R5              ;RETURN
1212  010074  000000                        77$:    0
1213  010076  000000                        OTHER:  0
1214
1215                                        ;COMPARE $GDDAT AND $BDDAT;;
1216  010100  012537  001124                COMPAR: MOV     (R5)+,$GDDAT    ;GET GOOD DATA
1217  010104  013537  001364                        MOV     @(R5)+,SPREAD   ;GET SPREAD
1218  010110  013737  001360  001126                MOV     TEMP,$BDDAT     ;GET BAD(ACTUAL) DATA
1219  010116  013700  001124                        MOV     $GDDAT,R0
1220  010122  163700  001126                        SUB     $BDDAT,R0       ;GET DIFFERENCE
1221  010126  100001                                BPL     7$
1222  010130  005400                                NEG     R0
1223  010132  020037  001364                7$:     CMP     R0,SPREAD       ;COMPARE IT TO SPREAD
1224  010136  003001                                BGT     10$             ;GO TO ERROR PRINTOUT
1225  010140  005725                                TST     (R5)+           ;BUMP RETURN POINTER AROUND ERROR CALL
1226  010142  000205                        10$:    RTS     R5
```

```
                                        ;;SUBROUTINE TO TYPE INTRPT. TST MSG.;;
010144  005737  001202          DUMW:   TST     $PASS
010150  001021                  BNE     20$
010152  012737  010214  001110          MOV     #20$,$LPERR
010160  012737  010214  001106          MOV     #20$,$LPADR
010166  104401  011515                  TYPE    ,METST          ;TYPE ASCIZ STRING
010172  010046                          MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
                                                                ;;TYPE TEST NO.
010174  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
010176     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
010177     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
010200  104401  011370                  TYPE    ,ONAD
010204  013746  001316                  MOV     STREG,-(SP)     ;;SAVE STREG FOR TYPEOUT
                                                                ;;TYPE BUS ADDRESS
010210  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
010212     006                          .BYTE   6               ;;TYPE 6 DIGITS
010213     001                          .BYTE   1               ;;TYPE LEADING ZEROS
010214  000207                  20$:    RTS     PC

010216  005737  001202          DUMC:   TST     $PASS
010222  001010                  BNE     30$
010224  012737  010244  001110          MOV     #30$,$LPERR
010232  012737  010244  001106          MOV     #30$,$LPADR
010240  104401  011133                  TYPE    ,DONE
010244  000207                  30$:    RTS     PC

                                        ;SUBROUTINE TO RESET & SET INTRPT. EN.;
010246  000005                  RST:    RESET
010250  052777  000100  170666          BIS     #100,@$TKS
010256  005046                          CLR     -(SP)           ;CLEAR PSW
010260  012746  010266                  MOV     #1$,-(SP)
010264  000002                          RTI
010266  000207                  1$:     RTS     PC

010270  000002                  V2:     2
010272  000012                  V12:    12

010274  052777  000100  170642  AGATST: BIS     #100,@$TKS
010302  000137                          JMP     @(PC)+
010304  001522                  AGTST:  BEGIN0
```

```
                        .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

010306                  $EOP:
000240                          NOP
005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
005037  001160                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
005237  001202                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
042737  100000  001202          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
005327                          DEC     (PC)+           ;;LOOP?
000001                  $EOPCT: .WORD   1
003022                          BGT     $DOAGN          ;;YES
012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
000001                  $ENDCT: .WORD   1
010334                          $EOPCT
104401  010413                  TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
013746  001202                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
104401  010410                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
000005                          RESET                   ;;CLEAR THE WORLD
004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
000240                          NOP                     ;;SAVE ROOM
000240                          NOP                     ;;FOR
000240                          NOP                     ;;ACT11
010404                  $DOAGN:
000137                          JMP     @(PC)+          ;;RETURN
010274                  $RTNAD: .WORD   AGATST
377     377     000     $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
015     042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
050040  051501  020123
000043
```

```
                                                .SBTTL          ASCII MESSAGES
1265  010430  020200  042522  047520  MSIO1:  .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  MSIO2:  .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  MSIO3:  .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  MSIO4:  .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  MSIO5:  .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  MSIO6:  .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  MSIO7:  .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
```

```
                011104  040     000
1274            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:  .ASCIZ  <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          .ASCIZ  \IS KWV11-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          .ASCIZ  \IS KWV11-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          .ASCIZ  \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          .ASCIZ  \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          .ASCIZ  \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:  .ASCIZ  <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          .ASCIZ  \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          .ASCIZ  \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          .ASCIZ  <15><12>/  /
1317            013176  005015  051511  045440  HEAD2:  .ASCIZ  <15><12>\IS KWV11-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:    .ASCIZ  /STATUS REG. ERROR/
                013254  020123  042522  027107
                013262  042440  051122  051117
                013270     000
1319            013271     106  044501  042514  EM2:    .ASCIZ  /FAILED TO INTERRUPT/
                013276  020104  047524  044440
                013304  052116  051105  052522
                013312  052120     000
```

```
 1320  013315     125  042516  050130  EM3:    .ASCIZ  /UNEXPECTED INTERRUPT/
       013322  041505  042524  020104
       013330  047111  042524  051122
       013336  050125  000124
 1321  013342  051105  047522  020122  EM4:    .ASCIZ  #ERROR ON A/D CHANNEL#
       013350  047117  040440  042057
       013356  041440  040510  047116
       013364  046105     000
 1322  013367     105  051122  041520  DH1:    .ASCIZ  /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:    .ASCIZ  /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:    .ASCIZ  /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
```

