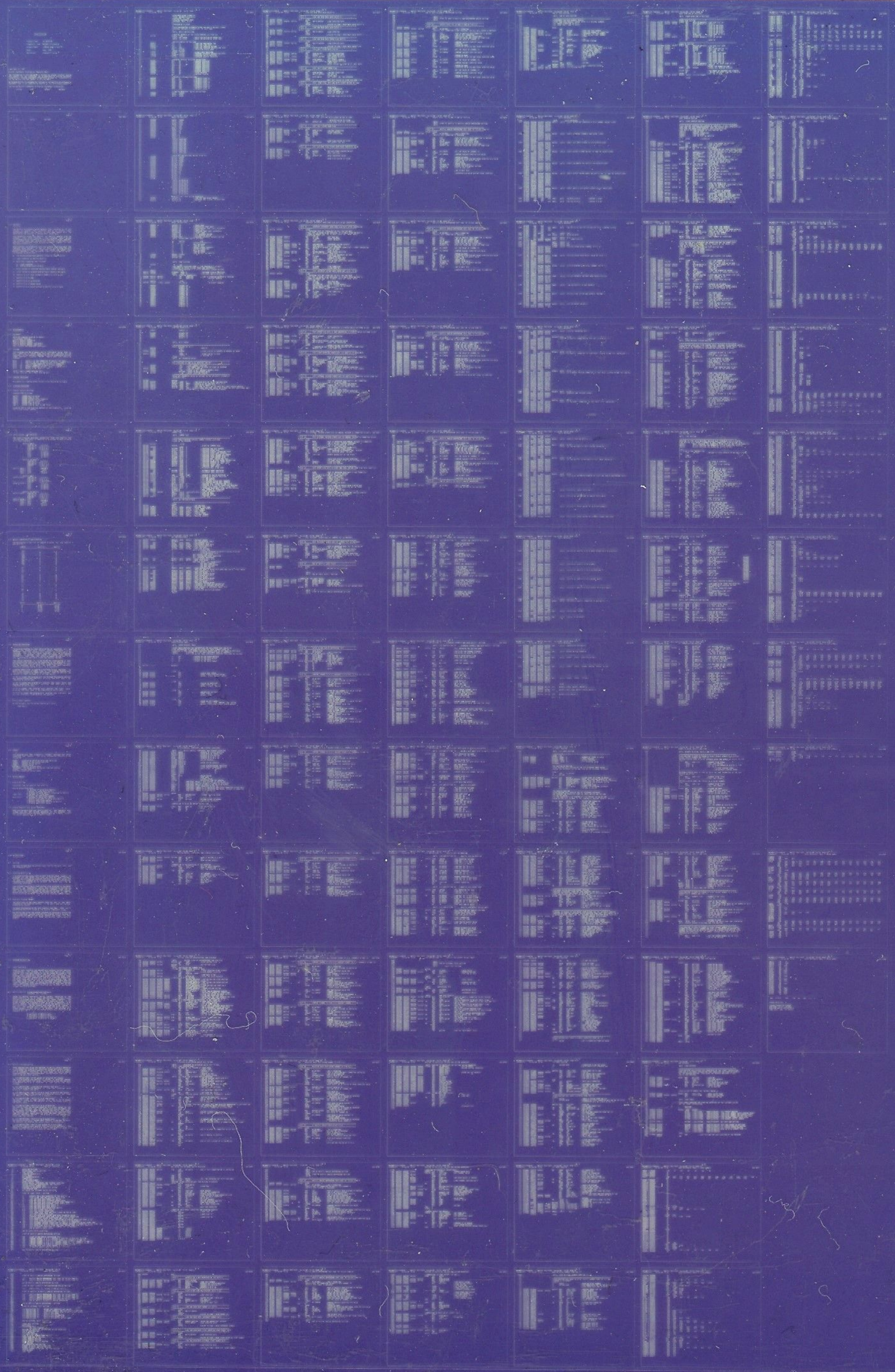
AXV11-C,
ADV11-C

AXV11-C/ADV11-C
CVAXAA0

AH-S895A-MC
FICHE 1 OF 1

OCT 1981



IDENTIFICATION

Product Code: AC-S893A-MC

Diagnostic Code: MAINDEC-11-CVAXA-A

Product Name: CVAXAA0 AXV11-C/ADV11-C

Date: Aug. 1981

Maintainer: Diagnostic Group

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

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

DEC DECUS DECTAPE

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

PDP11/03 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 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	D2-D3	D2-D3
P	P1-P2	P1-P2