```
 1332                                   .SBTTL  TTY INPUT ROUTINE
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   .ENABL  LSB
  (1)  013600  000000                   $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
  (1)  013602  000000                   $TKQIN: .WORD   0               ;;INPUT POINTER
  (1)  013604  000000                   $TKQOUT: .WORD  0               ;;OUTPUT POINTER
  (1)  013606  000040                   $TKQSRT: .BLKB  32.             ;;TTY KEYBOARD QUEUE
  (1)          013646                   $TKQEND=.
  (1)
  (1)                                   ;*TK INITIALIZE ROUTINE
  (1)                                   ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
  (1)                                   ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
  (1)                                   ;
  (1)                                   ;*CALL:
  (1)                                   ;*      JSR     PC,$TKINT
  (1)                                   ;*      RETURN
  (1)                                   ;
  (1)  013646  005037  013600           $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
  (1)  013652  012737  013606  013602           MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
  (1)  013660  013737  013602  013604           MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
  (1)  013666  012737  013716  000060           MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
  (1)  013674  012737  000200  000062           MOV     #200,@#TKVEC+2  ;;"BR" LEVEL 4
  (1)  013702  005777  165240                   TST     @$TKB           ;;CLEAR DONE FLAG
  (1)  013706  012777  000100  165230           MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
  (1)  013714  000207                           RTS     PC              ;;RETURN TO CALLER
  (1)
  (1)                                   ;*TK SERVICE ROUTINE
  (1)                                   ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
  (1)                                   ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
  (1)                                   ;*IT IN THE QUEUE.
  (1)                                   ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
  (1)                                   ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (BEGIN2)
  (1)                                   ;
  (1)  013716  117746  165224           $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
  (1)  013722  042716  177600                   BIC     #^C177,(SP)     ;;STRIP THE JUNK
  (1)  013726  021627  000021                   CMP     (SP),#$XON      ;;IS IT A RANDOM XON?                   ;RAN001
  (1)  013732  001002                           BNE     30$             ;;BRANCH IF NO                          ;RAN001
  (1)  013734  005726                           TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK            ;RAN001
  (1)  013736  000002                           RTI                     ;;RETURN                                ;RAN001
  (1)  013740                           30$:                                                                    ;RAN001
  (1)  013740  021627  000003                   CMP     (SP),#3         ;;IS IT A CONTROL C?
  (1)  013744  001007                           BNE     1$              ;;BRANCH IF NO
  (1)  013746  104401  015074                   TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
  (1)  013752  004737  013646                   JSR     PC,$TKINT       ;;INIT THE KEYBOARD
  (1)  013756  005726                           TST     (SP)+           ;;CLEAN UP STACK
  (1)  013760  000137  001530                   JMP     BEGIN2          ;;CONTROL C RESTART
  (1)  013764  021627  000007           1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
  (1)  013770  001004                           BNE     2$              ;;BRANCH IF NO
  (1)  013772  022737  000176  001140           CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
  (1)  014000  001500                           BEQ     6$              ;;GO TO SWR CHANGE
  (1)
  (1)  014002                           2$:
  (1)  014002  022737  000040  013600           CMP     #32.,$TKCNT     ;;IS THE QUEUE FULL?
  (1)  014010  001004                           BNE     3$              ;;BRANCH IF NO
  (1)  014012  104401  001164                   TYPE    ,$BELL          ;;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     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  014034  005726                        TST     (SP)+           ;;CLEAN CHAR OFF STACK
(1)  014036  105777  165102          31$:  TSTB    @$TKS           ;;WAIT FOR A CHAR
(1)  014042  100375                        BPL     31$             ;;LOOP UNTIL ITS THERE
(1)  014044  117746  165076                MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
(1)  014050  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  014054  022627  000021                CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  014060  001366                        BNE     31$             ;;BRANCH IF NO
(1)  014062  012777  000100  165054        MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
(1)  014070  000002                        RTI                     ;;RETURN
(1)  014072  005237  013600          32$:  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$:   MOVB    (SP)+,@$TKQIN   ;;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)                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)                                        ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
(1)                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
(1)                                        ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
(1)  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    @$TKS           ;;IS A CHAR WAITING?
(1)  014162  100121                        BPL     15$             ;;IF NOT, EXIT
(1)  014164  117746  164756                MOVB    @$TKB,-(SP)     ;;YES
(1)  014170  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
(1)  014174  021627  000007                CMP     (SP),#7         ;;IS IT A CONTROL-G?
(1)  014200  001300                        BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
(1)                                                                ;;AND EXIT
(1)
(2)                                        ;;**********************************************************************
(1)                                        ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
(1)                                        ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
(1)                                        ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
(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,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
(1)  014220  005077  164720                CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  014224  112737  000001  001135        MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
(1)
(1)  014232  104401  015106                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  014236  104401  015113  $GTSWR: TYPE  ,$MSWR          ;;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$:    ADD     #6,SP            ;;CLEAR UP STACK
(1)  014404  104401  001171        14$:    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)                                     ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDCHR                           ;;GET A CHARACTER FROM THE QUEUE
(1)                                     ;*      RETURN HERE                     ;;CHARACTER IS ON THE STACK
(1)                                     ;*                                      ;;WITH PARITY BIT STRIPPED OFF
(1)                                     ;
(1)
(1)  014510  011646                     $RDCHR: MOV     (SP),-(SP)              ;;PUSH DOWN THE PC AND
(1)  014512  016666  000004  000002             MOV     4(SP),2(SP)             ;;THE PS
(1)  014520  005066  000004                     CLR     4(SP)                   ;;GET READY FOR A CHARACTER
(2)  014524  005046                             CLR     -(SP)                   ;;PUT NEW PS ON STACK
(2)  014526  012746  014534                     MOV     #64$,-(SP)              ;;PUT NEW PC ON  STACK
(2)  014532  000002                             RTI                             ;;POP NEW PC AND PS
(2)  014534                             64$:
(1)  014534  005737  013600             1$:     TST     $TKCNT                  ;;WAIT ON A CHARACTER
(1)  014540  001775                             BEQ     1$
(1)  014542  005337  013600                     DEC     $TKCNT                  ;;DECREMENT THE COUNTER
(1)  014546  117766  177032  000004             MOVB    @$TKQOUT,4(SP)          ;;GET ONE CHARACTER
(1)  014554  005237  013604                     INC     $TKQOUT                 ;;UPDATE THE POINTER
(1)  014560  023727  013604  013646             CMP     $TKQOUT,#$TKQEND        ;;DID IT GO OFF OF THE END
(1)  014566  001003                             BNE     2$                      ;;BRANCH IF NO
(1)  014570  012737  013606  013604             MOV     #$TKQSRT,$TKQOUT        ;;RESET THE POINTER
(1)  014576  000002                     2$:     RTI                             ;;RETURN
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDLIN                           ;;INPUT A STRING FROM THE TTY
(1)                                     ;*      RETURN HERE                     ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;*                                      ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  014600  010346                     $RDLIN: MOV     R3,-(SP)                ;;SAVE R3
(1)  014602  005046                             CLR     -(SP)                   ;;CLEAR THE RUBOUT KEY
(1)  014604  012703  015034             1$:     MOV     #$TTYIN,R3              ;;GET ADDRESS
(1)  014610  022703  015074             2$:     CMP     #$TTYIN+32.,R3          ;;BUFFER FULL?
(1)  014614  101456                             BLOS    4$                      ;;BR IF YES
(1)  014616  104411                             RDCHR                           ;;GO READ ONE CHARACTER FROM THE TTY
(1)  014620  112613                             MOVB    (SP)+,(R3)              ;;GET CHARACTER
(1)  014622  122713  000177             10$:    CMPB    #177,(R3)               ;;IS IT A RUBOUT
(1)  014626  001022                             BNE     5$                      ;;BR IF NO
(1)  014630  005716                             TST     (SP)                    ;;IS THIS THE FIRST RUBOUT?
(1)  014632  001007                             BNE     6$                      ;;BR IF NO
(1)  014634  112737  000134  015032             MOVB    #'\,9$                  ;;TYPE A BACK SLASH
(1)  014642  104401  015032                     TYPE    ,9$
(1)  014646  012716  177777                     MOV     #-1,(SP)                ;;SET THE RUBOUT KEY
(1)  014652  005303                     6$:     DEC     R3                      ;;BACKUP BY ONE
(1)  014654  020327  015034                     CMP     R3,#$TTYIN              ;;STACK EMPTY?
(1)  014660  103434                             BLO     4$                      ;;BR IF YES
(1)  014662  111337  015032                     MOVB    (R3),9$                 ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  014666  104401  015032                     TYPE    ,9$                     ;;GO TYPE
(1)  014672  000746                             BR      2$                      ;;GO READ ANOTHER CHAR.
(1)  014674  005716                     5$:     TST     (SP)                    ;;RUBOUT KEY SET?
(1)  014676  001406                             BEQ     7$                      ;;BR IF NO
(1)  014700  112737  000134  015032             MOVB    #'\,9$                  ;;TYPE A BACK SLASH
(1)  014706  104401  015032                     TYPE    ,9$
```

```
(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    ,9$
(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          $TTYIN: .BLKB   32.                     ;;RESERVE 32. BYTES FOR TTY INPUT
(1)  015074  041536  005015     000  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL "C"
(1)  015101     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
(1)  015106  043536  005015     000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
(1)  015113     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  015120  036440  000040
(1)  015124  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
(1)  015132  020075     000
(1)          015136                  .EVEN
```

```
1334                                    .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 (1)                                    ;*CHANGE IT TO BINARY.
 (1)                                    ;*CALL:
 (1)                                    ;*      RDOCT                   ;;READ AN OCTAL NUMBER
 (1)                                    ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
 (1)                                    ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
 (1)
 (1)  015136  011646                    $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
 (1)  015140  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
 (3)  015146  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 (3)  015150  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 (3)  015152  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 (1)  015154  104412                    1$:     RDLIN                   ;;READ AN ASCIZ LINE
 (1)  015156  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
 (1)  015160  005001                            CLR     R1              ;;CLEAR DATA WORD
 (1)  015162  005002                            CLR     R2
 (1)  015164  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
 (1)  015166  001412                            BEQ     3$              ;;IF ZERO GET OUT
 (1)  015170  006301                            ASL     R1              ;;*2
 (1)  015172  006102                            ROL     R2
 (1)  015174  006301                            ASL     R1              ;;*4
 (1)  015176  006102                            ROL     R2
 (1)  015200  006301                            ASL     R1              ;;*8
 (1)  015202  006102                            ROL     R2
 (1)  015204  042716  177770                    BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
 (1)  015210  062601                            ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
 (1)  015212  000764                            BR      2$              ;;LOOP
 (1)  015214  005726                    3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
 (1)  015216  010166  000012                    MOV     R1,12(SP)       ;;SAVE THE RESULT
 (1)  015222  010237  015236                    MOV     R2,$HIOCT
 (3)  015226  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
 (3)  015230  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
 (3)  015232  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
 (1)  015234  000002                            RTI                     ;;RETURN
 (1)  015236  000000                    $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
1336                                    .SBTTL  POWER DOWN AND UP ROUTINES
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;POWER DOWN ROUTINE
 (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)                                    ;POWER UP ROUTINE
 (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                    ;REPORT THE POWER FAILURE
 (1)  015374  015406                    $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
 (1)  015376  000002                            RTI
 (1)  015400  000000                    $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
 (1)  015402  000776                            BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
 (1)  015404  000000                    $SAVR6: 0                       ;;PUT THE SP HERE
 (1)  015406  005015  047520  042527    $POWER: .ASCIZ  <15><12>''POWER''
 (1)  015414  000122
 (1)                                            .EVEN
```

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