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.

```
      1                          ;DEVELOPED USING SYSMAC.C4
     14                          .TITLE  MAINDEC-11-CVAXA-A
    (1)                          ;*COPYRIGHT (C) 1981
    (1)                          ;*DIGITAL EQUIPMENT CORP.
    (1)                          ;*MAYNARD, MASS. 01754
    (1)                          ;*
    (1)                          ;*PROGRAM BY R.SHOOP
    (1)                          ;*
    (1)                          ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
    (1)                          ;*PACKAGE (MAINDEC-11-DZQAC-C4), 31 JULY 1980.
    (1)                          ;*
     15                          .SBTTL  BASIC DEFINITIONS
    (1)
    (1)                          ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
    (1)        001100            STACK=  1100
    (1)                          .EQUIV  EMT,ERROR        ;;BASIC DEFINITION OF ERROR CALL
    (1)                          .EQUIV  IOT,SCOPE        ;;BASIC DEFINITION OF SCOPE CALL
    (1)
    (1)                          ;*MISCELLANEOUS DEFINITIONS
    (1)        000011            HT=     11               ;;CODE FOR HORIZONTAL TAB
    (1)        000012            LF=     12               ;;CODE FOR LINE FEED
    (1)        000015            CR=     15               ;;CODE FOR CARRIAGE RETURN
    (1)        000200            CRLF=   200              ;;CODE FOR CARRIAGE RETURN-LINE FEED
    (1)        177776            PS=     177776           ;;PROCESSOR STATUS WORD
    (1)                          .EQUIV  PS,PSW
    (1)        177774            STKLMT= 177774           ;;STACK LIMIT REGISTER
    (1)        177772            PIRQ=   177772           ;;PROGRAM INTERRUPT REQUEST REGISTER
    (1)        177570            DSWR=   177570           ;;HARDWARE SWITCH REGISTER
    (1)        177570            DDISP=  177570           ;;HARDWARE DISPLAY REGISTER
    (1)
    (1)                          ;*GENERAL PURPOSE REGISTER DEFINITIONS
    (1)        000000            R0=     %0               ;;GENERAL REGISTER
    (1)        000001            R1=     %1               ;;GENERAL REGISTER
    (1)        000002            R2=     %2               ;;GENERAL REGISTER
    (1)        000003            R3=     %3               ;;GENERAL REGISTER
    (1)        000004            R4=     %4               ;;GENERAL REGISTER
    (1)        000005            R5=     %5               ;;GENERAL REGISTER
    (1)        000006            R6=     %6               ;;GENERAL REGISTER
    (1)        000007            R7=     %7               ;;GENERAL REGISTER
    (1)        000006            SP=     %6               ;;STACK POINTER
    (1)        000007            PC=     %7               ;;PROGRAM COUNTER
    (1)
    (1)                          ;*PRIORITY LEVEL DEFINITIONS
    (1)        000000            PR0=    0                ;;PRIORITY LEVEL 0
    (1)        000040            PR1=    40               ;;PRIORITY LEVEL 1
    (1)        000100            PR2=    100              ;;PRIORITY LEVEL 2
    (1)        000140            PR3=    140              ;;PRIORITY LEVEL 3
    (1)        000200            PR4=    200              ;;PRIORITY LEVEL 4
    (1)        000240            PR5=    240              ;;PRIORITY LEVEL 5
    (1)        000300            PR6=    300              ;;PRIORITY LEVEL 6
    (1)        000340            PR7=    340              ;;PRIORITY LEVEL 7
    (1)
    (1)                          ;*"SWITCH REGISTER" SWITCH DEFINITIONS
    (1)        100000            SW15=   100000
    (1)        040000            SW14=   40000
    (1)        020000            SW13=   20000
    (1)           010000                    SW12=   10000
    (1)           004000                    SW11=   4000
    (1)           002000                    SW10=   2000
    (1)           001000                    SW09=   1000
    (1)           000400                    SW08=   400
    (1)           000200                    SW07=   200
    (1)           000100                    SW06=   100
    (1)           000040                    SW05=   40
    (1)           000020                    SW04=   20
    (1)           000010                    SW03=   10
    (1)           000004                    SW02=   4
    (1)           000002                    SW01=   2
    (1)           000001                    SW00=   1
    (1)                                     .EQUIV  SW09,SW9
    (1)                                     .EQUIV  SW08,SW8
    (1)                                     .EQUIV  SW07,SW7
    (1)                                     .EQUIV  SW06,SW6
    (1)                                     .EQUIV  SW05,SW5
    (1)                                     .EQUIV  SW04,SW4
    (1)                                     .EQUIV  SW03,SW3
    (1)                                     .EQUIV  SW02,SW2
    (1)                                     .EQUIV  SW01,SW1
    (1)                                     .EQUIV  SW00,SW0
    (1)
    (1)                                     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
    (1)           100000                    BIT15=  100000
    (1)           040000                    BIT14=  40000
    (1)           020000                    BIT13=  20000
    (1)           010000                    BIT12=  10000
    (1)           004000                    BIT11=  4000
    (1)           002000                    BIT10=  2000
    (1)           001000                    BIT09=  1000
    (1)           000400                    BIT08=  400
    (1)           000200                    BIT07=  200
    (1)           000100                    BIT06=  100
    (1)           000040                    BIT05=  40
    (1)           000020                    BIT04=  20
    (1)           000010                    BIT03=  10
    (1)           000004                    BIT02=  4
    (1)           000002                    BIT01=  2
    (1)           000001                    BIT00=  1
    (1)                                     .EQUIV  BIT09,BIT9
    (1)                                     .EQUIV  BIT08,BIT8
    (1)                                     .EQUIV  BIT07,BIT7
    (1)                                     .EQUIV  BIT06,BIT6
    (1)                                     .EQUIV  BIT05,BIT5
    (1)                                     .EQUIV  BIT04,BIT4
    (1)                                     .EQUIV  BIT03,BIT3
    (1)                                     .EQUIV  BIT02,BIT2
    (1)                                     .EQUIV  BIT01,BIT1
    (1)                                     .EQUIV  BIT00,BIT0
    (1)
    (1)                                     ;*BASIC "CPU" TRAP VECTOR ADDRESSES
    (1)           000004                    ERRVEC= 4                     ;;TIME OUT AND OTHER ERRORS
    (1)           000010                    RESVEC= 10                    ;;RESERVED AND ILLEGAL INSTRUCTIONS
    (1)           000014                    TBITVEC=14                    ;;"T" BIT
```

```
   (1)          000014                      TRTVEC= 14              ;;TRACE TRAP
   (1)          000014                      BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
   (1)          000020                      IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
   (1)          000024                      PWRVEC= 24              ;;POWER FAIL
   (1)          000030                      EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
   (1)          000034                      TRAPVEC=34              ;;"TRAP" TRAP
   (1)          000060                      TKVEC=  60              ;;TTY KEYBOARD VECTOR
   (1)          000064                      TPVEC=  64              ;;TTY PRINTER VECTOR
   (1)          000240                      PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
    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          000000                              CHAN00= 00
    29          000001                              CHAN01= 01
    30          000002                              CHAN02= 02
    31          000003                              CHAN03= 03
    32          000004                              CHAN04= 04
    33          000005                              CHAN05= 05
    34          000006                              CHAN06= 06
    35          000007                              CHAN07= 07
    36          000010                              CHAN10= 10
    37          000011                              CHAN11= 11
    38          000012                              CHAN12= 12
    39          000013                              CHAN13= 13
    40          000014                              CHAN14= 14
    41          000015                              CHAN15= 15
    42          000016                              CHAN16= 16
    43          000017                       CHAN17= 17
    44
    45          000000                       GAIN00= 00
    46          000004                       GAIN01= 04
    47          000010                       GAIN10= 10
    48          000014                       GAIN11= 14
    49
    50
    51                             .SBTTL  ACT11 HOOKS
   (1)
   (2)                             ;;*************************************************************
   (1)                             ;HOOKS REQUIRED BY ACT11
   (1)          000106                       $SVPC=.                  ;SAVE PC
   (1)          000046                       .=46
   (1)  000046  010342                       $ENDAD                   ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
   (1)          000052                       .=52
   (1)  000052  000000                       .WORD   0                ;;2)SET LOC.52 TO ZERO
   (1)          000106                       .=$SVPC                  ;; RESTORE PC
    52          001000             .=1000
    53                             .SBTTL  APT PARAMETER BLOCK
   (1)
   (2)                             ;;*************************************************************
   (1)                             ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
   (2)                             ;;*************************************************************
   (1)          001000                       .$X=.    ;;SAVE CURRENT LOCATION
   (1)          000024                       .=24     ;;SET POWER FAIL TO POINT TO START OF PROGRAM
   (1)  000024  000200                       200      ;;FOR APT START UP
   (1)          000044                       .=44     ;;POINT TO APT INDIRECT ADDRESS PNTR.
   (1)  000044  001000                       $APTHDR ;;POINT TO APT HEADER BLOCK
   (1)          001000                       .=.$X    ;;RESET LOCATION COUNTER
   (2)                             ;;*************************************************************
   (1)                             ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
   (1)                             ;INTERFACE SPEC.
   (1)
   (1)  001000                     $APTHD:
   (1)  001000  000000             $HIBTS: .WORD   0        ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
   (1)  001002  001174             $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
   (1)  001004  000550             $TSTM:  .WORD   360.     ;;RUN TIM OF LONGEST TEST
   (1)  001006  000132             $PASTM: .WORD   90.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
   (1)  001010  000550             $UNITM: .WORD   360.     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
   (1)  001012  000031                     .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
   54                                   .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             $DEVM:  .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                        $ERRTB:
    56
    57
    58
    67                                ;ITEM   1
    68  001256  013215                        EM1                     ;STATUS REG. ERROR
    69  001260  013335                        DH1                     ;ERRPC STREG EXPECTED ACTUAL
    70  001262  013504                        DT1                     ;$ERRPC, STREG, $GDDAT, $BDDAT
    71  001264  013544                        DF1
    72
    73
    74                                ;ITEM   2
    75  001266  013237                        EM2                     ;FAILED TO INTERRUPT
    76  001270  013454                        DH3                     ;ERRPC STREG ACTUAL
    77  001272  013534                        DT3                     ;$ERRPC, STREG,  $BDDAT
    78  001274  013544                        DF1
    79
    80                                ;ITEM   3
    81  001276  013263                        EM3                     ;UNEXPECTED INTERRUPT
    82  001300  013454                        DH3                     ;ERRPC STREG
    83  001302  013534                        DT3                     ;$ERRPC, STREG
    84  001304  013544                        DF1
    85
    86                                ;ITEM   4
    87  001306  013310                        EM4                     ;ERROR ON A/D CHANNEL
    88  001310  013375                        DH2                     ;ERRPC  STREG  CHAN  NOMINAL  TOL  ACTUAL
    89  001312  013516                        DT2                     ;$ERRPC,STREG,CHANL,$GDDAT,SPREAD,$BDDAT
    90  001314  013544                        DF1
```

```
  92                                       .SBTTL       MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
  93  001316  170400                       STREG:  ABASE                          ;ADDRESS OF STATUS REGISTER
  94  001320  170401                       ADST1:  ABASE+1                        ;UPPER BYTE OF STATUS REG.
  95  001322  170402                       ADBUFF: ABASE+2                        ;ADDRESS OF A/D BUFFER
  96  001324  170404                       DACA:   ABASE+4                        ;ADDRESS OF D TO A ''A''
  97  001326  170406                       DACB:   ABASE+6                        ;ADDRESS OF D TO A 'B''
  98  001330  000400                       VECTOR: AVECT1                         ;VECTOR ADDRESS
  99  001332  000402                       VECTR1: AVECT1+2
 100  001334  000404                       VECTR2: AVECT1+4                       ;ERROR VECTOR ADDRESS
 101  001336  000406                       VECTR3: AVECT1+6
 102  001340  170420                       KWCSR:  170420                         ;CLOCK STATUS/CONTROL REGISTER
 103  001342  170422                       KWBPR:  170422                         ;CLOCK PRESET/COUNTER REGISTER
 104  001344  170440                       DAC0:   170440                         ;AAV11-C DAC ''A'' ADDRESS
 105  001346  170442                       DAC1:   170442                         ;                'B''
 106  001350  170444                       DAC2:   170444                         ;                ''C''
 107  001352  170446                       DAC3:   170446                         ;                'D''
 108  001354  000020                       VWRAP:  20
 109  001356  001000                       BARF:   BIT9                           ;DELAY FACTOR
 110  001360  000000                       TEMP:   0                              ;WORK AREA
 111  001362  000000                       CHANL:  0                              ;CHANNEL VALUE
 112  001364  000000                       SPREAD: 0                              ;DEVIATION FROM THE NOMINAL
 113  001366  000000                       TC1:    0     ;NON-ZERO, AXV11-C TEST FIXTURE IS INSTALLED
 114  001370  000000                       TC2:    0     ;NON-ZERO, AAV11-C TO AXV11-C CABLE IN INSTALLED
 115  001372  000000                       ADV11C: 0     ;NON-ZERO, MODULE IS ADV11-C (NO DAC'S ON BOARD)
 116  001374  000000                       KWAD:   0     ;NON-ZERO, CLOCK CONNECTED TO RTC IN
 117  001376  000000                       KWEX:   0     ;NON-ZERO, JUMPER F2 IS INSTALLED AND CLOCK CONNECTED TO EXT TRIG
 118  001400  000000                       MAEX:   0     ;NON-ZERO, JUMPER F2 IS INSTALLED AND MANUAL TRIGGER IS CONNECTED
 119  001402  000000                       BTEX:   0     ;NON-ZERO, JUMPER F1 IS INSTALLED
 120
 121  001404                               UNEXP:
 (1)  001404  012737  001420  001162               MOV     #1$,$ESCAPE        ;;ESCAPE TO 1$ ON ERROR
 122  001412  005237  001103                       INC     $ERFLG
 123  001416  104003                               ERROR   3
 124  001420  005037  001162               1$:     CLR     $ESCAPE            ;RETURN ESCAPE TO NORMAL
 125  001424  000002                               RTI                        ;UNEXPECTED INTERRUPT
 126
 127                                       ;SUBROUTINE TO DELAY AN AMOUNT OF CPU TIME
 128
 129  001426  013700  001356               STALL:  MOV     BARF,R0            ;GET DELAY FACTOR
 130  001432  005300                       1$:     DEC     R0                 ;DELAY
 131  001434  001376                               BNE     1$
 132  001436  000207                               RTS     PC                 ;EXIT
```

```

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

                                        ;SUBROUTINE TO ASK QUESTIONS OF THE OPERATOR
001456  012537  001470                  ASKTA:  MOV     (R5)+,10$       ;GET THE ASCII POINTER
001462  104401  001171                          TYPE    ,$CRLF          ;MAKE A FRESH LINE
001466  104401                                  TYPE                    ;TELL THE OPERATOR A MESSAGE
001470  011505                          10$:    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
```

```
   155                                             .SBTTL          INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
   156  001522  005037  001360                     BEGIN0: CLR     TEMP                    ;CLEAR RESTART FLAG
   157  001526  000402                                     BR      BEGST
   158  001530  005237  001360                     BEGIN2: INC     TEMP                    ;SET RESTART FLAG
   159  001534                                     BEGST:
   (1)                                             .SBTTL  INITIALIZE THE COMMON TAGS
   (1)                                             ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
   (1)  001534  012706  001100                             MOV     #$CMTAG,R6              ;;FIRST LOCATION TO BE CLEARED
   (1)  001540  005026                                     CLR     (R6)+                   ;;CLEAR MEMORY LOCATION
   (1)  001542  022706  001140                             CMP     #SWR,R6 ;;DONE?
   (1)  001546  001374                                     BNE     .-6                     ;;LOOP BACK IF NO
   (1)  001550  012706  001100                             MOV     #STACK,SP               ;;SETUP THE STACK POINTER
   (1)                                             ;;INITIALIZE A FEW VECTORS
   (1)  001554  012737  015352  000020                     MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
   (1)  001562  012737  000340  000022                     MOV     #340,@#IOTVEC+2 ;;LEVEL 7
   (1)  001570  012737  015632  000030                     MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
   (1)  001576  012737  000340  000032                     MOV     #340,@#EMTVEC+2 ;;LEVEL 7
   (1)  001604  012737  017516  000034                     MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
   (1)  001612  012737  000340  000036                     MOV     #340,@#TRAPVEC+2;LEVEL 7
   (1)  001620  012737  015174  000024                     MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
   (1)  001626  012737  000340  000026                     MOV     #340,@#PWRVEC+2 ;;LEVEL 7
   (1)  001634  013737  010310  010302                     MOV     $ENDCT,$EOPCT    ;;SETUP END-OF-PROGRAM COUNTER
   (1)  001642  005037  001160                             CLR     $TIMES           ;;INITIALIZE NUMBER OF ITERATIONS
   (1)  001646  005037  001162                             CLR     $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
   (1)  001652  112737  000001  001115                     MOVB    #1,$ERMAX        ;;ALLOW ONE ERROR PER TEST
   (1)  001660  012737  001660  001106                     MOV     #.,$LPADR        ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
   (1)  001666  012737  001666  001110                     MOV     #.,$LPERR        ;;SETUP THE ERROR LOOP ADDRESS
   (2)                                             ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   (2)                                             ;;EQUAL TO A ''-1'', SETUP FOR A SOFTWARE SWITCH REGISTER.
   (2)  001674  013746  000004                             MOV     @#ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
   (2)  001700  012737  001734  000004                     MOV     #64$,@#ERRVEC    ;;SET UP ERROR VECTOR
   (2)  001706  012737  177570  001140                     MOV     #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
   (2)  001714  012737  177570  001142                     MOV     #DDISP,DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
   (2)  001722  022777  177777  177210                     CMP     #-1,@SWR         ;;TRY TO REFERENCE HARDWARE SWR
   (2)  001730  001012                                     BNE     66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
   (2)                                                                              ;;AND  THE HARDWARE SWR IS NOT = -1
   (2)  001732  000403                                     BR      65$              ;;BRANCH IF NO TIMEOUT
   (2)  001734  012716  001742                     64$:    MOV     #65$,(SP)        ;;SET UP FOR TRAP RETURN
   (2)  001740  000002                                     RTI
   (2)  001742  012737  000176  001140             65$:    MOV     #SWREG,SWR       ;;POINT TO SOFTWARE SWR
   (2)  001750  012737  000174  001142                     MOV     #DISPREG,DISPLAY
   (2)  001756  012637  000004                     66$:    MOV     (SP)+,@#ERRVEC   ;;RESTORE ERROR VECTOR
   (1)
   (2)  001762  005037  001202                             CLR     $PASS            ;;CLEAR PASS COUNT
   (2)  001766  132737  000200  001215                     BITB    #APTSIZE,$ENVM   ;;TEST USER SIZE UNDER APT
   (2)  001774  001403                                     BEQ     67$              ;;YES,USE NON-APT SWITCH
   (2)  001776  012737  001216  001140                     MOV     #$SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
   (2)  002004                                     67$:
   160  002004  012737  005046  016166                     MOV     #5046,$TYPE      ;A WAY TO LOWER
   161  002012  012737  012746  016170                     MOV     #12746,$TYPE+2   ; PS FOR
   162  002020  012737  016200  016172                     MOV     #$TYPE+12,$TYPE+4
   163  002026  012737  000002  016174                     MOV     #RTI,$TYPE+6     ;    TTY OUTPUT
   164  002034  004737  013614                             JSR     PC,$TKINT        ;INIT THE CONSOLE VECTORS
```

```
  166                                          .SBTTL         DIALOGUE TO DETERMINE WHICH TEST TO RUN
  167                                          .SBTTL   TYPE PROGRAM NAME
  (1)                                          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
  (1)   002040  005227  177777                          INC     #-1                 ;;FIRST TIME?
  (1)   002044  001053                                  BNE     68$                 ;;BRANCH IF NO
  (1)   002046  022737  010342  000042                  CMP     #$ENDAD,@#42        ;;ACT-11?
  (1)   002054  001447                                  BEQ     68$                 ;;BRANCH IF YES
  (1)   002056  104401  002124                          TYPE    ,69$                ;;TYPE ASCIZ STRING
  (2)                                          .SBTTL   GET VALUE FOR SOFTWARE SWITCH REGISTER
  (2)   002062  005737  000042                          TST     @#42                ;;ARE WE RUNNING UNDER XXDP/ACT?
  (2)   002066  001012                                  BNE     70$                 ;;BRANCH IF YES
  (2)   002070  123727  001214  000001                  CMPB    $ENV,#1             ;;ARE WE RUNNING UNDER APT?
  (2)   002076  001406                                  BEQ     70$                 ;;BRANCH IF YES
  (2)   002100  023727  001140  000176                  CMP     SWR,#SWREG          ;;SOFTWARE SWITCH REG SELECTED?
  (2)   002106  001005                                  BNE     71$                 ;;BRANCH IF NO
  (2)   002110  104407                                  GTSWR                       ;;GET SOFT-SWR SETTINGS
  (2)   002112  000403                                  BR      71$
  (2)   002114  112737  000001  001134  70$:            MOVB    #1,$AUTOB           ;;SET AUTO-MODE INDICATOR
  (2)   002122                                 71$:
  (1)   002122  000424                                  BR      68$                 ;;GET OVER THE ASCIZ
  (1)                                          ;;69$:   .ASCIZ  <CRLF># CVAXAA  AXV11-C/ADV11-C DIAGNOSTIC  #<CRLF>
  (1)   002174                                 68$:
  168   002174  004737  007506                          JSR     PC,FIXONE           ;INITIALIZE ADDRESSES
  169   002200  005737  001360                 77$:     TST     TEMP                ;ARE WE RESTARTING THE PROGRAM
  170   002204  001062                                  BNE     40$                 ;BR IF YES
  171   002206  005737  001134                          TST     $AUTOB              ;IS IT CHAINED?
  172   002212  001402                                  BEQ     1$
  173   002214  000137  007360                          JMP     BEGIND  ;RUN ONLY THE LOGIC TEST AND SELECTED WRAPAROUND IF APT/XXDP CHA
  174   002220  004537  001456                 1$:      JSR     R5,ASKTA            ;ASK OPERATOR ABOUT DIFFERENT CONFIG.
  175   002224  011505                                  MSKWAD                      ;IS KWV11-C CONNECTED TO CLOCK START
  176   002226  001374                                  KWAD                        ;
  177   002230  000240                                  NOP
  178   002232  005037  001400                          CLR     MAEX                ;ENSURE CLEARED FLAG
  179   002236  004537  001456                          JSR     R5,ASKTA            ;ASK IF KWV11-C CONNECTED TO EXT. START
  180   002242  011567                                  MSKWEX
  181   002244  001376                                  KWEX
  182   002246  000403                                  BR      2$
  183   002250  000415                                  BR      4$                  ;IF ANSWER WAS YES, BYPASS NEXT QUESTION
  184   002252  005037  001402                          CLR     BTEX                ;ENSURE CLEARED FLAG
  185   002256  004537  001456                 2$:      JSR     R5,ASKTA            ;ASK IF MANUAL TRIGGER IS CONNECTED TO EXT. START
  186   002262  011676                                  MSMAEX
  187   002264  001400                                  MAEX
  188   002266  000401                                  BR      3$
  189   002270  000405                                  BR      4$
  190   002272  004537  001456                 3$:      JSR     R5,ASKTA            ;ASK IF B EVENT IS CONNECTED TO EXT TRIG
  191   002276  012054                                  MSBTEX
  192   002300  001402                                  BTEX
  193   002302  000240                                  NOP
  194   002304  004537  001456                 4$:      JSR     R5,ASKTA            ;ASK IF MODULE IS ADV11-C
  195   002310  012147                                  MSADV
  196   002312  001372                                  ADV11C
  197   002314  000240                                  NOP
  198   002316  004537  001456                 10$:     JSR     R5,ASKTA            ;ASK IF TEST FIXTURE #1 IS INSTALLED
  199   002322  012176                                  MSTC1
  200   002324  001366                                  TC1
  201   002326  000240                                  NOP
```

```
002330  004537  001456          11$:    JSR     R5,ASKTA        ;ASK IF TEST CONNECTOR #2 IS INSTALLED
002334  012255                          MSTC2
002336  001370                          TC2
002340  000240                          NOP
002342  000240                  12$:    NOP
002344  000240                  20$:    NOP
002346  104401  012345          30$:    TYPE,   MSG70           ;TELL THE OPERATOR THE TESTS AVAILABLE
002352  104401  011377          40$:    TYPE    ,MSG71
                                ;ROUTINE TO ASK OPERATOR WHAT SUB-SECTION TO EXECUTE
002356  104412                  TRYAG:  RDLIN
002360  052777  000100  176556          BIS     #100,@$TKS
002366  005046                          CLR     -(SP)           ;CLEAR PSW
002370  012746  002376                  MOV     #1$,-(SP)
002374  000002                          RTI
002376  012600                  1$:     MOV     (SP)+,R0        ;READ ANSWER
002400  011000                          MOV     (R0),R0         ;GET THE 1ST CHARACTER
002402  042700  177600                  BIC     #177600,R0      ;REMOVE EXTRA BITS
002406  012701  002434                  MOV     #OKCHAR,R1      ;LOAD POINTER TO GOOD CHARACTER LIST
002412  020021                  2$:     CMP     R0,(R1)+        ;CHECK IF VALID CHARACTER
002414  001002                          BNE     3$              ;BR IF NOT
002416  011101                          MOV     (R1),R1         ;GET THE ADDRESS
002420  000111                          JMP     @R1             ;DO THE SELECTED SUB-TEST
002422  005721                  3$:     TST     (R1)+           ;BUMP THE POINTER
002424  001372                          BNE     2$              ;BR IF MORE CHARACTERS
002426  104401  011077          6$:     TYPE    ,QUEST
002432  000751                          BR      TRYAG           ;WAIT FOR CHARACTER

                                ;TABLE OF VALID MENU CHARACTERS AND STARTING ADDRESS
002434  000141                  OKCHAR: 141                     ;LOWER CASE ''A''
002436  007320                          BEGINA
002440  000154                          154                     ;LOWER CASE 'L''
002442  007302                          BEGINL
002444  000167                          167                     ;LOWER CASE 'W''
002446  007342                          BEGINW
002450  000101                          'A
002452  007320                          BEGINA
002454  000114                          'L
002456  007302                          BEGINL
002460  000127                          'W
002462  007342                          BEGINW
002464  000061  006306                  '1      ,IOTST1
002470  000062  006462                  '2      ,IOTST2
002474  000063  006664                  '3,     IOTST3
002500  000064  006772                  '4,     IOTST4
002504  000065  007062                  '5,     IOTST5
002510  000066  007150                  '6,     IOTST6
002514  000067  007216                  '7,     IOTST7
002520  000000  000000  000000          0,0,0,0
002526  000000
```

```
 256  002530                              BEGL:
 257                                      ;;*****************************************************************
 (3)                                      ;*TEST 1         ADDRESS THE 4 BUS ADDRESSES OF THE AXV11-C
 (3)                                      ;;*****************************************************************
 (2)  002530  012737  002530  001106      TST1:   MOV     #TST1,$LPADR
 258  002536  012737  000001  001102              MOV     #$TN-1,$TSTNM    ;LOAD TEST NUMBER
 259  002544  005777  176546                      TST     @STREG           ;ADDRESS A/D STATUS REGISTER
 260  002550  005777  176546                      TST     @ADBUFF          ;ADDRESS A/D DATA BUFFER
 261  002554  005777  176544                      TST     @DACA            ;ADDRESS D TO A ''A''
 262  002560  005777  176542                      TST     @DACB            ;ADDRESS D TO A 'B''
 263                                      ;;*****************************************************************
 (3)                                      ;*TEST 2         FLOAT A ONE THRU MULTIPLEXER (BITS 11-8)
 (3)                                      ;;*****************************************************************
 (2)  002564  000004                      TST2:   SCOPE
 264  002566  012737  000400  001124              MOV     #BIT8,$GDDAT     ;LOAD FIRST BIT
 265  002574  104415                      2$:     CHKIT
 266  002576  104001                              ERROR   1                ;FAILED TO LOAD + READ BIT
 267  002600  006337  001124              1$:     ASL     $GDDAT           ;GET NEXT BIT
 268  002604  023727  001124  010000              CMP     $GDDAT,#BIT12    ;FINISHED?
 269  002612  001370                              BNE     2$               ;;NO,GO TO NEXT TEST
 270
 271                                      ;;*****************************************************************
 (3)                                      ;*TEST 3         LOAD AND READ BACK ERROR I.E. BIT14
 (3)                                      ;;*****************************************************************
 (2)  002614  000004                      TST3:   SCOPE
 272  002616  012737  040000  001124              MOV     #BIT14,$GDDAT
 273  002624  104415                              CHKIT
 274  002626  104001                              ERROR   1                ;FAILED TO LOAD + READ ERROR I.E.
 275                                      ;;*****************************************************************
 (3)                                      ;*TEST 4         LOAD AND READ BACK INTERRUPT ENABLE BIT6
 (3)                                      ;;*****************************************************************
 (2)  002630  000004                      TST4:   SCOPE
 276  002632  012777  001404  176470              MOV     #UNEXP,@VECTOR   ;SETUP FOR UNEXPECTED INTERUPT
 277  002640  012737  000100  001124              MOV     #BIT6,$GDDAT     ;LOAD EXPECTED DATA
 278  002646  104415                              CHKIT
 279  002650  104001                              ERROR   1                ;FAILED TO LOAD + READ INTERRUPT ENABLE
 280
 281                                      ;;*****************************************************************
 (3)                                      ;*TEST 5         LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BIT5
 (3)                                      ;;*****************************************************************
 (2)  002652  000004                      TST5:   SCOPE
 282  002654  012737  000040  001124              MOV     #BIT5,$GDDAT     ;LOAD EXPECTED DATA
 283  002662  104415                              CHKIT
 284  002664  104001                              ERROR   1                ;FAILED TO LOAD + READ CLOCK OVERFLOW START ENABLE
 285                                      ;;*****************************************************************
 (3)                                      ;*TEST 6         LOAD AND READ BACK EXTERNAL START ENABLE BIT4
 (3)                                      ;;*****************************************************************
 (2)  002666  000004                      TST6:   SCOPE
 286  002670  012737  000020  001124              MOV     #BIT4,$GDDAT     ;LOAD EXPECTED DATA
 287  002676  104415                              CHKIT
 288  002700  104001                              ERROR   1                ;FAILED TO LOAD + READ EXT. START ENABLE
```

```
                                        ;;*******************************************************************
                                        ;*TEST 7         LOAD AND READ BACK GAIN SELECT 0
                                        ;;*******************************************************************
002702  000004                          TST7:   SCOPE
002704  012737  000004  001124                  MOV     #BIT2,$GDDAT    ;LOAD EXPECTED DATA
002712  104415                                  CHKIT
002714  104001                                  ERROR   1               ;FAILED TO LOAD + READ BACK GAIN SELECT 0
                                        ;;*******************************************************************
                                        ;*TEST 10        LOAD AND READ BACK GAIN SELECT 1
                                        ;;*******************************************************************
002716  000004                          TST10:  SCOPE
002720  012737  000010  001124                  MOV     #BIT3,$GDDAT    ;LOAD EXPECTED
002726  104415                                  CHKIT
002730  104001                                  ERROR   1               ;FAILED TO LOAD + READ BACK GAIN SELECT 1

                                        ;;*******************************************************************
                                        ;*TEST 11        LOAD AND READ BACK ERROR FLAG (BIT15)
                                        ;;*******************************************************************
002732  000004                          TST11:  SCOPE
002734  012737  100000  001124                  MOV     #BIT15,$GDDAT   ;LOAD EXPECTED DATA
002742  104415                                  CHKIT
002744  104001                                  ERROR   1               ;FAILED TO LOAD + READ BACK ERROR FLAG
                                        ;;*******************************************************************
                                        ;*TEST 12        TEST INIT CLEARS BITS 2-6,14
                                        ;;*******************************************************************
002746  000004                          TST12:  SCOPE
002750  012737  000300  001160                  MOV     #300,$TIMES     ;;DO 300 ITERATIONS
002756  005037  001124                          CLR     $GDDAT          ;LOAD EXPECTED DATA
002762  012777  040174  176326                  MOV     #40174,@STREG   ;SET STATUS REGISTER
002770  000005                                  RESET                   ;INITIALIZE
002772  052777  000100  176144                  BIS     #100,@$TKS      ;SET INTRPT. ENABLE
003000  017737  176312  001126                  MOV     @STREG,$BDDAT   ;READ STATUS REGISTER
003006  001401                                  BEQ     TST13           ;;NEXT TEST
003010  104001                                  ERROR   1               ;RESET FAILED TO CLEAR AD ST. REG. BITS

                                        ;;*******************************************************************
                                        ;*TEST 13        TEST INIT CLEARS ERROR FLAG
                                        ;;*******************************************************************
003012  000004                          TST13:  SCOPE
003014  012737  000300  001160                  MOV     #300,$TIMES     ;;DO 300 ITERATIONS
003022  012777  100000  176266                  MOV     #BIT15,@STREG   ;SET BIT 15
003030  000005                                  RESET                   ;ISSUE INIT
003032  052777  000100  176104                  BIS     #100,@$TKS      ;SET INTRPT. EN. FOR KEYBOARD
003040  104414                                  CHECK
003042  104001                                  ERROR   1               ;BUS INIT FAILED TO CLEAR A/D DONE FLAG
                                        ;;*******************************************************************
                                        ;*TEST 14        TEST  DONE FLAG  SETS AND BIT0 CLEARS ON END OF CONV.
                                        ;;*******************************************************************
003044  000004                          TST14:  SCOPE
003046  017700  176250                          MOV     @ADBUFF,R0      ;READ DATA
003052  005277  176240                          INC     @STREG          ;START CONVERSION
003056  012737  000200  001124                  MOV     #BIT7,$GDDAT    ;LOAD EXPECTED
003064  004737  001426                          JSR     PC,STALL        ;DELAY AN AMOUNT OF TIME
003070  042777  100000  176220                  BIC     #BIT15,@STREG   ;MASK OUT ERROR BIT
003076  104414                                  CHECK
003100  104001                                  ERROR   1               ;A/D DONE FLAG FAILED TO SET
```

```
    326                                                                      ;  OR BIT0 FAILED TO CLEAR
    327  003102  017700  176214                   MOV     @ADBUFF,R0        ;CLEAR DONE FLAG FOR ITERATIONS
    328
    329                                     ;;*****************************************************************
    (3)                                     ;*TEST 15          TEST INIT CLEARS DONE FLAG
    (3)                                     ;;*****************************************************************
    (2)  003106  000004                     TST15:  SCOPE
    (1)  003110  012737  000300  001160           MOV     #300,$TIMES       ;;DO 300 ITERATIONS
    330  003116  005037  001124                   CLR     $GDDAT            ;CLEAR EXPECTED
    331  003122  005277  176170                   INC     @STREG            ;START CONVERSION
    332  003126  105777  176164         2$:       TSTB    @STREG
    333  003132  100375                           BPL     2$
    334  003134  000005                           RESET
    335  003136  104414                           CHECK
    336  003140  104001                           ERROR   1                 ;DONE FLAG FAILED TO CLEAR
    337  003142  052777  000100  175774           BIS     #100,@$TKS        ;SET INTRPT. EN. BIT
    338
    339                                     ;;*****************************************************************
    (3)                                     ;*TEST 16          TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
    (3)                                     ;;*****************************************************************
    (2)  003150  000004                     TST16:  SCOPE
    340  003152  005277  176140                   INC     @STREG            ;SET A/D START CONVERSION BIT
    341  003156  105777  176134         1$:       TSTB    @STREG            ;WAIT FOR FLAG
    342  003162  100375                           BPL     1$
    343  003164  017700  176132                   MOV     @ADBUFF,R0        ;READ CONVERTED VALUE
    344  003170  104414                           CHECK
    345  003172  104001                           ERROR   1                 ;DONE FLAG FAILED TO CLEAR
```

```
  347                                        ;;**************************************************************
  (3)                                        ;*TEST 17        GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
  (3)                                        ;;**************************************************************
  (2)  003174  000004                        TST17:  SCOPE
  348                                                ;* ''ENTERING TEST 17'' TYPED OUT TO TELL YOU THE NEXT
  (1)                                                ;*TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
  (1)                                                ;*THERE IS DANGER THAT THE ''Q BUSS'' COULD GET ''HUNG'' WHILE
  (1)                                                ;*EXECUTING TEST ''17''.
  (1)  003176  012700  000017                        MOV     #17,R0             ;GET TEST NO.
  (1)  003202  004737  010112                        JSR     PC,DUMW            ;PRINT MESSAGE
  349  003206  005046                                CLR     -(SP)              ;RESET PRIORITY
  350  003210  012746  003216                        MOV     #3$,-(SP)
  351  003214  000002                                RTI
  352  003216  012777  003272  176104  3$:           MOV     #1$,@VECTOR        ;INTERRUPT VECTOR ADDRESS
  353  003224  012777  000200  176100                MOV     #200,@VECTR1       ;SET UP NEW PSW
  354  003232  012777  000101  176056                MOV     #BIT6!BIT0,@STREG ;SET INTERRUPT ENABLE BIT + START CONVERSION
  355  003240  105777  176052          2$:           TSTB    @STREG             ;WAIT FOR DONE
  356  003244  100375                                BPL     2$                 ;FLAG TO SET
  357  003246  017737  176044  001126                MOV     @STREG,$BDDAT      ;READ STATUS REGISTER
  358  003254  012737  000300  001124                MOV     #BIT7!BIT6,$GDDAT ;GOOD DATA
  359  003262  104002                                ERROR   2                  ;FAILED TO INTERRUPT ON DONE
  360  003264  004737  010164                        JSR     PC,DUMC            ;TYPE COMPLETED
  361  003270  000414                                BR      TST20              ;;BRANCH TO NEXT TEST
  362  003272  022626                  1$:           CMP     (SP)+,(SP)+        ;RESET STACK POINTER
  363  003274  012777  001404  176026                MOV     #UNEXP,@VECTOR     ;SET UP FOR UNEXPECTED INTERRUPT
  364  003302  005046                                CLR     -(SP)              ;CLEAR PSW
  365  003304  012746  003312                        MOV     #4$,-(SP)
  366  003310  000002                                RTI
  367  003312  004737  010164          4$:           JSR     PC,DUMC            ;TYPE COMPLETED
  368  003316  005777  176000                        TST     @ADBUFF            ;CLEAR DONE BIT
  369                                        ;;**************************************************************
  (3)                                        ;*TEST 20        TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET
  (3)                                        ;;**************************************************************
  (2)  003322  000004                        TST20:  SCOPE
  370                                                ;* ''ENTERING TEST 20'' TYPED OUT TO TELL YOU THE NEXT
  (1)                                                ;*TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
  (1)                                                ;*THERE IS DANGER THAT THE ''Q BUSS'' COULD GET ''HUNG'' WHILE
  (1)                                                ;*EXECUTING TEST ''20''.
  (1)  003324  012700  000020                        MOV     #20,R0             ;GET TEST NO.
  (1)  003330  004737  010112                        JSR     PC,DUMW            ;PRINT MESSAGE
  371  003334  012777  003374  175772                MOV     #1$,@VECTR2        ;SETUP VECTOR ADDRESS
  372  003342  012777  140000  175746                MOV     #BIT15!BIT14,@STREG        ;CAUSE AN INTERRUPT
  373  003350  017737  175742  001126                MOV     @STREG,$BDDAT      ;BAD DATA
  374  003356  012737  140000  001124                MOV     #BIT15!BIT14,$GDDAT        ;GOOD DATA
  375  003364  104002                                ERROR   2
  376  003366  004737  010164                        JSR     PC,DUMC            ;TYPE COMPLETED
  377  003372  000753                                BR      TST20
  378  003374  022626                  1$:           CMP     (SP)+,(SP)+        ;POP STACK
  379  003376  004737  010164                        JSR     PC,DUMC
  380  003402  005077  175710                        CLR     @STREG
```

```
                                            ;;*******************************************************************
                                            ;;*TEST 21         TEST ERROR FLAG SETS IF 2ND CONVERSION IS STARTED WHILE A/D DONE IS SET
                                            ;;*******************************************************************
003406  000004                              TST21:   SCOPE
003410  012777  000001  175700                       MOV     #BIT0,@STREG      ;START CONVERSION
003416  105777  175674                      1$:      TSTB    @STREG            ;WAIT FOR
003422  100375                                       BPL     1$
003424  012737  100200  001124                       MOV     #BIT15!BIT7,$GDDAT ;LOAD EXPECTED VALUE
003432  012777  000001  175656                       MOV     #BIT0,@STREG      ;START 2ND CONVERSION
003440  104414                                       CHECK
003442  104001                                       ERROR   1                 ;ERROR FLAG NOT SET WHEN 2ND
                                                             ; CONVERSION WAS STARTED BEFORE READING BUFFER FROM FIRST
003444  017700  175652                               MOV     @ADBUFF,R0        ;CLEAR DONE FLAG


                                            ;;*******************************************************************
                                            ;;*TEST 22         TEST CLOCK OVERFLOW STARTS A/D  (IF KWV11-C IS AVAILABLE)
                                            ;;*******************************************************************
003450  000004                              TST22:   SCOPE
003452  005737  001374                               TST     KWAD              ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
003456  001424                                       BEQ     TST23             ;;BR IF NO CLOCK THERE
003460  012737  000240  001124                       MOV     #BIT7!BIT5,$GDDAT         ;LOAD EXPECTED A/D STATUS
003466  013777  001124  175622                       MOV     $GDDAT,@STREG     ;ENABLE THE A/D STATUS REGISTER
003474  012777  177776  175640                       MOV     #177776,@KWBPR    ;LOAD KWV11-C CLOCK PRESET REGISTER
003502  012777  000011  175630                       MOV     #11,@KWCSR        ;START CLOCK
003510  004737  001426                               JSR     PC,STALL          ;DELAY FOR A CLOCK TICK
003514  104414                                       CHECK                     ;CHECK A/D STATUS AGAINST EXPECTED
003516  104001                                       ERROR   1                 ;A/D DONE FAILED TO SET WITH CLOCK STARTS
003520  005777  175576                               TST     @ADBUFF           ;CLEAR A/D DONE
003524  005077  175566                               CLR     @STREG            ;CLEAR A/D CONTROL


                                            ;;*******************************************************************
                                            ;;*TEST 23         TEST EXTERNAL TRIGGER STARTS A/D (IF KW11-C IS CONNECTED TO EXT START TA
                                            ;;*******************************************************************
003530  000004                              TST23:   SCOPE
003532  005737  001376                               TST     KWEX              ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
003536  001424                                       BEQ     TST24             ;;BR IF NO CLOCK THERE
003540  012737  000220  001124                       MOV     #BIT7!BIT4,$GDDAT         ;LOAD EXPECTED A/D STATUS
003546  013777  001124  175542                       MOV     $GDDAT,@STREG     ;ENABLE THE A/D STATUS REGISTER
003554  012777  177776  175560                       MOV     #177776,@KWBPR    ;LOAD KWV11-C CLOCK PRESET REGISTER
003562  012777  000011  175550                       MOV     #11,@KWCSR        ;START CLOCK
003570  004737  001426                               JSR     PC,STALL          ;DELAY FOR CLOCK TICKS
003574  104414                                       CHECK                     ;CHECK A/D STATUS AGAINST EXPECTED
003576  104001                                       ERROR   1                 ;A/D DONE FAILED TO SET WITH EXTERNAL STARTS
003600  005777  175516                               TST     @ADBUFF           ;CLEAR A/D DONE
003604  005077  175506                               CLR     @STREG            ;CLEAR A/D CONTROL