```
(1)  015636  011637  001106                MOV     (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
(1)  015642  011637  001110                MOV     (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
(1)  015646  005037  001162                CLR     $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  015652  112737  000001  001115        MOVB    #1,$ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  015660  013777  001102  163254  $OVER: MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER
(1)  015666  013716  001106                MOV     $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
(1)  015672  000002                        RTI                      ;;FIXES PS
(1)  015674  003720                $MXCNT: 2000.                    ;;MAX. NUMBER OF ITERATIONS
1339                                       .SBTTL  ERROR HANDLER ROUTINE
(1)
(2)                                        ;;**********************************************************************
(1)                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)                                        ;*AND GO TO $ERRTYP ON ERROR
(1)                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                        ;*SW15=1        HALT ON ERROR
(1)                                        ;*SW13=1        INHIBIT ERROR TYPEOUTS
(1)                                        ;*SW10=1        BELL ON ERROR
(1)                                        ;*SW09=1        LOOP ON ERROR
(1)                                        ;*CALL
(1)                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1)  015676                                $ERROR:
(1)  015676  104410                        CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
(1)  015700  105237  001103                7$:     INCB    $ERFLG           ;;SET THE ERROR FLAG
(1)  015704  001775                        BEQ     7$               ;;DON'T LET THE FLAG GO TO ZERO
(1)  015706  013777  001102  163226        MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  015714  032777  002000  163216        BIT     #BIT10,@SWR      ;;BELL ON ERROR?
(1)  015722  001402                        BEQ     1$               ;;NO - SKIP
(1)  015724  104401  001164                TYPE    ,$BELL           ;;RING BELL
(1)  015730  005237  001112                1$:     INC     $ERTTL           ;;COUNT THE NUMBER OF ERRORS
(1)  015734  011637  001116                MOV     (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  015740  162737  000002  001116        SUB     #2,$ERRPC
(1)  015746  117737  163144  001114        MOVB    @$ERRPC,$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  015754  032777  020000  163156        BIT     #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
(1)  015762  001004                        BNE     20$              ;;SKIP TYPEOUTS
(1)  015764  004737  016076                JSR     PC,$ERRTYP       ;;GO TO USER ERROR ROUTINE
(1)  015770  104401  001171                TYPE    ,$CRLF
(1)  015774                                20$:
(1)  015774  122737  000001  001214        CMPB    #APTENV,$ENV     ;;RUNNING IN APT MODE
(1)  016002  001007                        BNE     2$               ;;NO,SKIP APT ERROR REPORT
(1)  016004  113737  001114  016016        MOVB    $ITEMB,21$       ;;SET ITEM NUMBER AS ERROR NUMBER
(1)  016012  004737  016604                JSR     PC,$ATY4         ;;REPORT FATAL ERROR TO APT
(1)  016016     000                        21$:    .BYTE   0
(1)  016017     000                                BYTE    0
(1)  016020  000777                        22$:    BR      22$              ;;APT ERROR LOOP
(1)  016022  005777  163112                2$:     TST     @SWR             ;;HALT ON ERROR
(1)  016026  100002                        BPL     3$               ;;SKIP IF CONTINUE
(1)  016030  000000                        HALT                     ;;HALT ON ERROR!
(1)  016032  104410                        CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
(1)  016034  032777  001000  163076        3$:     BIT     #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
(1)  016042  001402                        BEQ     4$               ;;BR IF NO
(1)  016044  013716  001110                MOV     $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
(1)  016050  005737  001162                4$:     TST     $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
(1)  016054  001402                        BEQ     5$               ;;BR IF NONE
(1)  016056  013716  001162                MOV     $ESCAPE,(SP)     ;;FUDGE RETURN ADDRESS FOR ESCAPE
```

```
(1)  016062                                5$:
(1)  016062  022737  010374  000042              CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
(1)  016070  001001                              BNE     6$              ;;BRANCH IF NO
(1)  016072  000000                              HALT                    ;;YES
(1)  016074                                6$:
(1)  016074  000002                              RTI                     ;;RETURN
1340                                       .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                        ;;*****************************************************************
(1)                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1)                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1)                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  016076                                $ERRTYP:
(1)  016076  104401  001171                      TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016102  010046                              MOV     R0,-(SP)        ;;SAVE R0
(1)  016104  005000                              CLR     R0              ;;PICKUP THE ITEM INDEX
(1)  016106  153700  001114                      BISB    @#$ITEMB,R0
(1)  016112  001004                              BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                                      ;;TYPE THE PC OF THE ERROR
(2)  016114  013746  001116                      MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                                      ;;ERROR ADDRESS
(2)  016120  104402                              TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016122  000426                              BR      6$              ;;GET OUT
(1)  016124  005300                        1$:   DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1)  016126  006300                              ASL     R0              ;;      WORK FOR THE ERROR TABLE
(1)  016130  006300                              ASL     R0
(1)  016132  006300                              ASL     R0
(1)  016134  062700  001256                      ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(1)  016140  012037  016150                      MOV     (R0)+,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$:   MOV     (R0)+,4$        ;;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$:   MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
(1)  016176  001004                              BNE     7$              ;;GO TYPE THE DATA
(1)  016200  012600                        6$:   MOV     (SP)+,R0        ;;RESTORE R0
(1)  016202  104401  001171                      TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016206  000207                              RTS     PC              ;;RETURN
(1)  016210                                7$:
(2)  016210  013046                              MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
(2)  016212  104402                              TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016214  005710                              TST     (R0)            ;;IS THERE ANOTHER NUMBER?
(1)  016216  001770                              BEQ     6$              ;;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
```

```
                                .SBTTL  TYPE ROUTINE

                                ;;*****************************************************************
                                ;*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
                                ;*

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

```
                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                          $TYPEC:
(1)  016444  105777  162474                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
(1)  016450  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
(1)  016452  017746  162470                  MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
(1)  016456  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
(1)  016462  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
(1)  016466  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
(1)  016470                          101$:                                                           ;MJD001
(1)  016470  105777  162450                  TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
(1)  016474  100375                          BPL     101$                                            ;MJD001
(1)  016476  117716  162444                  MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
(1)  016502  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
(1)  016506  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
(1)  016512  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
(1)  016514                          102$:                                                           ;MJD001
(1)  016514  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
(1)  016516                          10$:                                                            ;MJD001
(1)  016516  105777  162426                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  016522  100375                          BPL     10$                                             ;MJD001
(1)  016524  116677  000002  162420          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  016532  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  016540  001003                          BNE     1$              ;;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    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  016642  001425                          BEQ     3$              ;;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     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  016700  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  016702  010037  001212          MOV     R0,$MSGLGT      ;;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  3$:    MOV     @4(SP),4$       ;;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  APTCSUP=040
```

```
1345                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
 (1)                                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
 (1)                                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
 (1)                                    ;*CALL:
 (1)                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 (1)                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
 (1)                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
 (1)                                    ;*      .BYTE   M               ;;M=1 OR 0
 (1)                                    ;*                                      ;;1=TYPE LEADING ZEROS
 (1)                                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
 (1)                                    ;*
 (1)                                    ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
 (1)                                    ;*$TYPOS OR $TYPOC
 (1)                                    ;*CALL:
 (1)                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 (1)                                    ;*      TYPON                   ;;CALL FOR TYPEOUT
 (1)                                    ;*
 (1)                                    ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
 (1)                                    ;*CALL:
 (1)                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 (1)                                    ;*      TYPOC                   ;;CALL FOR TYPEOUT
 (1)
 (1)  017034  017646  000000            $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
 (1)  017040  116637  000001  017257            MOVB    1(SP),$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$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  017200  052703  000060            5$:     BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  017204  052703  000040                    BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  017210  110337  017254                    MOVB    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)                                    ;*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)  017262  010146                    $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
(1)  017264  016601  000006                    MOV     6(SP),R1        ;;GET THE INPUT NUMBER
(1)  017270  000261                            SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1)  017272  112737  000060  017334    1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
(1)  017300  006101                            ROL     R1              ;;GET THIS BIT
(1)  017302  001406                            BEQ     2$              ;;DONE?
(1)  017304  105537  017334                    ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1)  017310  104401  017334                    TYPE    ,$BIN           ;;GO TYPE THIS BIT
(1)  017314  000241                            CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
(1)  017316  000765                            BR      1$              ;;GO DO THE NEXT BIT
(1)  017320  012601                    2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
(1)  017322  016666  000002  000004            MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  017330  012616                            MOV     (SP)+,(SP)
(1)  017332  000002                            RTI                     ;;RETURN TO USER
(1)  017334     000     000            $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
1347                                   .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)                                    ;;**************************************************************
(1)                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)                                    ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)                                    ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)                                    ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)                                    ;*REPLACED WITH SPACES.
(1)                                    ;*CALL:
(1)                                    ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
(1)                                    ;*      TYPDS                   ;;GO TO THE ROUTINE
```

```
(1)
(1)  017336                          $TYPDS:
(3)  017336  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  017340  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  017342  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  017344  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  017346  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  017350  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  017354  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  017360  100004                          BPL     1$              ;;BR IF INPUT IS POS.
(1)  017362  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  017364  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  017372  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  017374  012703  017552                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  017400  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  017404  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  017406  016001  017542                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  017412  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