```

```
                                        ;;*************************************************************
                                        ;*TEST 24        TEST EXTERNAL TRIGGER STARTS A/D (IF MANUAL TRIGGER IS CONNECTED TO EXT
                                        ;;*************************************************************
003610  000004                          TST24:  SCOPE
003612  005737  001400                          TST     MAEX            ;TEST IF OPERATOR SAID MANUAL TRIGGER IS CONNECTED
003616  001427                                  BEQ     TST25           ;;BR IF NO EXT. TRIGGER AVAILABLE
003620  005737  001202                          TST     $PASS           ;TEST IF FIRST PASS OF PROGRAM
003624  001024                                  BNE     TST25           ;;BR IF NOT FIRST PASS
003626  012737  000220  001124                  MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPECTED A/D STATUS
003634  013777  001124  175454                  MOV     $GDDAT,@STREG   ;ENABLE THE EXT START SIGNAL
003642  104401  012016                          TYPE    ,MSGNEX         ;TELL OPERATOR TO GENERATE EXT. TRIGGER
003646  104401  011276                          TYPE    ,CRWR           ;TELL OPERATOR ABOUT 'RETURN''
003652  104412                                  RDLIN
003654  012600                                  MOV     (SP)+,R0        ;REMOVE ANSWER OFF OF THE STACK
003656  000240                                  NOP
003660  000240                                  NOP
003662  104414                                  CHECK                   ;CHECK A/D STATUS AGAINST EXPECTED
003664  104001                                  ERROR   1               ;A/D DONE FAILED TO SET WITH EXTERNAL START
003666  005777  175430                          TST     @ADBUFF         ;CLEAR A/D DONE
003672  005077  175420                          CLR     @STREG          ;CLEAR A/D CONTROL

                                        ;;*************************************************************
                                        ;*TEST 25        TEST ERROR FLAG SETS IS START 2ND CONV. BEFORE DONE FLAG SETS (KWV11-C)
                                        ;;*************************************************************
003676  000004                          TST25:  SCOPE
003700  005737  001374                          TST     KWAD            ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
003704  001436                                  BEQ     TST26           ;;BR IF NO CLOCK PRESENT
003706  012737  100240  001124                  MOV     #BIT15!BIT7!BIT5,$GDDAT ;LOAD EXPECTED
003714  012777  177776  175420                  MOV     #-2,@KWBPR      ;LOAD CLOCK PRESET
003722  012777  000040  175366                  MOV     #BIT5,@STREG    ;ENABLE CLOCK START
003730  017700  175366                          MOV     @ADBUFF,R0      ;ENSURE CLEARED A/D DONE
003734  012777  000011  175376                  MOV     #11,@KWCSR      ;START CLOCK
003742  105777  175372                  1$:     TSTB    @KWCSR          ;WAIT FOR CLOCK READY
003746  100375                                  BPL     1$
003750  152777  000001  175340                  BISB    #BIT0,@STREG    ;CLOCK OVERFLOW SHOULD HAVE STARTED A/D
                                                                        ;TRY TO START IT AGAIN AND GET AN ERROR
003756  017737  175334  001126                  MOV     @STREG,$BDDAT   ;READ A/D STATUS
003764  023737  001124  001126                  CMP     $GDDAT,$BDDAT   ;COMPARE TO EXPECTED
003772  001401                                  BEQ     2$              ;;BR IF SAME
003774  104001                                  ERROR   1               ;ERROR FLAG NOT SET WHEN 2ND CONVERT STARTED
                                                                        ; WHILE FIRST IS IN PROGRESS
003776  017700  175320                  2$:     MOV     @ADBUFF,R0      ;READ AND CLEAR A/D DONE
```

```
 457                                        ;;*****************************************************************
 (3)                                        ;*TEST 26        TEST 'B EVENT'' STARTS A/D (IF JUMPER 'F2'' IS PRESENT)
 (3)                                        ;;*****************************************************************
 (2)  004002  000004                        TST26:  SCOPE
 458  004004  005737  001402                        TST     BTEX            ;TEST IF OPERATOR SAID 'F2'' IS INSTALLED
 459  004010  001416                                BEQ     TST27           ;;BR IF NOT THERE
 460  004012  012737  000220  001124                MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPECTED A/D STATUS
 461  004020  013777  001124  175270                MOV     $GDDAT,@STREG   ;ENABLE THE A/D STATUS REGISTER
 462  004026  004737  001426                        JSR     PC,STALL        ;DELAY AN AMOUNT OF TIME
 463  004032  104414                                CHECK                   ;CHECK A/D STATUS AGAINST EXPECTED
 464  004034  104001                                ERROR   1               ;A/D DONE FAILED TO SET WITH 'B EVENT''
 465  004036  005077  175254                        CLR     @STREG          ;CLEAR A/D CONTROL
 466  004042  005777  175254                        TST     @ADBUFF         ;CLEAR A/D DONE
 467
 468
 469                                        ;;*****************************************************************
 (3)                                        ;*TEST 27        END OF ADV11-C LOGIC TESTS
 (3)                                        ;;*****************************************************************
 (2)  004046  000004                        TST27:  SCOPE
 470  004050  000207                                RTS     PC              ;RETURN TO TEST SECTION
 471
 472
 473                                                .SBTTL
 474                                                .SBTTL  END OF LOGIC TESTS - SECTION
 475
 476
 477                                        ;;SUBROUTINE FOR LOGIC TESTS;;
 478  004052  013777  001124  175236        TESTIT: MOV     $GDDAT,@STREG   ;LOAD EXPECTED VALUE
 479  004060  017737  175232  001126        TEST:   MOV     @STREG,$BDDAT   ;READ ST. REG.
 480  004066  023737  001124  001126                CMP     $GDDAT,$BDDAT   ;COMPARE RESULTS
 481  004074  001002                                BNE     RETERR          ;;ERROR RETURN
 482  004076  062716  000002                        ADD     #2,(SP)         ;BUMP RETURN ADDRESS TO GET AROUND ERROR
 483  004102  000002                        RETERR: RTI
 484
 485                                                .SBTTL
 486                                                .SBTTL  START OF ADV11-C ANALOG WRAPAROUND SECTION
 487                                                .SBTTL
```

```
  489  004104                              WRAP:
  (4)                                      ;;******************************************************************
  (3)                                      ;*TEST 30        SETUP TO RUN ANALOG WRAPAROUND TEST
  (3)                                      ;;******************************************************************
  (2)  004104  012737  000030  001102      TST30:  MOV     #$TN,$TSTNM
  (1)  004112  012737  000001  001160              MOV     #1,$TIMES          ;;DO 1 ITERATION
  490                                      ;LOAD AXV11-C DAC TO MAX OUTPUT VOLTAGE
  491  004120  012777  007777  175176              MOV     #7777,@DACA        ;LOAD DAC ''A''
  492  004126  012777  007777  175172              MOV     #7777,@DACB        ;LOAD DAC 'B''
  493  004134  012737  004156  001110              MOV     #1$,$LPERR         ;LOAD ERROR ADDRESS
  494  004142  012737  004156  001106              MOV     #1$,$LPADR         ;LOAD LOOP ADDRESS
  495                                      ;DELAY SUFFICIENT TIME TO LET THE DAC'S SETTLE
  496  004150  012700  000002                      MOV     #2,R0              ;LOAD DELAY TIMER
  497  004154  005001                              CLR     R1                 ;CLEAR DELAY COUNT
  498  004156  005301                      1$:     DEC     R1                 ;DELAY
  499  004160  001376                              BNE     1$
  500  004162  005300                              DEC     R0                 ;DELAY
  501  004164  001374                              BNE     1$
  502
  503                                      ;;******************************************************************
  (3)                                      ;*TEST 31        COMPARE CHANNEL 0 (F.S.) AGAINST 1 (1/2 FS), 2 (1/4 FS), 3 (1/8)
  (3)                                      ;;******************************************************************
  (2)  004166  000004                      TST31:  SCOPE
  (1)  004170  012737  000001  001160              MOV     #1,$TIMES          ;;DO 1 ITERATION
  504  004176  005737  001366              1$:     TST     TC1                ;TEST IF TEST FIXTURE IS INSTALLED
  505  004202  001440                              BEQ     TST32              ;;BR IF NOT
  506  004204  004537  007710                      JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
  507  004210  000000                              CHAN00                     ;CHANNEL 0
  508  004212  004537  010046                      JSR     R5,COMPAR          ;COMPARE RESULTS
  509  004216  007777                              7777
  510  004220  001354                              VWRAP
  511  004222  104004                              ERROR   4                  ;ERROR AN A/D CHANNEL 0 - VALUE DID NOT
  512                                                                         ;    EQUAL EXPECTED VALUE
  513  004224  004537  007710                      JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
  514  004230  000001                              CHAN01                     ;CHANNEL 1
  515  004232  004537  010046                      JSR     R5,COMPAR          ;COMPARE RESULTS
  516  004236  006000                              6000                       ;EXPECTED VALUE
  517  004240  001354                              VWRAP                      ;USING A KNOWN SPREAD
  518  004242  104004                              ERROR   4                  ;ERROR ON A/D CHANNEL 1 - VALUE DID NOT
  519                                                                         ;   EQUAL EXPECTED
  520  004244  004537  007710                      JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
  521  004250  000002                              CHAN02                     ;CHANNEL 2
  522  004252  004537  010046                      JSR     R5,COMPAR          ;COMPARE RESULTS
  523  004256  005000                              5000                       ;AGAINST THIS VALUE FOR CHANNEL 2
  524  004260  001354                              VWRAP                      ;USING A KNOWN SPREAD
  525  004262  104004                              ERROR   4                  ;ERROR ON A/D CHANNEL 2 - VALUE DID NOT
  526                                                                         ;   EQUAL EXPECTED
  527  004264  004537  007710                      JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
  528  004270  000003                              CHAN03                     ;CHANNEL 03
  529  004272  004537  010046                      JSR     R5,COMPAR          ;COMPARE RESULTS
  530  004276  004400                              4400                       ;AGAINST THIS VALUE FOR CHANNEL 3
  531  004300  001354                              VWRAP                      ;USING A KNOWN SPREAD
  532  004302  104004                              ERROR   4                  ;ERROR ON A/D CHANNEL 3 - VALUE DID NOT
  533                                                                         ;  EQUAL EXPECTED
```

```
    535                                       ;;*****************************************************************
    (3)                                       ;*TEST 32       COMPARE CHANNEL 0 (F.S.) AGAINST OTHER F.S. CHANNELS (4 AND 10)
    (3)                                       ;;*****************************************************************
    (2)   004304  000004                      TST32:  SCOPE
    (1)   004306  012737  000001  001160              MOV     #1,$TIMES          ;;DO 1 ITERATION
    536   004314  005737  001366                      TST     TC1                ;TEST IF TEST FIXTURE IS INSTALLED
    537   004320  001431                              BEQ     TST33              ;;BR IF NOT
    538   004322  004537  007710                      JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
    539   004326  000000                              CHAN00                     ;CHANNEL 0
    540   004330  013737  001360  004356              MOV     TEMP,4$            ;SAVE CHANNEL 00 CONVERTED VALUE
    541   004336  013737  001360  004376              MOV     TEMP,10$           ;
    542
    543   004344  004537  007710                      JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
    544   004350  000004                              CHAN04                     ;CHANNEL 4
    545   004352  004537  010046                      JSR     R5,COMPAR          ;COMPARE RESULTS
    546   004356  000000                      4$:     0                          ;AGAINST THIS VALUE FOR CHANNEL 0
    547   004360  010236                              V2                         ;USING A SPREAD OF 2 COUNTS
    548   004362  104004                              ERROR   4                  ;ERROR ON A/D CHANNEL 4 - VALUE DID NOT
    549                                                                          ;  EQUAL VALUE OF CHANNEL 0
    550
    551   004364  004537  007710                      JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
    552   004370  000010                              CHAN10                     ;CHANNEL 10
    553   004372  004537  010046                      JSR     R5,COMPAR          ;COMPARE RESULTS
    554   004376  000000                      10$:    0                          ;AGAINST THIS VALUE FOR CHANNEL 0
    555   004400  010236                              V2                         ;USING A SPREAD OF 2 COUNTS
    556   004402  104004                              ERROR   4                  ;ERROR ON A/D CHANNEL 10 - VALUE DID NOT
    557                                                                          ;  EQUAL VALUE OF CHANNEL 0
```

```
 559
 560                                        ;;*******************************************************************
 (3)                                        ;;*TEST 33        COMPARE CHANNEL 1 (1/2 F.S.) AGAINST OTHER 1/2 F.S. CHANNELS (5 AND 11)
 (3)                                        ;;*******************************************************************
 (2)  004404  000004                        TST33:  SCOPE
 (1)  004406  012737  000001  001160                MOV     #1,$TIMES       ;;DO 1 ITERATION
 561  004414  005737  001366                        TST     TC1             ;TEST IF TEST FIXTURE IS INSTALLED
 562  004420  001431                                BEQ     TST34           ;;BR IF NOT
 563  004422  004537  007710                        JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 564  004426  000001                                CHAN01                  ;CHANNEL 1
 565  004430  013737  001360  004456                MOV     TEMP,4$         ;SAVE CHANNEL 1 CONVERTED VALUE
 566  004436  013737  001360  004476                MOV     TEMP,10$        ;SAVE IT AGAIN
 567
 568  004444  004537  007710                        JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 569  004450  000005                                CHAN05                  ;CHANNEL 5
 570  004452  004537  010046                        JSR     R5,COMPAR       ;COMPARE RESULTS
 571  004456  000000                        4$:     0                       ;AGAINST THIS VALUE FOR CHANNEL 1
 572  004460  010236                                V2                      ;USING A SPREAD OF 2 COUNTS
 573  004462  104004                                ERROR   4               ;ERROR ON A/D CHANNEL 5 - VALUE DID NOT
 574                                                                        ;  EQUAL VALUE OF CHANNEL 0
 575
 576  004464  004537  007710                        JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 577  004470  000011                                CHAN11                  ;CHANNEL 11
 578  004472  004537  010046                        JSR     R5,COMPAR       ;COMPARE RESULTS
 579  004476  000000                        10$:    0                       ;AGAINST THIS VALUE FOR CHANNEL 1
 580  004500  010236                                V2                      ;USING A SPREAD OF 2 COUNTS
 581  004502  104004                                ERROR   4               ;ERROR ON A/D CHANNEL 11 - VALUE DID NOT
 582                                                                        ;  EQUAL VALUE OF CHANNEL 1
 583
```

```
                                    ;;*******************************************************************
                                    ;*TEST 34        COMPARE CHANNEL 2 (1/4 F.S.) AGAINST OTHER 1/4 F.S. CHANNELS (6 AND 12)
                                    ;;*******************************************************************
004504  000004                      TST34:  SCOPE
004506  012737  000001  001160              MOV     #1,$TIMES           ;;DO 1 ITERATION
004514  005737  001366                      TST     TC1                 ;TEST IF TEST FIXTURE IS INSTALLED
004520  001431                              BEQ     TST35               ;;BR IF NOT
004522  004537  007710                      JSR     R5,CONVRT           ;GET THE AVERAGE VALUE FOR
004526  000002                              CHAN02                      ;CHANNEL 2
004530  013737  001360  004556              MOV     TEMP,4$             ;SAVE CHANNEL 2 CONVERTED VALUE
004536  013737  001360  004576              MOV     TEMP,10$            ;SAVE IT AGAIN

004544  004537  007710                      JSR     R5,CONVRT           ;GET THE AVERAGE VALUE FOR
004550  000006                              CHAN06                      ;CHANNEL 6
004552  004537  010046                      JSR     R5,COMPAR           ;COMPARE RESULTS
004556  000000                      4$:     0                           ;AGAINST THIS VALUE FOR CHANNEL 2D
004560  010236                              V2                          ;USING A SPREAD OF 2 COUNTS
004562  104004                              ERROR   4                   ;ERROR ON A/D CHANNEL 6 - VALUE DID NOT
                                                                        ;  EQUAL VALUE OF CHANNEL 2

004564  004537  007710                      JSR     R5,CONVRT           ;GET THE AVERAGE VALUE FOR
004570  000012                              CHAN12                      ;CHANNEL 12
004572  004537  010046                      JSR     R5,COMPAR           ;COMPARE RESULTS
004576  000000                      10$:    0                           ;AGAINST THIS VALUE FOR CHANNEL 2
004600  010236                              V2                          ;USING A SPREAD OF 2 COUNTS
004602  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.)
                                    ;;*******************************************************************
004604  000004                      TST35:  SCOPE
004606  012737  000001  001160              MOV     #1,$TIMES           ;;DO 1 ITERATION
004614  005737  001366                      TST     TC1                 ;TEST IF TEST FIXTURE IS INSTALLED
004620  001416                              BEQ     TST36               ;;BR IF NOT
004622  004537  007710                      JSR     R5,CONVRT           ;GET THE AVERAGE VALUE FOR
004626  000003                              CHAN03                      ;CHANNEL 3
004630  013737  001360  004650              MOV     TEMP,4$             ;SAVE CHANNEL 3 CONVERTED VALUE

004636  004537  007710                      JSR     R5,CONVRT           ;GET THE AVERAGE VALUE FOR
004642  000007                              CHAN07                      ;CHANNEL 7
004644  004537  010046                      JSR     R5,COMPAR           ;COMPARE RESULTS
004650  000000                      4$:     0                           ;AGAINST THIS VALUE FOR CHANNEL 3
004652  010236                              V2                          ;USING A SPREAD OF 2 COUNTS
004654  104004                              ERROR   4                   ;ERROR ON A/D CHANNEL 7 - VALUE DID NOT
                                                                        ;  EQUAL VALUE OF CHANNEL 3