(1)  017414  002402                          BLT     4$              ;;BR IF DONE
(1)  017416  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
(1)  017420  000774                          BR      3$
(1)  017422  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  017424  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  017426  001002                          BNE     5$              ;;FALL THROUGH IF 0
(1)  017430  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  017432  100407                          BMI     7$              ;;BR IF YES
(1)  017434  106316                  5$:     ASLB    (SP)            ;;MSD?
(1)  017436  103003                          BCC     6$              ;;BR IF NO
(1)  017440  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  017446  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  017452  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  017456  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  017460  005720                          TST     (R0)+           ;;JUST INCREMENTING
(1)  017462  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
(1)  017466  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
(1)  017470  003002                          BGT     8$              ;;GO TO EXIT
(1)  017472  010502                          MOV     R5,R2           ;;GET THE LSD
(1)  017474  000764                          BR      6$              ;;GO CHANGE TO ASCII
(1)  017476  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  017500  100003                          BPL     9$              ;;BR IF NO
(1)  017502  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  017510  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
(3)  017512  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  017514  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  017516  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  017520  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  017522  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  017524  104401  017552                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  017530  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  017536  012616                          MOV     (SP)+,(SP)
(1)  017540  000002                          RTI                     ;;RETURN TO USER
(1)  017542  023420                  $DTBL:  10000.
(1)  017544  001750                          1000.
(1)  017546  000144                          100.
(1)  017550  000012                          10.
(1)  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)  017562  010046                   $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
  (1)  017564  016600  000002                   MOV     2(SP),R0        ;;GET TRAP ADDRESS
  (1)  017570  005740                           TST     -(R0)           ;;BACKUP BY 2
  (1)  017572  111000                           MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
  (1)  017574  006300                           ASL     R0              ;;POSITION FOR INDEXING
  (1)  017576  016000  017616                   MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
  (1)  017602  000200                           RTS     R0              ;;GO TO ROUTINE
  (1)
  (1)
  (1)                                   ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
  (1)
  (1)  017604  011646                   $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
  (1)  017606  016666  000004  000002           MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
  (1)  017614  000002                           RTI                     ;;RESTORE THE PSW
  (1)
  (3)                                   .SBTTL  TRAP TABLE
  (3)
  (3)                                   ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
  (3)                                   ;*BY THE "TRAP" INSTRUCTION.
  (3)
  (3)                                   ;       ROUTINE
  (3)                                   ;       -------
  (3)  017616  017604                   $TRPAD: .WORD   $TRAP2
  (3)  017620  016232                           $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
  (3)  017622  017060                           $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
  (3)  017624  017034                           $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
  (3)  017626  017074                           $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
  (3)  017630  017336                           $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
  (3)  017632  017262                           $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
  (1)
  (3)  017634  014236                           $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
  (1)
  (3)  017636  014146                           $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
  (3)  017640  014510                           $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
  (3)  017642  014600                           $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
  (3)  017644  015136                           $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
 1350  017646  004112                           TEST    ;;CALL=CHECK    TRAP+14(104414)
 1351  017650  004104                           TESTIT  ;;CALL=CHKIT    TRAP+15(104415)
 1352                                   .EVEN
 1353  017652  000240                           NOP             ;JUST TO FIND THE LAST LOCATION OF THE PROGRAM
 1354           000001                 .END
```

```
ABASE = 170400              17#    57     96     97     98     99    100
ACDW1 = 000000              57
ACDW2 = 000000              57
ACPUOP= 000000              57
ADBUFF  001322              98#   267    326    334    350    375    398    411    424    443    452    462    474
                           697    725    754    797    828    857    885    985   1095   1159*  1163*  1202
ADDW0 = 000000              57
ADDW1 = 000000              57
ADDW10= 000000              57
ADDW11= 000000              57
ADDW12= 000000              57
ADDW13= 000000              57
ADDW14= 000000              57
ADDW15= 000000              57
ADDW2 = 000000              57
ADDW3 = 000000              57
ADDW4 = 000000              57
ADDW5 = 000000              57
ADDW6 = 000000              57
ADDW7 = 000000              57
ADDW8 = 000000              57
ADDW9 = 000000              57
ADEVCT= 000000              57
ADEVM = 000000              57
ADOT    011146             962   1279#
ADST1   001320              97#   1158*  1162*
ADV11C  001372             118#    203    671    692    720   1147*
AENV  = 000000              57
AENVM = 000000              57
AFATAL= 000000              57
AGATST  010274            1258#   1262
AGTST   010304            1113*   1120*  1126*  1260#
AMADR1= 000000              57
AMADR2= 000000              57
AMADR3= 000000              57
AMADR4= 000000              57
AMAMS1= 000000              57
AMAMS2= 000000              57
AMAMS3= 000000              57
AMAMS4= 000000              57
AMSG    011101            1273#
AMSGAD= 000000              57
AMSGLG= 000000              57
AMSGTY= 000000              57
AMTYP1= 000000              57
AMTYP2= 000000              57
AMTYP3= 000000              57
AMTYP4= 000000              57
APASS = 000000              57
APRIOR= 000200              19#    57
APTCSU= 000040            1342   1343#
APTENV= 000001            1339   1342   1343#
APTSIZ= 000200             162   1343#
APTSPO= 000100            1342   1343#