```

```
 625                                            ;;*****************************************************************
 (3)                                            ;*TEST 36         RELATIVE GAIN TEST USING CHANNEL 3 (1/8 F.S.)
 (3)                                            ;;*****************************************************************
 (2)  004656  000004                            TST36:  SCOPE
 (1)  004660  012737  000001  001160                    MOV     #1,$TIMES        ;;DO 1 ITERATION
 626  004666  005737  001366                            TST     TC1              ;TEST IF AXV11 OR ADV11 CONNECTOR INSTALLED
 627  004672  001454                                    BEQ     TST37            ;;BR IF NO CONNECTOR
 628  004674  012737  000000  010044                    MOV     #GAIN00,OTHER    ;SELECT GAIN OF 00
 629  004702  004537  007714                            JSR     R5,CONVTR        ;GET THE VALUE OF CHANNEL 03
 630  004706  000003                                    CHAN03
 631  004710  004537  010046                            JSR     R5,COMPAR        ;TEST GAIN
 632  004714  004400                                    4400                     ;EXPECTED VALUE
 633  004716  001354                                    VWRAP                    ; USING KNOWN SPREAD
 634  004720  104004                                    ERROR   4                ;GAIN SELECT OF 00 FAILED TO EQUAL EXPECTED VALUE
 635
 636  004722  012737  000004  010044                    MOV     #GAIN01,OTHER    ;SELECT GAIN OF 01
 637  004730  004537  007714                            JSR     R5,CONVTR        ;GET THE VALUE OF CHANNEL 03
 638  004734  000003                                    CHAN03
 639  004736  004537  010046                            JSR     R5,COMPAR        ;TEST GAIN 01
 640  004742  005000                                    5000                     ;EXPECTED VALUE
 641  004744  001354                                    VWRAP                    ; USING KNOWN SPREAD
 642  004746  104004                                    ERROR   4                ;GAIN SELECT OF 01 FAILED TO INCREASE
 643                                                                             ;  CONVERTED VALUE CORRECTLY
 644  004750  012737  000010  010044                    MOV     #GAIN10,OTHER    ;SET GAIN SELECT = 10
 645  004756  004537  007714                            JSR     R5,CONVTR        ;GET VALUE OF CHANNEL 03
 646  004762  000003                                    CHAN03
 647  004764  004537  010046                            JSR     R5,COMPAR        ;TEST GAIN 10 VALUE AGAINST 01
 648  004770  006000                                    6000                     ;EXPECTED VALUE
 649  004772  001354                                    VWRAP                    ; USING KNOWN SPREAD
 650  004774  104004                                    ERROR   4                ;GAIN SELECT OF 10 FAILED TO INCREASE
 651                                                                             ;  CONVERTED VALUE CORRECTLY
 652  004776  012737  000014  010044                    MOV     #GAIN11,OTHER    ;SET GAIN SELECT = 11
 653  005004  004537  007714                            JSR     R5,CONVTR        ;GET VALUE OF CHANNEL 03
 654  005010  000003                                    CHAN03
 655  005012  004537  010046                            JSR     R5,COMPAR        ;TEST GAIN 11 VALUE AGAINST 10
 656  005016  007777                                    7777                     ;EXPECTED VALUE
 657  005020  001354                                    VWRAP                    ;  USING KNOWN SPREAD
 658  005022  104004                                    ERROR   4                ;GAIN SELECT OF 11 FAILED TO INCREASE
 659                                                                             ;  CONVERTED VALUE CORRECTLY
 660
 661                                            ;;*****************************************************************
 (3)                                            ;*TEST 37         IF ADV11-C VERIFY CH13 IS AT + F.S.
 (3)                                            ;;*****************************************************************
 (2)  005024  000004                            TST37:  SCOPE
 (1)  005026  012737  000001  001160                    MOV     #1,$TIMES        ;;DO 1 ITERATION
 662  005034  012777  004000  174264                    MOV     #4000,@DACB      ;SET DAC "B" TO MIDRANGE
 663  005042  005737  001372                            TST     ADV11C           ;TEST IF ADV11-C
 664  005046  001410                                    BEQ     TST40            ;;BR IF NOT ADV11-C
 665  005050  004537  007710                            JSR     R5,CONVRT        ;GET THE CONVERTED VALUE FOR CH13
 666  005054  000013                                    CHAN13
 667  005056  004537  010046                            JSR     R5,COMPAR        ;TEST CH13 AGAINST EXPECTED
 668  005062  007777                                    7777                     ;+ F.S.
 669  005064  010236                                    V2
 670  005066  104004                                    ERROR   4                ;CH13 WAS NOT PULLED UP TO +F.S.
 672                                              .SBTTL
 673                                              .SBTTL  END OF ADV11-C ANALOG WRAPAROUND SECTION
 674                                              .SBTTL
 675                                              .SBTTL  START OF AXV11-C ANALOG WRAPAROUND SECTION
 676                                              .SBTTL
 677
 678                                      ;;*****************************************************************
 (3)                                      ;*TEST 40       AXV11-C ANALOG WRAPAROUND TEST (DAC ''A'' TO A/D CHAN 0)
 (3)                                      ;;*****************************************************************
 (2)   005070  000004                     TST40:  SCOPE
 (1)   005072  012737  000001  001160             MOV     #1,$TIMES             ;;DO 1 ITERATION
 679                                              ;AXV11-C DAC ''A'' CONNECTED TO AXV11-C A/D CHANNEL 0
 680                                              ;AXV11-C TEST FIXTURE IS REQUIRED
 681
 682   005100  005737  001366                     TST     TC1                   ;TEST IF AXV11-C TEST FIXTURE IS PRESENT
 683   005104  001445                             BEQ     TST41                 ;;BR IF NO TEST FIXTURE
 684   005106  005737  001372                     TST     ADV11C                ;TEST IF THE MODULE IS A ADV11-C
 685   005112  001042                             BNE     TST41                 ;;BR IF NO DAC'S PRESENT
 686   005114  012737  000000  005154             MOV     #0,2$                 ;PRIME THE DAC OUTPUT VALUE
 687   005122  013777  005154  174174             MOV     2$,@DACA              ;PRIME THE DAC OUTPUT STAGE
 688   005130  012777  000000  174160             MOV     #0,@STREG             ;INITIILIZE THE A/D STATUS REG
 689   005136  017700  174160                     MOV     @ADBUFF,R0            ;READ A/D VALUE AND CLEAR A/D DONE FLAG
 690   005142  004537  007710             1$:     JSR     R5,CONVRT             ;GET THE VALUE OF CHANNEL 0
 691   005146  000000                             CHAN00
 692   005150  004537  010046                     JSR     R5,COMPAR             ;COMPARE AGAINST EXPECTED D/A VALUE
 693   005154  000000                     2$:     0                             ;EXPECTED
 694   005156  001354                             VWRAP                         ;SPREAD ALLOWED
 695   005160  000413                             BR      3$                    ;CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
 696   005162  062737  000010  005154             ADD     #10,2$                ;UPDATE THE D/A OUTPUT VALUE
 697   005170  013777  005154  174126             MOV     2$,@DACA              ;UPDATE THE D/A OUTPUT VOLTAGE
 698   005176  022737  010000  005154             CMP     #10000,2$             ;TEST IF LAST STEP
 699   005204  001356                             BNE     1$
 700   005206  000401                             BR      4$                    ;;BR TO NEXT TEST
 701   005210  104004                     3$:     ERROR   4                     ;CONVERTED A/D VALUE DID NOT EQUAL EXPECTED VALUE
 702   005212  012777  007777  174104     4$:     MOV     #7777,@DACA           ;LOAD DAC ''A'' TO +F.S.
 703
```

```
                                         ;;*******************************************************************
(3)                                      ;*TEST 41         AXV11-C ANALOG WRAPAROUND TEST (DAC 'B'' TO A/D CHAN 13)
(3)                                      ;;*******************************************************************
(2)   005220  000004                     TST41:  SCOPE
(1)   005222  012737  000001  001160             MOV     #1,$TIMES              ;;DO 1 ITERA.ION
                                                 ;AXV11-C DAC ''B'' CONNECTED TO AXV11-C A/D CHANNEL 13
                                                 ;AXV11-C TEST CABLE IS REQUIRED

      005230  005737  001366                     TST     TC1                    ;TEST IF AXV11-C TEST FIXTURE IS PRESENT
      005234  001445                             BEQ     TST42                  ;;BR IF NO TEST FIXTURE
      005236  005737  001372                     TST     ADV11C                 ;TEST IF MODULE IS AN ADV11-C
      005242  001042                             BNE     TST42                  ;;BR IF NO DAC'A PRESENT
      005244  012737  000000  005304             MOV     #0,2$                  ;PRIME THE DAC OUTPUT VALUE
      005252  013777  005304  174046             MOV     2$,@DACB               ;PRIME THE DAC OUTPUT STAGE
      005260  012777  000000  174030             MOV     #0,@STREG              ;INITIILIZE THE A/D STATUS REG
      005266  017700  174030                     MOV     @ADBUFF,R0             ;READ A/D VALUE AND CLEAR A/D DONE FLAG
      005272  004537  007710             1$:     JSR     R5,CONVRT              ;GET THE VALUE OF CHANNEL 13
      005276  000013                             CHAN13
      005300  004537  010046                     JSR     R5,COMPAR              ;COMPARE AGAINST EXPECTED D/A VALUE
      005304  000000                     2$:     0                              ;EXPECTED
      005306  001354                             VWRAP                          ;SPREAD ALLOWED
      005310  000413                             BR      3$                     ;CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
      005312  062737  000010  005304             ADD     #10,2$                 ;UPDATE THE D/A OUTPUT VALUE
      005320  013777  005304  174000             MOV     2$,@DACB               ;UPDATE THE D/A OUTPUT VOLTAGE
      005326  022737  010000  005304             CMP     #10000,2$              ;TEST IF LAST STEP
      005334  001356                             BNE     1$
      005336  000401                             BR      4$                     ;;BR TO NEXT TEST
      005340  104004                     3$:     ERROR   4                      ;CONVERTED D/A VALUE DID NOT EQUAL EXPECTED
      005342  012777  007777  173756     4$:     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.
                                        ;;*****************************************************************
005350  000004                          TST42:  SCOPE
005352  012737  000001  001160                  MOV     #1,$TIMES               ;;DO 1 ITERATION
                                                ;AAV11-C TEST CONNECTOR IS NOT REQUIRED (IN FACT WILL ERROR IF PRESENT)

005360  005737  001370                          TST     TC2                     ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
005364  001045                                  BNE     TST43                   ;;BR IF TEST CONNECTOR
005366  012777  000000  173722                  MOV     #0,@STREG               ;INITIILIZE THE A/D STATUS REG
005374  017700  173722                          MOV     @ADBUFF,R0              ;READ A/D VALUE AND CLEAR A/D DONE FLAG
005400  004537  007710                          JSR     R5,CONVRT               ;GET THE VALUE OF CHANNEL 14
005404  000014                                  CHAN14
005406  004537  010046                          JSR     R5,COMPAR               ;COMPARE AGAINST EXPECTED VALUE
005412  004000                                  4000                            ;EXPECTED
005414  010236                                  V2                              ;SPREAD ALLOWED
005416  104004                                  ERROR   4                       ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

005420  004537  007710                          JSR     R5,CONVRT               ;GET THE VALUE OF CHANNEL 15
005424  000015                                  CHAN15
005426  004537  010046                          JSR     R5,COMPAR               ;COMPARE AGAINST EXPECTED VALUE
005432  004000                                  4000
005434  010236                                  V2                              ;SPREAD ALLOWED
005436  104004                                  ERROR   4                       ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

005440  004537  007710                          JSR     R5,CONVRT               ;GET THE VALUE OF CHANNEL 16
005444  000016                                  CHAN16
005446  004537  010046                          JSR     R5,COMPAR               ;COMPARE AGAINST EXPECTED VALUE
005452  004000                                  4000
005454  010236                                  V2                              ;SPREAD ALLOWED
005456  104004                                  ERROR   4                       ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE

005460  004537  007710                          JSR     R5,CONVRT               ;GET THE VALUE OF CHANNEL 17
005464  000017                                  CHAN17
005466  004537  010046                          JSR     R5,COMPAR               ;COMPARE AGAINST EXPECTED VALUE
005472  004000                                  4000
005474  010236                                  V2                              ;SPREAD ALLOWED
005476  104004                                  ERROR   4                       ;CONVERTED VLAUE DID NOT EQUAL EXPECTED VALUE
```

```

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

                                  ;;********************************************************************
(3)                               ;*TEST 43       AAV11-C ANALOG WRAPAROUND TEST (DAC ''A'' TO A/D CHAN 14)
(3)                               ;;********************************************************************
(2) 005500 000004                 TST43:  SCOPE
(1) 005502 012737 000001 001160           MOV     #1,$TIMES         ;;DO 1 ITERATION
                                          ;AAV11-C TEST CONNECTOR IS REQUIRED

    005510 005737 001370                  TST     TC2               ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
    005514 001452                         BEQ     TST44             ;;BR IF NO TEST CONNECTOR
    005516 012737 000000 005562           MOV     #0,2$             ;PRIME THE DAC OUTPUT VALUE
    005524 012777 007777 173612           MOV     #7777,@DACO       ;PRIME THE DAC OUTPUT STAGE
    005532 012777 000000 173556           MOV     #0,@STREG         ;INITIILIZE THE A/D STATUS REG
    005540 017700 173556                  MOV     @ADBUFF,R0        ;READ A/D VALUE AND CLEAR A/D DONE FLAG
    005544 000240                         NOP
    005546 000240                         NOP

    005550 004537 007710          1$:     JSR     R5,CONVRT         ;GET THE VALUE OF CHANNEL 14
    005554 000014                         CHAN14
    005556 004537 010046                  JSR     R5,COMPAR         ;COMPARE AGAINST EXPECTED D/A VALUE
    005562 000000                 2$:     0
    005564 001354                         VWRAP                     ;SPREAD ALLOWED
    005566 000424                         BR      10$               ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
    005570 062737 000010 005562           ADD     #10,2$            ;UPDATE THE D/A OUTPUT VALUE
    005576 013737 005562 005636           MOV     2$,7$             ;COPY VALUE
    005604 005137 005636                  COM     7$                ;INVERT DATA
    005610 042737 170000 005636           BIC     #170000,7$        ;REMOVE EXTRA BITS
    005616 013777 005636 173520           MOV     7$,@DACO          ;UPDATE THE D/A OUTPUT VOLTAGE
    005624 022737 010000 005562           CMP     #10000,2$         ;TEST IF LAST STEP
    005632 001346                         BNE     1$
    005634 000402                         BR      TST44             ;;BR TO NEXT TEST
    005636 000000                 7$:     0
    005640 104004                 10$:    ERROR   4                 ;CONVERTED D/A VALUE DID NOT EQUAL EXPECTED

```

```


                                          ;;*****************************************************************
(3)                                       ;*TEST 44          AAV11-C ANALOG WRAPAROUND TEST (DAC 'B'' TO A/D CHAN 15)
(3)                                       ;;*****************************************************************
(2)   005642  000004                      TST44:  SCOPE
(1)   005644  012737  000001  001160              MOV     #1,$TIMES          ;;DO 1 ITERATION
                                                  ;AAV11-C TEST CONNECTOR IS REQUIRED

      005652  005737  001370                      TST     TC2                ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
      005656  001450                              BEQ     TST45              ;;BR IF NO TEST CONNECTOR
      005660  012737  000000  005720              MOV     #0,2$              ;PRIME THE DAC OUTPUT VALUE
      005666  012777  007777  173452              MOV     #7777,@DAC1        ;PRIME THE DAC OUTPUT STAGE
      005674  012777  000000  173414              MOV     #0,@STREG          ;INITIILIZE THE A/D STATUS REG
      005702  017700  173414                      MOV     @ADBUFF,R0         ;READ A/D VALUE AND CLEAR A/D DONE FLAG

      005706  004537  007710              1$:     JSR     R5,CONVRT          ;GET THE VALUE OF CHANNEL 15
      005712  000015                              CHAN15
      005714  004537  010046                      JSR     R5,COMPAR          ;COMPARE AGAINST EXPECTED D/A VALUE
      005720  000000                      2$:     0
      005722  001354                              VWRAP                      ;SPREAD ALLOWED
      005724  000424                              BR      10$                ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
      005726  062737  000010  005720              ADD     #10,2$             ;UPDATE THE D/A OUTPUT VALUE
      005734  013737  005720  005774              MOV     2$,7$              ;COPY VALUE
      005742  005137  005774                      COM     7$                 ;INVERT DATA
      005746  042737  170000  005774              BIC     #170000,7$         ;REMOVE EXTRA BITS
      005754  013777  005774  173364              MOV     7$,@DAC1           ;UPDATE THE D/A OUTPUT VOLTAGE
      005762  022737  010000  005720              CMP     #10000,2$          ;TEST IF LAST STEP
      005770  001346                              BNE     1$
      005772  000402                              BR      TST45              ;;BR TO NEXT TEST
      005774  000000                      7$:     0
      005776  104004                      10$:    ERROR   4                  ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED

```

```
  840
  841                                       ;;*****************************************************************
  (3)                                       ;*TEST 45        AAV11-C ANALOG WRAPAROUND TEST (DAC ''C'' TO A/D CHAN 16)
  (3)                                       ;;*****************************************************************
  (2)  006000  000004                       TST45:  SCOPE
  (1)  006002  012737  000001  001160               MOV     #1,$TIMES               ;;DO 1 ITERATION
  842                                               ;AAV11-C TEST CONNECTOR IS REQUIRED
  843
  844  006010  005737  001370                       TST     TC2                     ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
  845  006014  001450                               BEQ     TST46                   ;;BR IF NO TEST CONNECTOR
  846  006016  012737  000000  006056               MOV     #0,2$                   ;PRIME THE DAC OUTPUT VALUE
  847  006024  012777  007777  173316               MOV     #7777,@DAC2             ;PRIME THE DAC OUTPUT STAGE
  848  006032  012777  000000  173256               MOV     #0,@STREG               ;INITIILIZE THE A/D STATUS REG
  849  006040  017700  173256                       MOV     @ADBUFF,RO              ;READ A/D VALUE AND CLEAR A/D DONE FLAG
  850
  851  006044  004537  007710               1$:     JSR     R5,CONVRT               ;GET THE VALUE OF CHANNEL 16
  852  006050  000016                               CHAN16
  853  006052  004537  010046                       JSR     R5,COMPAR               ;COMPARE AGAINST EXPECTED D/A VALUE
  854  006056  000000                       2$:     0
  855  006060  001354                               VWRAP                           ;SPREAD ALLOWED
  856  006062  000424                               BR      10$                     ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
  857  006064  062737  000010  006056               ADD     #10,2$                  ;UPDATE THE D/A OUTPUT VALUE
  858  006072  013737  006056  006132               MOV     2$,7$                   ;COPY VALUE
  859  006100  005137  006132                       COM     7$                      ;INVERT DATA
  860  006104  042737  170000  006132               BIC     #170000,7$              ;REMOVE EXTRA BITS
  861  006112  013777  006132  173230               MOV     7$,@DAC2                ;UPDATE THE D/A OUTPUT VOLTAGE
  862  006120  022737  010000  006056               CMP     #10000,2$               ;TEST IF LAST STEP
  863  006126  001346                               BNE     1$
  864  006130  000402                               BR      TST46                   ;;BR TO NEXT TEST
  865  006132  000000                       7$:     0
  866  006134  104004                       10$:    ERROR   4                       ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
  867
```

```
 869
 870                                                ;;*************************************************************
 (3)                                                ;*TEST 46           AAV11-C ANALOG WRAPAROUND TEST (DAC ''D'' TO A/D CHAN 17)
 (3)                                                ;;*************************************************************
 (2)   006136  000004                               TST46:  SCOPE
 (1)   006140  012737  000001  001160                       MOV     #1,$TIMES          ;;DO 1 ITERATION
 871                                                        ;AAV11-C TEST CONNECTOR IS REQUIRED
 872   006146  005737  001370                               TST     TC2                ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
 873   006152  001450                                       BEQ     TST47              ;;BR IF NO TEST CONNECTOR
 874   006154  012737  000000  006214                       MOV     #0,2$              ;PRIME THE DAC OUTPUT VALUE
 875   006162  012777  007777  173162                       MOV     #7777,@DAC3        ;PRIME THE DAC OUTPUT STAGE
 876   006170  012777  000000  173120                       MOV     #0,@STREG          ;INITIILIZE THE A/D STATUS REG
 877   006176  017700  173120                               MOV     @ADBUFF,R0         ;READ A/D VALUE AND CLEAR A/D DONE FLAG
 878
 879   006202  004537  007710                       1$:     JSR     R5,CONVRT          ;GET THE VALUE OF CHANNEL 17
 880   006206  000017                                       CHAN17
 881   006210  004537  010046                               JSR     R5,COMPAR          ;COMPARE AGAINST EXPECTED D/A VALUE
 882   006214  000000                               2$:     0
 883   006216  001354                                       VWRAP                      ;SPREAD ALLOWED
 884   006220  000424                                       BR      10$                ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
 885   006222  062737  000010  006214                       ADD     #10,2$             ;UPDATE THE D/A OUTPUT VALUE
 886   006230  013737  006214  006270                       MOV     2$,7$              ;COPY DATA
 887   006236  005137  006270                               COM     7$                 ;INVERT DATA
 888   006242  042737  170000  006270                       BIC     #170000,7$         ;REMOVE EXTRA BITS
 889   006250  013777  006270  173074                       MOV     7$,@DAC3           ;UPDATE THE D/A OUTPUT VOLTAGE
 890   006256  022737  010000  006214                       CMP     #10000,2$          ;TEST IF LAST STEP
 891   006264  001346                                       BNE     1$
 892   006266  000402                                       BR      TST47              ;;BR TO NEXT TEST
 893   006270  000000                               7$:     0
 894   006272  104004                               10$:    ERROR   4                  ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
 895                                                ;;*************************************************************
 (3)                                                ;*TEST 47           END OF AAV11-C TO AXV11-C ANALOG WRAPAROUND
 (3)                                                ;;*************************************************************
 (2)   006274  000004                               TST47:  SCOPE
 (1)   006276  012737  000001  001160                       MOV     #1,$TIMES          ;;DO 1 ITERATION
 896   006304  000207                                       RTS     PC                 ;EXIT AND RETURN TO CALLING ROUTINE
 904
```