ASKTA   001456             144#   181    186    192    197    201    205    209
ASWREG= 000000              57
```

```
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#
BEGIN0  001522              22     159#    1260
BEGIN2  001530              23     161#    1332
BEGL    002560             263#   1112     1118
BEGST   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    1145
BIT6  = 000100              15#     284     361     365
BIT7  = 000200              15#     328     365     393     404     417     433     449     468
BIT8  = 000400              15#     271
BIT9    001000              15#     112
BPTxxx  000014              15#
BTEX    001402             122#     191*    199     466    1144*
CCHAN   011154             921     1005    1282#
CH      011121             931      956    1275#
CHANL   001362             114#    1190*   1328
CHAN00= 000000              31#     515     547     699
CHAN01= 000001              32#     522     572
CHAN02= 000002              33#     529     597
CHAN03= 000003              34#     536     621     638     646     654     662
CHAN04= 000004              35#     552
CHAN05= 000005              36#     577
CHAN06= 000006              37#     602
CHAN07= 000007              38#     625
CHAN10= 000010              39#     560
CHAN11= 000011              40#     585
```

```
CHAN12= 000012   41#  610
CHAN13= 000013   42#  674  727
CHAN14= 000014   43#  756  802
CHAN15= 000015   44#  763  831
CHAN16= 000016   45#  770  860
CHAN17= 000017   46#  777  888
CHECK = 104414  323  331  342  351  395  409  422  441  471  1350#
CHKIT = 104415  272  280  285  290  294  299  303  308  1351#
CKSWR = 104410 1338 1339 1349#
CMSG    011074 1272#
COMPAR  010100  516  523  530  537  553  561  578  586  603  611  626  639  647
                655  663  675  700  728  757  764  771  778  803  832  861  889
               1216#
CONVRT  007742  514  521  528  535  546  551  559  571  576  584  596  601  609
                620  624  673  698  726  755  762  769  776  801  830  859  887
               1188#
CONVTR  007746  637  645  653  661  934 1016 1189#
CR    = 000015   15#  1342
CRLF  = 000200   15#  174  1342
CRWR    011330  436 1285#
DACA    001324   99#  268  499*  695*  705*  710* 1020* 1032* 1041* 1056* 1060* 1064* 1076*
               1079* 1094* 1160* 1164*
DACB    001326  100#  269  500*  670*  723*  733*  738* 1021* 1033* 1042* 1057* 1061* 1065*
               1077* 1080* 1161* 1165*
DAC0    001344  107#  795* 811*
DAC1    001346  108#  826* 840*
DAC2    001350  109#  855* 869*
DAC3    001352  110#  883* 897*
DDISP = 177570   15#   57  162
DF1     013576   74   81   87   93  1330#
DH1     013367   72 1322#
DH2     013427   91 1323#
DH3     013506   79   85 1324#
DISPLA  001142   57#  162* 1338* 1339*
DISPRE  000174   22#  162
DONE    011133 1243 1278#
DSWR  = 177570   15#   57  162
DT1     013536   73 1327#
DT2     013550   92 1328#
DT3     013566   80   86 1329#
DUMC    010216  367  374  383  386  1239#
DUMW    010144  355  377 1229#
ECHAN   011262 1284#
EMTVEC= 000030   15#  162* 164*
EM1     013247   71 1318#
EM2     013271   78 1319#
EM3     013315   84 1320#
EM4     013342   90 1321#
ERRVEC= 000004   15#  162* 1156* 1338*
FIXONE  007540  175 1156#
GAIN00= 000000   48#  636
GAIN01= 000004   49#  644
GAIN10= 000010   50#  652
GAIN11= 000014   51#  660
GCHAN   011214  925 1009 1283#
GMSG    011106 1274#
```

```
GNS    = ****** U          22     174    1349    1350    1351
GTSWR  = 104407            174    1349#
HEAD2    013176           1317#
HEAD5    011475           1288#
HT     = 000011            15#    1342
IOTST1   006340            249     916#
IOTST2   006514            250     948#
IOTST3   006716            251    1004#
IOTST4   007024            252    1027#
IOTST5   007114            253    1051#
IOTST6   007202            254    1072#
IOTST7   007250            255    1086#
IOTVEC=  000020             15#    162*    163*
KWAD     001374            119#    183     402     447    1132*
KWBPR    001342            106#    406*    419*    450*
KWCSR    001340            105#    407*    420*    453*     454
KWEX     001376            120#    188     415    1135*
LF     = 000012             15#    1342
MAEX     001400            121#    185*    194     429
METST    011515           1233    1289#
MSADV    012201            202    1300#
MSBTEX   012106            198    1298#
MSGNEX   012050            435    1296#
MSG70    012377            215    1305#
MSG71    011431            216    1287#
MSIO1    010430            917    1265#
MSIO2    010506            948    1266#
MSIO3    010546           1004    1267#
MSIO4    010611           1027    1268#
MSIO5    010664           1051    1269#
MSIO6    010731           1072    1270#
MSIO7    011022           1086    1271#
MSKWAD   011537            147     182    1290#
MSKWEX   011621            187    1292#
MSMAEX   011730            193    1294#
MSTC1    012230            206    1301#
MSTC2    012307            210    1303#
OKCHAR   002464            226     237#
ONAD     011370           1235    1286#
OTHER    010076            636*    644*    652*    660*    927*    928*    929*    930*   1011*   1012*   1013*   1014*   1188*
                          1192    1213#
PIRQ   = 177772             15#
PIRQVE=  000240             15#
PRO    = 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#
```

```
RDLIN = 104412          148     218     437     1058    1062    1066    1334    1349#
RDOCT = 104413          922     926     1006    1010    1349#
RESVEC= 000010          15#
RETERR  004134          489     491#
RETURN  001440          138#    1198
RST     010246          1247#
SLASH   011152          1281#
SPACE   011124          936     984     1102    1276#
SPREAD  001364          115#    1217*   1223    1328
STACK = 001100          15#     162
STALL   001426          132#    329     408     421     470     1078    1081
STKLMT= 177774          15#
STREG   001316          96#     266     312*    315     320*    327*    330*    338*    339     347*    348     361*    362
                        364     379*    380     337*    390*    391     394*    399*    405*    412*    418*    425*    434*
                        444*    451*    456*    458     463*    469*    473*    486*    487     696*    724*    753*    796*
                        827*    856*    884*    916*    979*    980*    981     1091*   1092    1157*   1194*   1200*   1236
                        1327    1328    1329
SWR     001140          57#     162*    174     1332*   1336*   1338    1339
SWREG   000176          22#     162     174     1332
SW0   = 000001          15#
SW00  = 000001          15#
SW01  = 000002          15#
SW02  = 000004          15#
SW03  = 000010          15#
SW04  = 000020          15#
SW05  = 000040          15#
SW06  = 000100          15#
SW07  = 000200          15#
SW08  = 000400          15#
SW09  = 001000          15#
SW1   = 000002          15#
SW10  = 002000          15#
SW11  = 004000          15#
SW12  = 010000          15#
SW13  = 020000          15#
SW14  = 040000          15#
SW15  = 100000          15#
SW2   = 000004          15#
SW3   = 000010          15#
SW4   = 000020          15#
SW5   = 000040          15#
SW6   = 000100          15#
SW7   = 000200          15#
SW8   = 000400          15#
SW9   = 001000          15#
TBITVE= 000014          15#
TC1     001366          116#    207     512     544     569     594     618     634     690     718     1138*
TC2     001370          117#    211     751     792     823     852     880     1141*
TEMP    001360          113#    159*    161*    176     548     549     573     574     598     599     622     937     1019*
                        1020    1021    1193*   1204*   1207*   1208*   1209*   1210*   1218
TEST    004112          487#    1350
TESTIT  004104          486#    1351
TKVEC = 000060          15#     1332*