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

     006306  005077  173004         IOTST1: CLR     @STREG              ;CLEAR STATUS REGISTER
     006312  104401  010376                 TYPE    ,MSI01              ;TYPE OUT HEADING
     006316  005046                         CLR     -(SP)               ;CLEAR PSW
     006320  012746  006326                 MOV     #77$,-(SP)
     006324  000002                         RTI
     006326  104401  011122         77$:    TYPE    ,CCHAN              ;ASK OPERATOR FOR CHANNEL
     006332  104413                         RDOCT
     006334  012637  006422                 MOV     (SP)+,10$           ;GET ANSWER
     006340  042737  177760  006422         BIC     #177760,10$         ;REMOVE EXTRA BITS
     006346  104401  011162                 TYPE    ,GCHAN              ;ASK OPERATOR FOR GAIN
     006352  104413                         RDOCT
     006354  012637  010044                 MOV     (SP)+,OTHER         ;GET ANSWER
     006360  006137  010044                 ROL     OTHER               ;MOVE TO BITS
     006364  006137  010044                 ROL     OTHER               ;2 + 3
     006370  042737  177763  010044         BIC     #177763,OTHER       ;REMOVE ANY UNWANTED BITS
     006376  104401  011067         1$:     TYPE    ,CH
     006402  013746  006422                 MOV     10$,-(SP)           ;;SAVE 10$ FOR TYPEOUT
(1)                                                                     ;;TYPE CHANNEL
(1)  006406  104403                         TYPOS                       ;;GO TYPE--OCTAL ASCII
(1)  006410     002                         .BYTE   2                   ;;TYPE 2 DIGIT(S)
(1)  006411     000                         .BYTE   0                   ;;SUPPRESS LEADING ZEROS
     006412  012702  000010         2$:     MOV     #10,R2              ;TYPEOUT COUNTER
     006416  004537  007714         3$:     JSR     R5,CONVTR           ;GET AN AVERAGED VALUE FOR THIS CHANNEL
     006422  000000                 10$:    0
     006424  104401  011072         4$:     TYPE    ,SPACE
     006430  013746  001360                 MOV     TEMP,-(SP)          ;;SAVE TEMP FOR TYPEOUT
(1)                                                                     ;;PRINT OCTAL CONVERTED VALUE
(1)  006434  104403                         TYPOS                       ;;GO TYPE--OCTAL ASCII
(1)  006436     004                         .BYTE   4                   ;;TYPE 4 DIGIT(S)
(1)  006437     001                         .BYTE   1                   ;;TYPE LEADING ZEROS
     006440  012701  010000                 MOV     #10000,R1
     006444  005301                 5$:     DEC     R1
     006446  001376                         BNE     5$
     006450  005302                         DEC     R2                  ;DECREMENT THE COUNTER
     006452  001361                         BNE     3$                  ;NO CARRIAGE RETURN
     006454  104401  001171                 TYPE    ,$CRLF              ;CARRIAGE RETURN
     006460  000746                         BR      1$                  ;REPEAT CONVERSION
```

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

006462  104401  010454           IOTST2: TYPE    ,MSI02                  ;TELL OPERATOR THE SECTION NAME

006466  005002                           CLR     R2                      ;INITILIZE THE CHANNEL SCANNER
006470  005003                           CLR     R3                      ;INITILIZE THE GAIN SELECT VALUE

006472  104401  001171           1$:     TYPE    ,$CRLF                  ;MAKE A FRESH OUTPUT LINE
006476  012704  000007                   MOV     #7,R4                   ;LOAD LINE WIDTH COUNTER

006502  104401  011067                   TYPE    ,CH                     ;SHOW "CH" TEXT

006506  010246                           MOV     R2,-(SP)                ;LOAD THE CHANNEL CODE
006510  104403                           TYPOS
006512     002     001                   .BYTE   2,1

006514  104401  011114                   TYPE    ,ADOT                   ;SEPERATE CH FROM GS

006520  112737  000060  011116           MOVB    #'0,AZERO               ;LOAD ASCII 0
006526  132703  000010                   BITB    #10,R3                  ;TEST IF GS1 = 1
006532  001402                           BEQ     2$                      ;BR IF NOT SET
006534  105237  011116                   INCB    AZERO                   ;MAKE IT A ONE
006540  104401  011116           2$:     TYPE    ,AZERO                  ;REPORT GS1 STATUS

006544  112737  000060  011116           MOVB    #'0,AZERO               ;LOAD ASCII 0
006552  132703  000004                   BITB    #4,R3                   ;TEST IF GS0 = 1
006556  001402                           BEQ     3$                      ;BR IF NOT SET
006560  105237  011116                   INCB    AZERO                   ;MAKE IT A ONE
006564  104401  011116           3$:     TYPE    ,AZERO                  ;REPORT GS0 STATUS

006570  010200                           MOV     R2,R0                   ;GET CURRENT CHANNEL VALUE
006572  000300                           SWAB    R0                      ;MOVE TO MUX POSITION
006574  050300                           BIS     R3,R0                   ;ADD THE GAIN SELECT BITS
006576  010077  172514                   MOV     R0,@STREG               ;SELECT MUX AND GAIN BITS
006602  105277  172510           4$:     INCB    @STREG                  ;START CONVERSION
006606  105777  172504           5$:     TSTB    @STREG                  ;WAIT FOR A/D DONE
006612  100375                           BPL     5$

006614  104401  011072                   TYPE    ,SPACE                  ;ENSURE SOME OUTPUT ROOM
006620  017746  172476                   MOV     @ADBUFF,-(SP)           ;READ CONVERTED VALUE AND SAVE FOR TYPOUT
006624  104403                           TYPOS
006626     004     001                   .BYTE   4,1

006630  105304                           DECB    R4                      ;FINISHED A LINE ACROSS THE PAGE
006632  001363                           BNE     4$                      ;BR AND CONVERT WITH CURRENT GAIN AND CHANNEL

006634  005202                           INC     R2                      ;BUMP CHANNEL VALUE
006636  062703  000004                   ADD     #4,R3                   ;BUMP GAIN SELECT VALUE
006642  042703  177763                   BIC     #177763,R3              ;REMOVE EXTRA BITS
006646  122702  000020                   CMPB    #20,R2                  ;TEST IS LAST CHANNEL
006652  001307                           BNE     1$                      ;BR IF NOT
006654  005002                           CLR     R2                      ;INITILIZE THE CHANNEL
006656  104401  001171                   TYPE    ,$CRLF                  ;INSERT ANOTHER FRESH OUTPUT LINE
006662  000703                           BR      1$                      ;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

006664  104401  010514           IOTST3:  TYPE    ,MSI03             ;TELL OPERATOR THE NAME
006670  104401  011122                    TYPE    ,CCHAN             ;ASK OPER. FOR THE CHANNEL
006674  104413                            RDOCT
006676  012637  006744                    MOV     (SP)+,10$
006702  042737  177760  006744            BIC     #177760,10$        ;REMOVE EXTRA BITS
006710  104401  011162                    TYPE    ,GCHAN             ;ASK OPER FOR THE GAIN SELECT VALUE
006714  104413                            RDOCT
006716  012637  010044                    MOV     (SP)+,OTHER        ;GET THE ANSWER
006722  006337  010044                    ASL     OTHER              ;MOVE INTO
006726  006337  010044                    ASL     OTHER              ;GAIN SELECT POSITION
006732  042737  177763  010044            BIC     #177763,OTHER      ;REMOVE EXTRA BITS

006740  004537  007714           4$:      JSR     R5,CONVTR          ;CONVERT SELECTED CHANNEL AND GAIN
006744  000000                   10$:     0

006746  042737  170000  001360            BIC     #170000,TEMP       ;REMOVE EXTRA BITS
006754  013777  001360  172342            MOV     TEMP,@DACA         ;LOAD DAC "A"
006762  013777  001360  172336            MOV     TEMP,@DACB         ;LOAD DAC "B"

006770  000763                            BR      4$                 ;LOOP BACK AND REPEAT

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

006772  104401  010557           IOTST4:  TYPE    ,MSI04             ;TELL OPERATOR THE NAME
006776  012703  000000                    MOV     #0,R3              ;LOAD DAC - F.S. VALUE
007002  012704  007777                    MOV     #7777,R4           ;LOAD DAC + F.S. VALUE

007006  012705  010000           1$:      MOV     #BIT12,R5          ;LOAD LOOP COUNT
007012  010377  172306           2$:      MOV     R3,@DACA           ;LOAD DAC "A"
007016  010477  172304                    MOV     R4,@DACB           ;LOAD DAC "B"
007022  005305                            DEC     R5                 ;FINISHED ALL BITS ?
007024  001403                            BEQ     3$                 ;BR IF DONE
007026  005304                            DEC     R4                 ;LOWER DAC "B" VALUE
007030  005203                            INC     R3                 ;RAISE DAC "A" VALUE
007032  000767                            BR      2$                 ;DO NEXT COUNT

007034  012705  010000           3$:      MOV     #BIT12,R5          ;LOAD LOOP COUNT
007040  010377  172260           4$:      MOV     R3,@DACA           ;LOAD DAC "A"
007044  010477  172256                    MOV     R4,@DACB           ;LOAD DAC "B"
007050  005305                            DEC     R5                 ;FINISHED ALL BITS ?
007052  001755                            BEQ     1$
007054  005303                            DEC     R3                 ;LOWER DAC "A" VALUE
007056  005204                            INC     R4                 ;RAISE DAC "B" VALUE
007060  000767                            BR      4$                 ;DO NEXT COUNT
```

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

007062  104401  010632         IOTST5: TYPE    ,MSI05          ;TELL OPERATOR THE NAME
007066  012703  000000                 MOV     #0,R3           ;LOAD DAC - F.S. VALUE
007072  012704  007777                 MOV     #7777,R4        ;LOAD DAC + F.S. VALUE
007076  012705  004000                 MOV     #4000,R5        ;LOAD 0.0 F.S. VALUE

007102  010377  172216         1$:     MOV     R3,@DACA        ;LOAD DAC "A" TO - F.S.
007106  010377  172214                 MOV     R3,@DACB        ;LOAD DAC "B" TO - F.S.
007112  104412                         RDLIN
007114  012600                         MOV     (SP)+,R0        ;REMOVE CHARACTER
007116  010477  172202                 MOV     R4,@DACA        ;LOAD DAC "A" TO + F.S.
007122  010477  172200                 MOV     R4,@DACB        ;LOAD DAC "B" TO + F.S.
007126  104412                         RDLIN
007130  012600                         MOV     (SP)+,R0        ;REMOVE CHARACTER
007132  010577  172166                 MOV     R5,@DACA        ;LOAD DAC "A" TO MID POINT
007136  010577  172164                 MOV     R5,@DACB        ;LOAD DAC "B" TO MID POINT
007142  104412                         RDLIN
007144  012600                         MOV     (SP)+,R0        ;REMOVE CHARACTER
007146  000755                         BR      1$

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

007150  104401  010677         IOTST6: TYPE    ,MSI06          ;TELL OPERATOR THE NAME
007154  012703  000000                 MOV     #0,R3           ;LOAD DAC - F.S.
007160  012704  007777                 MOV     #7777,R4        ;LOAD DAC + F.S.

007164  010377  172134         1$:     MOV     R3,@DACA        ;LOAD DAC "A" TO MIN LEVEL
007170  010377  172132                 MOV     R3,@DACB        ;LOAD DAC "B" TO MIN LEVEL
007174  004737  001426                 JSR     PC,STALL        ;DELAY
007200  010477  172120                 MOV     R4,@DACA        ;LOAD DAC "A" TO MAX LEVEL
007204  010477  172116                 MOV     R4,@DACB        ;LOAD DAC "B" TO MAX LEVEL
007210  004737  001426                 JSR     PC,STALL        ;DELAY
007214  000763                         BR      1$              ;LOOP BACK AND DO AGAIN

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

007216  104401  010770         IOTST7: TYPE    ,MSI07          ;TELL OPERATOR THE SUB-SECTION NAME
007222  005003                         CLR     R3              ;INITILIZE THE DAC VALUE
007224  104401  001171         1$:     TYPE    ,$CRLF          ;ENSURE FRESH OUTPUT LINE
007230  012705  000010                 MOV     #10,R5          ;LOAD LINE WIDTH COUNTER

007234  105277  172056         2$:     INCB    @STREG          ;START CONVERSION
007240  105777  172052         3$:     TSTB    @STREG          ;WAIT FOR A/D DONE
007244  100375                         BPL     3$
007246  010377  172052                 MOV     R3,@DACA        ;LOAD "DAC A" OUTPUT VALUE
007252  017746  172044                 MOV     @ADBUFF,-(SP)   ;READ AND STORE A/D VALUE
007256  104403                         TYPOS
007260     004     001                 .BYTE   4,1
007262  005203                         INC     R3              ;UPDATE TO NEXT D/A VALUE
007264  042703  170000                 BIC     #170000,R3      ;ENSURE ONLY 12 BITS LONG
007270  005305                         DEC     R5              ;IS THE WIDTH FINISHED ?
007272  001754                         BEQ     1$              ;BR AND START FRESH OUTPUT LINE
007274  104401  011072                 TYPE    ,SPACE          ;ENSURE SOME ROOM
007300  000755                         BR      2$              ;AND DO ANOTHER CONVERSION
```

```

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

007302                                  BEGINL:
007302  004737  002530                  1$:     JSR     PC,BEGL                 ;LOGIC TESTS
007306  012737  007302  010252                  MOV     #1$,AGTST               ;ADDRESS FOR EOP
007314  000137  010254                          JMP     $EOP                    ;TYPE END OF PASS

                                        .SBTTL        AUTO TEST
007320                                  BEGINA:
007320  004737  002530                  1$:     JSR     PC,BEGL                 ;LOGIC TESTS
007324  004737  004104                          JSR     PC,WRAP
007330  012737  007320  010252                  MOV     #1$,AGTST               ;ADDRESS FOR EOP
007336  000137  010254                          JMP     $EOP                    ;TYPE END OF PASS

                                        .SBTTL        WRAPAROUND TEST
007342                                  BEGINW:
007342  004737  004104                  1$:     JSR     PC,WRAP                 ;WRAPAROUND TESTS
007346  012737  007342  010252                  MOV     #1$,AGTST
007354  000137  010254                          JMP     $EOP                    ;INCREMENTS $PASS

                                        .SBTTL          DMT TEST STARTUP
007360  032737  000001  001252          BEGIND: BIT     #BIT0,$DEVM     ;TEST IF KWV11-C CONNECTED TO RTC TRIGGER
007366  001402                                  BEQ     1$              ;BR IF NOT
007370  005237  001374                          INC     KWAD            ;SET KW CONNECTED TO AD RTC TRIG - FLAG
007374  032737  000002  001252          1$:     BIT     #BIT1,$DEVM     ;TEST IF KWV11-C CONNECTED TO EXT TRIG AND ''F2''
007402  001402                                  BEQ     2$              ;BR IF NOT
007404  005237  001376                          INC     KWEX            ;SET KW CONNECTED TO AD EXT TRIG - FLAG
007410  032737  000004  001252          2$:     BIT     #BIT2,$DEVM     ;TEST IF TEST FIXTURE CONNECTED
007416  001402                                  BEQ     3$              ;BR IF NOT
007420  005237  001366                          INC     TC1             ;SET TEST FIXTURE PRESENT FLAG
007424  032737  000010  001252          3$:     BIT     #BIT3,$DEVM     ;TEST IF AAV11-C CONNECTED TO TEST FIXTURE
007432  001402                                  BEQ     4$              ;BR IF NOT
007434  005237  001370                          INC     TC2             ;SET AAV11-C ANALOG WRAPAROUND FLAG
007440  032737  000020  001252          4$:     BIT     #BIT4,$DEVM     ;TEST IF BEVENT AND ''F1'' CONNECTED
007446  001402                                  BEQ     5$              ;BR IF NOT
007450  005237  001402                          INC     BTEX            ;SET BEVENT AND ''F1'' FLAG
007454  032737  000040  001252          5$:     BIT     #BIT5,$DEVM     ;TEST IF MODULE IS AN ''ADV11-C''
007462  001402                                  BEQ     6$              ;BR IF NOT
007464  005237  001372                          INC     ADV11C          ;SET ''ADV11-C'' FLAG
007470  000240                          6$:     NOP
007472  000240                                  NOP
007474  000240                                  NOP
007476  000240                                  NOP
007500  000240                                  NOP
007502  000137  007320                          JMP     BEGINA          ;RUN THE ''AUTO-MODE'' TESTS
```

```
                                         .SBTTL  ROUTINE TO INITILIZE THE BUS AND VECTOR ADDRESSES
007506  012737  000006  000004   FIXONE: MOV     #6,@#ERRVEC                     ;SET UP ERRVEC
007514  013737  001250  001316           MOV     $BASE,STREG                     ;RELOAD INITIAL ADDRESSES
007522  013737  001250  001320           MOV     $BASE,ADST1
007530  013737  001250  001322           MOV     $BASE,ADBUFF
007536  013737  001250  001324           MOV     $BASE,DACA                      ;PRIME DAC ''A'' ADDRESS
007544  013737  001250  001326           MOV     $BASE,DACB                      ;           ''B''
007552  005237  001320                   INC     ADST1
007556  062737  000002  001322           ADD     #2,ADBUFF
007564  062737  000004  001324           ADD     #4,DACA
007572  062737  000006  001326           ADD     #6,DACB
007600  013737  001244  001330           MOV     $VECT1,VECTOR
007606  042737  170000  001330           BIC     #170000,VECTOR
007614  013737  001330  001332           MOV     VECTOR,VECTR1
007622  062737  000002  001332           ADD     #2,VECTR1
007630  013737  001330  001334           MOV     VECTOR,VECTR2
007636  062737  000004  001334           ADD     #4,VECTR2
007644  013737  001330  001336           MOV     VECTOR,VECTR3
007652  062737  000006  001336           ADD     #6,VECTR3
                                 ;;LOAD .+2 AND HALT TRAP CATCH;;
007660  012700  000216                   MOV     #216,R0                         ;FILL .+2
007664  012701  000214                   MOV     #214,R1                         ;LOAD HALT
007670  010021                   1$:     MOV     R0,(R1)+
007672  005021                           CLR     (R1)+
007674  010100                           MOV     R1,R0
007676  005720                           TST     (R0)+
007700  020027  001002                   CMP     R0,#1002
007704  001371                           BNE     1$
007706  000207                           RTS     PC                              ;TEST NEXT A/D
```

```
                                ;;ROUTINE TO AVERAGE 8 CONVERSIONS;;
007710  005037  010044          CONVRT: CLR     OTHER              ;REMOVE EXTRA BITS
007714  012500                  CONVTR: MOV     (R5)+,R0           ;GET CHANNEL VALUE
007716  010037  001362                  MOV     R0,CHANL
007722  000300                          SWAB    R0
007724  053700  010044                  BIS     OTHER,R0           ;ADD GAIN SELECT IF NEEDED
007730  005037  001360                  CLR     TEMP
007734  010077  171356                  MOV     R0,@STREG          ;LOAD CHANNEL INTO MIX BITS
007740  012700  010000                  MOV     #10000,R0
007744  005300                  2$:     DEC     R0
007746  001376                          BNE     2$
007750  012777  001440  171352          MOV     #RETURN,@VECTOR ;LOAD VECTOR
007756  012700  000010                  MOV     #10,R0             ;SET UP COUNTER
007762  152777  000101  171326  1$:     BISB    #101,@STREG        ;SET INTRPT. EN., START CONV.
007770  000001                          WAIT                       ;WAIT FOR CONVERSION
007772  017737  171324  010042          MOV     @ADBUFF,77$        ;READ CONVERTED VALUE
010000  042737  170000  010042          BIC     #170000,77$        ;REMOVE HIGH BITS
010006  063737  010042  001360          ADD     77$,TEMP           ;READ BUFFER
010014  005300                          DEC     R0
010016  001361                          BNE     1$                 ;DO 8 TIMES
010020  006237  001360                  ASR     TEMP               ;AVERAGE VALUE
010024  006237  001360                  ASR     TEMP
010030  006237  001360                  ASR     TEMP
010034  005537  001360                  ADC     TEMP
010040  000205                          RTS     R5                 ;RETURN
010042  000000                  77$:    0
010044  000000                  OTHER:  0

                                ;COMPARE $GDDAT AND $BDDAT;;
010046  012537  001124          COMPAR: MOV     (R5)+,$GDDAT       ;GET GOOD DATA
010052  013537  001364                  MOV     @(R5)+,SPREAD      ;GET SPREAD
010056  013737  001360  001126          MOV     TEMP,$BDDAT        ;GET BAD(ACTUAL) DATA
010064  013700  001124                  MOV     $GDDAT,R0
010070  163700  001126                  SUB     $BDDAT,R0          ;GET DIFFERENCE
010074  100001                          BPL     7$
010076  005400                          NEG     R0
010100  020037  001364          7$:     CMP     R0,SPREAD          ;COMPARE IT TO SPREAD
010104  003001                          BGT     10$                ;GO TO ERROR PRINTOUT
010106  005725                          TST     (R5)+              ;BUMP RETURN POINTER AROUND ERROR CALL
010110  000205                  10$:    RTS     R5
```

```
                                          ;;SUBROUTINE TO TYPE INTRPT. TST MSG.;;
   010112  005737  001202                 DUMW:   TST     $PASS
   010116  001021                                 BNE     20$
   010120  012737  010162  001110                 MOV     #20$,$LPERR
   010126  012737  010162  001106                 MOV     #20$,$LPADR
   010134  104401  011463                         TYPE    ,METST                ;TYPE ASCIZ STRING
   010140  010046                                 MOV     R0,-(SP)              ;;SAVF R0 FOR TYPEOUT
(1)                                                                             ;;TYPE TEST NO.
(1) 010142  104403                                TYPOS                         ;;GO TYPE--OCTAL ASCII
(1) 010144     002                                .BYTE   2                     ;;TYPE 2 DIGIT(S)
(1) 010145     000                                .BYTE   0                     ;;SUPPRESS LEADING ZEROS
   010146  104401  011336                         TYPE    ,ONAD
   010152  013746  001316                         MOV     STREG,-(SP)           ;;SAVE STREG FOR TYPEOUT
(1)                                                                             ;;TYPE BUS ADDRESS
(1) 010156  104403                                TYPOS                         ;;GO TYPE--OCTAL ASCII
(1) 010160     006                                .BYTE   6                     ;;TYPE 6 DIGITS
(1) 010161     001                                .BYTE   1                     ;;TYPE LEADING ZEROS
   010162  000207                         20$:    RTS     PC

   010164  005737  001202                 DUMC:   TST     $PASS
   010170  001010                                 BNE     30$
   010172  012737  010212  001110                 MOV     #30$,$LPERR
   010200  012737  010212  001106                 MOV     #30$,$LPADR
   010206  104401  011101                         TYPE    ,DONE
   010212  000207                         30$:    RTS     PC

                                          ;SUBROUTINE TO RESET & SET INTRPT. EN.;
   010214  000005                         RST:    RESET
   010216  052777  000100  170720                 BIS     #100,@$TKS
   010224  005046                                 CLR     -(SP)                 ;CLEAR PSW
   010226  012746  010234                         MOV     #1$,-(SP)
   010232  000002                                 RTI
   010234  000207                         1$:     RTS     PC


   010236  000002                         V2:     2
   010240  000012                         V12:    12

   010242  052777  000100  170674         AGATST: BIS     #100,@$TKS
   010250  000137                                 JMP     @(PC)+
   010252  001522                         AGTST:  BEGIN0
```

```
                                        .SBTTL  END OF PASS ROUTINE
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;*INCREMENT THE PASS NUMBER ($PASS)
 (1)                                    ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
 (1)                                    ;*IF THERES A MONITOR GO TO IT
 (1)                                    ;*IF THERE ISN'T JUMP TO AGATST
 (1)
 (1) 010254                             $EOP:
 (2) 010254  000240                             NOP
 (1) 010256  005037  001102                     CLR     $TSTNM              ;;ZERO THE TEST NUMBER
 (1) 010262  005037  001160                     CLR     $TIMES              ;;ZERO THE NUMBER OF ITERATIONS
 (1) 010266  005237  001202                     INC     $PASS               ;;INCREMENT THE PASS NUMBER
 (1) 010272  042737  100000  001202             BIC     #100000,$PASS       ;;DON'T ALLOW A NEG. NUMBER
 (1) 010300  005327                             DEC     (PC)+               ;;LOOP?
 (1) 010302  000001                     $EOPCT: .WORD   1
 (1) 010304  003022                             BGT     $DOAGN              ;;YES
 (1) 010306  012737                             MOV     (PC)+,@(PC)+        ;;RESTORE COUNTER
 (1) 010310  000001                     $ENDCT: .WORD   1
 (1) 010312  010302                             $EOPCT
 (1) 010314  104401  010361                     TYPE    ,$ENDMG             ;;TYPE "END PASS #"
 (2) 010320  013746  001202                     MOV     $PASS,-(SP)         ;;SAVE $PASS FOR TYPEOUT
 (2) 010324  104405                             TYPDS                       ;;GO TYPE--DECIMAL ASCII WITH SIGN
 (1) 010326  104401  010356                     TYPE    ,$ENULL             ;;TYPE A NULL CHARACTER
 (1) 010332  013700  000042             $GET42: MOV     @#42,R0             ;;GET MONITOR ADDRESS
 (1) 010336  001405                             BEQ     $DOAGN              ;;BRANCH IF NO MONITOR
 (1) 010340  000005                             RESET                       ;;CLEAR THE WORLD
 (1) 010342  004710                     $ENDAD: JSR     PC,(R0)             ;;GO TO MONITOR
 (1) 010344  000240                             NOP                         ;;SAVE ROOM
 (1) 010346  000240                             NOP                         ;;FOR
 (1) 010350  000240                             NOP                         ;;ACT11
 (1) 010352                             $DOAGN:
 (1) 010352  000137                             JMP     @(PC)+              ;;RETURN
 (1) 010354  010242                     $RTNAD: .WORD   AGATST
 (1) 010356     377     377     000     $ENULL: .BYTE   -1,-1,0             ;;NULL CHARACTER STRING
 (1) 010361     015  042412  042116     $ENDMG: .ASCIZ  <15><12>/END PASS #/
 (1) 010366  050040  051501  020123
 (1) 010374  000043
```

```
1256                                              .SBTTL          ASCII MESSAGES
1257   010376   020200   042522   047520   MSI01:   .ASCIZ   <200>\ REPORTING CONVERTED A TO D CHANNEL VALUES \<200>
       010404   052122   047111   020107
       010412   047503   053116   051105
       010420   042524   020104   020101
       010426   047524   042040   041440
       010434   040510   047116   046105
       010442   053040   046101   042525
       010450   020123   000200
1258   010454   020200   041523   047101   MSI02:   .ASCIZ   <200>\ SCANNING CHANNELS AND GAINS \<200>
       010462   044516   043516   041440
       010470   040510   047116   046105
       010476   020123   047101   020104
       010504   040507   047111   020123
       010512   000200
1259   010514   020200   027501   020104   MSI03:   .ASCIZ   <200>\ A/D INPUT ECHOED TO D/A OUTPUTS\<200>
       010522   047111   052520   020124
       010530   041505   047510   042105
       010536   052040   020117   027504
       010544   020101   052517   050124
       010552   052125   100123      000
1260   010557      200   047440   052125   MSI04:   .ASCIZ   <200>\ OUTPUT A RAMP ON DAC "A" AND "B" OUTPUT\<200>
       010564   052520   020124   020101
       010572   040522   050115   047440
       010600   020116   040504   020103
       010606   040442   020042   047101
       010614   020104   041042   020042
       010622   052517   050124   052125
       010630   000200
1261   010632   020200   040503   044514   MSI05:   .ASCIZ   <200>\ CALIBRATE THE AXV11-C D/A OUTPUTS\<200>
       010640   051102   052101   020105
       010646   044124   020105   054101
       010654   030526   026461   020103
       010662   027504   020101   052517
       010670   050124   052125   100123
       010676      000
1262   010677      200   047440   052125   MSI06:   .ASCIZ   <200>\ OUTPUT SQUARE WAVES ON AXV11-C DAC "A" AND "B" OUTPUT\<200>
       010704   052520   020124   050523
       010712   040525   042522   053440
       010720   053101   051505   047440
       010726   020116   054101   030526
       010734   026461   020103   040504
       010742   020103   040442   020042
       010750   047101   020104   041042
       010756   020042   052517   050124
       010764   052125   000200
1263   010770   020200   054101   030526   MSI07:   .ASCIZ   <200>\ AXV11-C D/A OUTPUT ECHOED TO A/D INPUT\<200>
       010776   026461   020103   027504
       011004   020101   052517   050124
       011012   052125   042440   044103
       011020   042517   020104   047524
       011026   040440   042057   044440
       011034   050116   052125   000200
1264   011042      136      103      040   CMSG:    .BYTE    136,103,40,40,0           ;CONTROL C ECHO
       011045      040      000
1265   011047      136      101      040   AMSG:    .BYTE    136,101,40,40,0           ;CONTROL A ECHO
```

```
      011052     040     000
1266  011054     136     107     015  GMSG:    .BYTE   136,107,15,12,123,127,122,105,107,72,0  ;CONTROL G ECHO
      011057     012     123     127
      011062     122     105     107
      011065     072     000
1267  011067     103  000110          CH:      .ASCIZ  /CH/
1268  011072     040     040     040  SPACE:   .BYTE   40,40,40,40,0
      011075     040     000
1269  011077     077     000          QUEST:   .BYTE   77,0
1270  011101     040  020040  042040  DONE:    .ASCIZ  /     DONE/<15><12>
      011106  047117  006505  000012
1271  011114  000056                  ADOT:    .ASCIZ  \.\
1272  011116  000060                  AZERO:   .ASCIZ  \0\
1273  011120  000057                  SLASH:   .ASCIZ  #/#
1274  011122  005015  051525  047111  CCHAN:   .ASCIZ  <15><12>/USING OCTAL CHANNEL (0-17) ? /
      011130  020107  041517  040524
      011136  020114  044103  047101
      011144  042516  020114  030050
      011152  030455  024467  037440
      011160  000040
1275  011162  005015  051525  047111  GCHAN:   .ASCIZ  <15><12>/USING GAIN SELECT VALUE OF (0-3) ? /
      011170  020107  040507  047111
      011176  051440  046105  041505
      011204  020124  040526  052514
      011212  020105  043117  024040
      011220  026460  024463  037440
      011226  000040
1276  011230  005015  047105  044504  ECHAN:   .ASCIZ  <15><12>/ENDING WITH OCTAL CHANNEL (0-17) ? /
      011236  043516  053440  052111
      011244  020110  041517  040524
      011252  020114  044103  047101
      011260  042516  020114  030050
      011266  030455  024467  037440
      011274  000040
1277  011276  005015  042504  051120  CRWR:    .ASCIZ  <15><12>/DEPRESS "RETURN" WHEN READY/<15><12>
      011304  051505  020123  051042
      011312  052105  051125  021116
      011320  053440  042510  020116
      011326  042522  042101  006531
      011334  000012
1278  011336  047440  020116  054101  ONAD:    .ASCIZ  \ ON AXV/ADV11-C AT BUS ADDRESS  \
      011344  027526  042101  030526
      011352  026461  020103  052101
      011360  041040  051525  040440
      011366  042104  042522  051523
      011374  020040     000
1279  011377     015  052012  050131  MSG71:   .ASCIZ  <15><12>/TYPE LETTER AND DEPRESS "RETURN" /
      011404  020105  042514  052124
      011412  051105  040440  042116
      011420  042040  050105  042522
      011426  051523  021040  042522
      011434  052524  047122  020042
      011442     000
1280  011443     015  050012  044522  HEAD5:   .ASCII  <15><12>/PRINT VALUES--/
      011450  052116  053040  046101
      011456  042525  026523     055
```

```
 1281 011463     015  020012  047105  METST:   .ASCIZ  <15><12>/ ENTERING TEST /
      011470  042524  044522  043516
      011476  052040  051505  020124
      011504     000
 1282 011505     015     012          MSKWAD: .BYTE   15,12
 1283 011507     111  020123  053513          .ASCIZ  \IS KWV11-C CONNECTED TO 'RTC IN'' (J1-PIN 21) ? \
      011514  030526  026461  020103
      011522  047503  047116  041505
      011530  042524  020104  047524
      011536  021040  052122  020103
      011544  047111  020042  045050
      011552  026461  044520  020116
      011560  030462  020051  020077
      011566     000
 1284 011567     015     012          MSKWEX: .BYTE   15,12
 1285 011571     111  020123  053513          .ASCIZ  \IS KWV11-C CONNECTED TO 'EXT TRIG'' (J1-PIN 19 AND 'F2'' INSTALLED) ? \
      011576  030526  026461  020103
      011604  047503  047116  041505
      011612  042524  020104  047524
      011620  021040  054105  020124
      011626  051124  043511  020042
      011634  045050  026461  044520
      011642  020116  034461  040440
      011650  042116  021040  031106
      011656  020042  047111  052123
      011664  046101  042514  024504
      011672  037440  000040
 1286 011676     015     012          MSMAEX: .BYTE   15,12
 1287 011700  051511  040440  046440          .ASCIZ  \IS A MANUAL TRIGGER CONNECTED TO 'EXT TRIG'' (J1-PIN 19 AND 'F2'' INSTALL
      011706  047101  040525  020114
      011714  051124  043511  042507
      011722  020122  047503  047116
      011730  041505  042524  020104
      011736  047524  021040  054105
      011744  020124  051124  043511
      011752  020042  045050  026461
      011760  044520  020116  034461
      011766  040440  042116  021040
      011774  031106  020042  047111
      012002  052123  046101  042514
      012010  024504  037440  000040
 1288 012016     015     012          MSGNEX: .BYTE   15,12
 1289 012020  042507  042516  040522          .ASCIZ  \GENERATE ONE TRIGGER SIGNAL\
      012026  042524  047440  042516
      012034  052040  044522  043507
      012042  051105  051440  043511
      012050  040516  000114
 1290 012054     015     012          MSBTEX: .BYTE   15,12
 1291 012056  051511  021040  020102          .ASCIZ  \IS 'B EVENT'' CONNECTED TO 'EXT TRIG'' ('F1'' INSTALLED) ? \
      012064  053105  047105  021124
      012072  041440  047117  042516
      012100  052103  042105  052040
      012106  020117  042442  052130
      012114  052040  044522  021107
      012122  024040  043042  021061
      012130  044440  051516  040524
```