TPVEC = 000064          15#
TRAPVE= 000034          15#     162*    165*
TRTVEC= 000014          15#
```

```
TRYAG   002406            218#    234
TST1    002560            264#
TST10   002746            301#
TST11   002762            306#
TST12   002776            310#
TST13   003042            316     319#
TST14   003074            325#
TST15   003136            336#
TST16   003200            346#
TST17   003224            354#
TST2    002614            270#
TST20   003352            368     376#    384
TST21   003436            389#
TST22   003504            401#
TST23   003564            403     414#
TST24   003644            416     428#
TST25   003732            430     432     446#
TST26   004034            448     465#
TST27   004100            467     477#
TST3    002644            278#
TST30   004136            497#
TST31   004220            511#
TST32   004336            513     543#
TST33   004436            545     568#
TST34   004536            570     593#
TST35   004636            595     617#
TST36   004710            619     633#
TST37   005056            635     669#
TST4    002660            282#
TST40   005122            672     686#
TST41   005252            691     693     714#
TST42   005402            719     721     748#
TST43   005532            752     789#
TST44   005674            793     814     820#
TST45   006032            824     843     849#
TST46   006170            853     872     878#
TST47   006326            881     900     903#
TST5    002702            288#
TST6    002716            292#
TST7    002732            297#
TYPBN = 104406            1349#
TYPDS = 104405            1262    1349#
TYPE  = 104401             145     146     174     215     216     233     435     436     917     921     925     931     936
                           943     948     953     956     962     968     974     984     998    1004    1005    1009    1027
                          1051    1072    1086    1088    1102    1233    1235    1243    1262    1332    1336    1339    1340
                          1342    1345    1346    1347    1349#
TYPOC = 104402            1332    1340    1349#
TYPON = 104404            1349#
TYPOS = 104403             932     937     959     986    1096    1234    1236    1349#
UNEXP   001404             124#    283     370
VECTOR  001330             101#    283*    359*    370*   1166*   1167*   1168    1170    1172    1198*
VECTR1  001332             102#    360*   1168*   1169*
VECTR2  001334             103#    378*   1170*   1171*
VECTR3  001336             104#   1172*   1173*
VWRAP   001354             111#    518     525     532     539     641     649     657     665     677     702     730     805
                           834     863     891
```

```
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
```

```
$GET42  010364          1262#
$GTSWR  014236          1332#   1349
$HD   = 000000            14
$HIBTS  001000            56#
$HIOCT  015236          1334#*
$ICNT   001104            57#   1338*
$ILLUP  015400          1336#
$INTAG  001135            57#   1332*
$ITEMB  001114            57#   1339*   1340
$LF     001172            57#   1332    1339    1342
$LFLG   017031          1343#*
$LPADR  001106            57#    162*    264*    502*   1232*   1242*   1338*
$LPERR  001110            57#    162*    501*   1231*   1241*   1338*   1339
$MADR1  001226            57#
$MADR2  001232            57#
$MADR3  001236            57#
$MADR4  001242            57#
$MAIL   001174            56      57#    162     174    1338    1339    1342
$MAMS1  001224            57#
$MAMS2  001230            57#
$MAMS3  001234            57#
$MAMS4  001240            57#
$MBADR  001002            56#
$MFLG   017030          1343#*
$MNEW   015124          1332#
$MSGAD  001210            57#   1343*
$MSGLG  001212            57#   1343*
$MSGTY  001174            57#   1343*
$MSWR   015113          1332#
$MTYP1  001225            57#
$MTYP2  001231            57#
$MTYP3  001235            57#
$MTYP4  001241            57#
$MXCNT  015674          1338#
$NULL   001154            57#   1342
$NWTST= 000001           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#
$OCNT   017256          1345#*
$OMODE  017260          1345#*
$OVER   015660          1338#
$PASS   001202            57#   162*    431     1229    1239    1262*   1338
$PASTM  001006            56#
$POWER  015406          1336#
$PWRDN  015240           162    1336#
$PWRMG  015374          1336#
$PWRUP  015312          1336#
$QUES   001170            57#   1332    1339    1342
$RDCHR  014510          1332#   1349
$RDDEC= ****** U        1349
$RDLIN  014600          1332#   1349
$RDOCT  015136          1334#   1349
$RDSZ = 000040          1332#
$RTNAD  010406          1262#
$R2A  = ****** U        1349
$SAVRE= ****** U        1349
```

```
$SAVR6  015404  1336#*
$SCOPE  015416  162  1338#
$SETUP= 000137  58#  162  174  1262  1332  1338  1339
$STUP = 177777  58#
$SVLAD  015624  1338#
$SVPC = 000144  54#
$SWR  = 167400  6#  14  16  57  162  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  1336  1338  1339
$SWREG  001216  57#  162
$SWRMK= 000000  16  1338
$TESTN  001200  57#  1338*
$TIMES  001160  57#  162*  310*  319*  336*  497*  511*  543*  568*  593*  617*  633*  669*
                686*  714*  748*  789*  820*  849*  878*  903*  1262*  1338*
$TKB    001146  57#  1332  1342
$TKCNT  013600  1332#*
$TKINT  013646  171  1332#
$TKQEN= 013646  1332#
$TKQIN  013602  1332#*
$TKQOU  013604  1332#*
$TKQSR  013606  1332#
$TKS    001144  57#  219*  314*  322*  344*  1248*  1258*  1332*  1342
$TKSRV  013716  1332#
$TN   = 000050  7#  14  264#  265  270#  278#  282#  288#  292#  297#  301#  306#  310#
                316  319#  325#  336#  346#  354#  355  368  376#  377  389#  401#  403
                414#  416  428#  430  432  446#  448  465#  467  477#  497#  511#  513
                543#  545  568#  570  593#  595  617#  619  633#  635  669#  672  686#
                691  693  714#  719  721  748#  752  789#  793  814  820#  824  843
                849#  853  872  878#  881  900  903#
$TPB    001152  57#  1342*
$TPFLG  001157  57#  1342
$TPS    001150  57#  1342
$TRAP   017562  162  1349#
$TRAP2  017604  1349#
$TRP  = 000016  1349#  1350#  1351#
$TRPAD  017616  1349#
$TSTM   001004  56#
$TSTNM  001102  57#  265*  497*  1262*  1338*  1339
$TTYIN  015034  1332#
$TYPBN  017262  1346#  1349
$TYPDS  017336  1347#  1349
$TYPE   016232  167*  168*  169*  170*  1342#  1343  1349
$TYPEC  016444  1332  1342#
$TYPEX  016564  1342#
$TYPOC  017060  1345#  1349
$TYPON  017074  1345#  1349
$TYPOS  017034  1345#  1349
$UNIT   001206  57#
$UNITM  001010  56#
$USWR   001220  57#
$VECT1  001244  57#  1166
$VECT2  001246  57#
$XOFF = 000023  1342
$XON  = 000021  1332  1342
$XTSTR  015430  1338#
```

```
$$GET4= 000000           1262#
$OFILL  017257           1345#*
$40CAT= ****** U         1338    1339
.       = 017654           22#     25#     28#     54#     55#     56#     57#     162     1262    1332#   1336    1338    1339
                         1340#   1342    1343#   1347#
.$ASTA= ****** U         1343
.$X     = 001000           56#
```

```
COMMEN    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#
NEWTST    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#
```

```
.$ACT1      11#     54
.$APTB      11#     57#
.$APTH      11#     56
.$APTY      11#   1343
.$CATC       8#     22
.$CMTA       8#     57
.$EOP        8#   1262
.$ERRO       8#   1339
.$ERRT      10#   1340
.$PARM       9#
.$POWE       9#   1336
.$RAND      11#
.$RDOC      11#   1334
.$READ       9#   1332
.$SAVE       9#
.$SCOP       9#   1338
.$SPAC      10#
.$SWDO      10#
.$TRAP      10#   1349
.$TYPB       9#   1346
.$TYPD      11#   1347
.$TYPE      10#   1342
.$TYPO       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)
```