```
        012136  046114  042105  020051
        012144  020077     000
 1292   012147     200  051511  052040  MSADV:  .ASCIZ  <200>\IS THIS AN ADV11-C ? \
        012154  044510  020123  047101
        012162  040440  053104  030461
        012170  041455  037440  000040
 1293   012176     015     012          MSTC1:  .BYTE   15,12
 1294   012200  051511  052040  042510          .ASCIZ  \IS THE AXV/ADV11-C TEST FIXTURE INSTALLED ? \
        012206  040440  053130  040457
        012214  053104  030461  041455
        012222  052040  051505  020124
        012230  044506  052130  051125
        012236  020105  047111  052123
        012244  046101  042514  020104
        012252  020077     000
 1295   012255     015     012          MSTC2:  .BYTE   15,12
 1296   012257     111  020123  044124          .ASCIZ  \IS THE AAV11-C TO AXV/ADV11-C TEST CABLE INSTALLED ? \
        012264  020105  040501  030526
        012272  026461  020103  047524
        012300  040440  053130  040457
        012306  053104  030461  041455
        012314  052040  051505  020124
        012322  040503  046102  020105
        012330  047111  052123  046101
        012336  042514  020104  020077
        012344     000
 1297   012345     015     012          MSG70:  .BYTE   15,12
 1298   012347     015  040412  020072          .ASCII  <15><12>/A: AUTOMATED RUNNING OF LOGIC AND ANALOG WRAPAROUND TESTS/
        012354  052501  047524  040515
        012362  042524  020104  052522
        012370  047116  047111  020107
        012376  043117  046040  043517
        012404  041511  040440  042116
        012412  040440  040516  047514
        012420  020107  051127  050101
        012426  051101  052517  042116
        012434  052040  051505  051524
 1299   012442  005015  035114  046040          .ASCII  <15><12>/L: LOGIC TESTS ONLY/
        012450  043517  041511  052040
        012456  051505  051524  047440
        012464  046116     131
 1300   012467     015  053412  020072          .ASCII  <15><12>/W: WRAPAROUND OF ANALOG TESTS ONLY/
        012474  051127  050101  051101
        012502  052517  042116  047440
        012510  020106  047101  046101
        012516  043517  052040  051505
        012524  051524  047440  046116
        012532     131
 1301   012533     015  030412  020072          .ASCII  <15><12>/1: PRINT VALUES OF SELECTED CHANNEL/
        012540  051120  047111  020124
        012546  040526  052514  051505
        012554  047440  020106  042523
        012562  042514  052103  042105
        012570  041440  040510  047116
        012576  046105
 1302   012600  005015  035062  050040          .ASCII  <15><12>/2: PRINT VALUES OF SCANNED CHANNEL AND GAIN/
```

```
        012606  044522  052116  053040
        012614  046101  042525  020123
        012622  043117  051440  040503
        012630  047116  042105  041440
        012636  040510  047116  046105
        012644  040440  042116  043440
        012652  044501     116
 1303   012655     015  031412  020072           .ASCII  <15><12>/3: AXV11-C A TO D INPUT ECHOED TO D TO A OUTPUT/
        012662  054101  030526  026461
        012670  020103  020101  047524
        012676  042040  044440  050116
        012704  052125  042440  044103
        012712  042517  020104  047524
        012720  042040  052040  020117
        012726  020101  052517  050124
        012734  052125
 1304   012736  005015  035064  040440           .ASCII  <15><12>/4: AXV11-C D TO A RAMP/
        012744  053130  030461  041455
        012752  042040  052040  020117
        012760  020101  040522  050115
 1305   012766  005015  035065  040440           .ASCII  <15><12>/5: AXV11-C D TO A CALIBRATION/
        012774  053130  030461  041455
        013002  042040  052040  020117
        013010  020101  040503  044514
        013016  051102  052101  047511
        013024     116
 1306   013025     015  033012  020072           .ASCII  <15><12>/6: AXV11-C D TO A SQUARE WAVES/
        013032  054101  030526  026461
        013040  020103  020104  047524
        013046  040440  051440  052521
        013054  051101  020105  040527
        013062  042526     123
 1307   013065     015  033412  020072           .ASCII  <15><12>/7: AXV11-C D TO A OUTPUT TO A TO D INPUT/
        013072  054101  030526  026461
        013100  020103  020104  047524
        013106  040440  047440  052125
        013114  052520  020124  047524
        013122  040440  052040  020117
        013130  020104  047111  052520
        013136     124
 1308   013137     015  020012  000040           .ASCIZ  <15><12>/  /
 1309   013144  005015  051511  045440  HEAD2:   .ASCIZ  <15><12>\IS KWV11-C CONNECTED TO AXV/ADV11-C ? \
        013152  053127  030461  041455
        013160  041440  047117  042516
        013166  052103  042105  052040
        013174  020117  054101  027526
        013202  042101  030526  026461
        013210  020103  020077     000
 1310   013215     123  040524  052524  EM1:     .ASCIZ  /STATUS REG. ERROR/
        013222  020123  042522  027107
        013230  042440  051122  051117
        013236     000
 1311   013237     106  044501  042514  EM2:     .ASCIZ  /FAILED TO INTERRUPT/
        013244  020104  047524  044440
        013252  052116  051105  052522
        013260  052120     000
 1312  013263     125  042516  050130  EM3:    .ASCIZ  /UNEXPECTED INTERRUPT/
       013270  041505  042524  020104
       013276  047111  042524  051122
       013304  050125  000124
 1313  013310  051105  047522  020122  EM4:    .ASCIZ  #ERROR ON A/D CHANNEL#
       013316  047117  040440  042057
       013324  041440  040510  047116
       013332  046105     000
 1314  013335     105  051122  041520  DH1:    .ASCIZ  /ERRPC   STREG   EXPECTED ACTUAL/
       013342  020040  051440  051124
       013350  043505  020040  042440
       013356  050130  041505  042524
       013364  020104  041501  052524
       013372  046101     000
 1315  013375     105  051122  041520  DH2:    .ASCIZ  /ERRPC   STREG   CHANNEL NOMINAL SPREAD  ACTUAL/
       013402  020040  051440  051124
       013410  043505  020040  041440
       013416  040510  047116  046105
       013424  047040  046517  047111
       013432  046101  051440  051120
       013440  040505  020104  040440
       013446  052103  040525  000114
 1316  013454  051105  050122  020103  DH3:    .ASCIZ  /ERRPC   STREG    ACTUAL/
       013462  020040  052123  042522
       013470  020107  020040  040440
       013476  052103  040525  000114
 1317                                          .EVEN
 1318
 1319  013504  001116  001316  001124  DT1:    $ERRPC, STREG, $GDDAT, $BDDAT,0
       013512  001126  000000
 1320  013516  001116  001316  001362  DT2:    $ERRPC,STREG,CHANL,$GDDAT,SPREAD,$BDDAT,0
       013524  001124  001364  001126
       013532  000000
 1321  013534  001116  001316  001126  DT3:    $ERRPC,STREG,$BDDAT,0
       013542  000000
 1322  013544  000000                  DF1:    0
```

```
1324                                     .SBTTL  TTY INPUT ROUTINE
 (1)
 (2)                                     ;;*****************************************************************
 (1)                                     .ENABL  LSB
 (1)  013546  000000                     $TKCNT: .WORD   0                  ;;NUMBER OF ITEMS IN QUEUE
 (1)  013550  000000                     $TKQIN: .WORD   0                  ;;INPUT POINTER
 (1)  013552  000000                     $TKQOUT:  .WORD  0                 ;;OUTPUT POINTER
 (1)  013554  000040                     $TKQSRT: .BLKB  32.                ;;TTY KEYBOARD QUEUE
 (1)          013614                     $TKQEND=.
 (1)
 (1)                                     ;*TK INITIALIZE ROUTINE
 (1)                                     ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
 (1)                                     ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
 (1)                                     ;
 (1)                                     ;*CALL:
 (1)                                     ;*      JSR     PC,$TKINT
 (1)                                     ;*      RETURN
 (1)                                     ;
 (1)  013614  005037  013546             $TKINT: CLR     $TKCNT             ;;CLEAR COUNT OF ITEMS IN QUEUE
 (1)  013620  012737  013554  013550             MOV     #$TKQSRT,$TKQIN    ;;MOVE THE STARTING ADDRESS OF THE
 (1)  013626  013737  013550  013552             MOV     $TKQIN,$TKQOUT     ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
 (1)  013634  012737  013664  000060             MOV     #$TKSRV,@#TKVEC    ;;INITIALIZE THE KEYBOARD VECTOR
 (1)  013642  012737  000200  000062             MOV     #200,@#TKVEC+2     ;;'BR' LEVEL 4
 (1)  013650  005777  165272                     TST     @$TKB              ;;CLEAR DONE FLAG
 (1)  013654  012777  000100  165262             MOV     #100,@$TKS         ;;ENABLE TTY KEYBOARD INTERRUPT
 (1)  013662  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)  013664  117746  165256             $TKSRV: MOVB    @$TKB,-(SP)        ;;PICKUP THE CHARACTER
 (1)  013670  042716  177600                     BIC     #^C177,(SP)        ;;STRIP THE JUNK
 (1)  013674  021627  000003                     CMP     (SP),#3            ;;IS IT A CONTROL C?
 (1)  013700  001007                             BNE     1$                 ;;BRANCH IF NO
 (1)  013702  104401  015030                     TYPE    ,$CNTLC            ;;TYPE A CONTROL-C (^C)
 (1)  013706  004737  013614                     JSR     PC,$TKINT          ;;INIT THE KEYBOARD
 (1)  013712  005726                             TST     (SP)+              ;;CLEAN UP STACK
 (1)  013714  000137  001530                     JMP     BEGIN2             ;;CONTROL C RESTART
 (1)  013720  021627  000007             1$:     CMP     (SP),#7            ;;IS IT A CONTROL G?
 (1)  013724  001004                             BNE     2$                 ;;BRANCH IF NO
 (1)  013726  022737  000176  001140             CMP     #SWREG,SWR         ;;IS SOFT-SWR SELECTED?
 (1)  013734  001500                             BEQ     6$                 ;;GO TO SWR CHANGE
 (1)
 (1)  013736                             2$:
 (1)  013736  022737  000040  013546             CMP     #32.,$TKCNT        ;;IS THE QUEUE FULL?
 (1)  013744  001004                             BNE     3$                 ;;BRANCH IF NO
 (1)  013746  104401  001164                     TYPE    ,$BELL             ;;RING THE TTY BELL
 (1)  013752  005726                             TST     (SP)+              ;;CLEAN CHARACTER OFF OF STACK
 (1)  013754  000451                             BR      5$                 ;;EXIT
 (1)  013756  021627  000023             3$:     CMP     (SP),#23           ;;IS IT A CONTROL-S?
 (1)  013762  001021                             BNE     32$                ;;BRANCH IF NO
 (1)  013764  005077  165154                     CLR     @$TKS              ;;DISABLE TTY KEYBOARD INTERRUPTS
```

```
(1)  013770  005726                          TST     (SP)+                 ;;CLEAN CHAR OFF STACK
(1)  013772  105777  165146          31$:    TSTB    @$TKS                 ;;WAIT FOR A CHAR
(1)  013776  100375                          BPL     31$                   ;;LOOP UNTIL ITS THERE
(1)  014000  117746  165142                  MOVB    @$TKB,-(SP)           ;;GET THE CHARACTER
(1)  014004  042716  177600                  BIC     #^C177,(SP)           ;;MAKE IT 7-BIT ASCII
(1)  014010  022627  000021                  CMP     (SP)+,#21             ;;IS IT A CONTROL-Q?
(1)  014014  001366                          BNE     31$                   ;;BRANCH IF NO
(1)  014016  012777  000100  165120          MOV     #100,@$TKS            ;;REENABLE TTY KEYBOARD INTERRUPTS
(1)  014024  000002                          RTI                           ;;RETURN
(1)  014026  005237  013546          32$:    INC     $TKCNT                ;;COUNT THIS CHARACTER
(1)  014032  021627  000140                  CMP     (SP),#140             ;;IS IT UPPER CASE?
(1)  014036  002405                          BLT     4$                    ;;BRANCH IF YES
(1)  014040  021627  000175                  CMP     (SP),#175             ;;IS IT A SPECIAL CHAR?
(1)  014044  003002                          BGT     4$                    ;;BRANCH IF YES
(1)  014046  042716  000040                  BIC     #40,(SP)              ;;MAKE IT UPPER CASE
(1)  014052  112677  177472          4$:     MOVB    (SP)+,@$TKQIN         ;;AND PUT IT IN QUEUE
(1)  014056  005237  013550                  INC     $TKQIN                ;;UPDATE THE POINTER
(1)  014062  023727  013550  013614          CMP     $TKQIN,#$TKQEND       ;;GO OFF THE END?
(1)  014070  001003                          BNE     5$                    ;;BRANCH IF NO
(1)  014072  012737  013554  013550          MOV     #$TKQSRT,$TKQIN       ;;RESET THE POINTER
(1)  014100  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)  014102  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR            ;;IS THE SOFT-SWR SELECTED
(1)  014110  001124                          BNE     15$                   ;;EXIT IF NOT
(1)  014112  105777  165026                  TSTB    @$TKS                 ;;IS A CHAR WAITING?
(1)  014116  100121                          BPL     15$                   ;;IF NOT, EXIT
(1)  014120  117746  165022                  MOVB    @$TKB,-(SP)           ;;YES
(1)  014124  042716  177600                  BIC     #^C177,(SP)           ;;MAKE IT  7-BIT ASCII
(1)  014130  021627  000007                  CMP     (SP),#7               ;;IS IT A CONTROL-G?
(1)  014134  001300                          BNE     2$                    ;;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)  014136  123727  001134  000001  6$:     CMPB    $AUTOB,#1             ;;ARE WE RUNNING IN AUTO-MODE?
(1)  014144  001674                          BEQ     2$                    ;;BRANCH IF YES
(1)  014146  005726                          TST     (SP)+                 ;;CLEAR CONTROL-G OFF STACK
(1)  014150  004737  013614                  JSR     PC,$TKINT             ;;FLUSH THE TTY INPUT QUEUE
(1)  014154  005077  164764                  CLR     @$TKS                 ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  014160  112737  000001  001135          MOVB    #1,$INTAG             ;;SET INTERRUPT MODE INDICATOR
(1)
(1)  014166  104401  015042                  TYPE    ,$CNTLG               ;;ECHO THE CONTROL-G (^G)
(1)  014172  104401  015047          $GTSWR: TYPE    ,$MSWR                ;;TYPE CURRENT CONTENTS
(2)  014176  013746  000176                  MOV     SWREG,-(SP)           ;;SAVE SWREG FOR TYPEOUT
(2)  014202  104402                          TYPOC                         ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  014204  104401  015060                  TYPE    ,$MNEW                ;;PROMPT FOR NEW SWR
(1)  014210  005046                  19$:    CLR     -(SP)                 ;;CLEAR COUNTER
(1)  014212  005046                          CLR     -(SP)                 ;;THE NEW SWR
(1)  014214  105777  164724          7$:     TSTB    @$TKS                 ;;CHAR THERE?
```

```
   (1)  014220  100375                              BPL     7$                  ;;IF NOT TRY AGAIN
   (1)
   (1)  014222  117746  164720                      MOVB    @$TKB,-(SP)         ;;PICK UP CHAR
   (1)  014226  042716  177600                      BIC     #^C177,(SP)         ;;MAKE IT 7-BIT ASCII
   (1)
   (1)  014232  021627  000003                      CMP     (SP),#3             ;;IS IT A CONTROL-C?
   (1)  014236  001015                              BNE     9$                  ;;BRANCH IF NOT
   (1)  014240  104401  015030                      TYPE    ,$CNTLC             ;;YES, ECHO CONTROL-C (^C)
   (1)  014244  062706  000006                      ADD     #6,SP               ;;CLEAN UP STACK
   (1)  014250  123727  001135  000001              CMPB    $INTAG,#1           ;;REENABLE TTY KEYBOARD INTERRUPTS?
   (1)  014256  001003                              BNE     8$                  ;;BRANCH IF NO
   (1)  014260  012777  000100  164656            . MOV     #100,@$TKS          ;;ALLOW TTY KEYBOARD INTERRUPTS
   (1)  014266  000137  001530          8$:         JMP     BEGIN2              ;;CONTROL-C RESTART
   (1)
   (1)
   (1)  014272  021627  000025          9$:         CMP     (SP),#25            ;;IS IT A CONTROL-U?
   (1)  014276  001005                              BNE     10$                 ;;BRANCH IF NOT
   (1)  014300  104401  015035                      TYPE    ,$CNTLU             ;;YES, ECHO CONTROL-U (^U)
   (1)  014304  062706  000006          20$:        ADD     #6,SP               ;;IGNORE PREVIOUS INPUT
   (1)  014310  000737                              BR      19$                 ;;LET'S TRY IT AGAIN
   (1)
   (1)
   (1)  014312  021627  000015          10$:        CMP     (SP),#15            ;;IS IT A <CR>?
   (1)  014316  001022                              BNE     16$                 ;;BRANCH IF NO
   (1)  014320  005766  000004                      TST     4(SP)               ;;YES, IS IT THE FIRST CHAR?
   (1)  014324  001403                              BEQ     11$                 ;;BRANCH IF YES
   (1)  014326  016677  000002  164604              MOV     2(SP),@SWR          ;;SAVE NEW SWR
   (1)  014334  062706  000006          11$:        ADD     #6,SP               ;;CLEAR UP STACK
   (1)  014340  104401  001171          14$:        TYPE    ,$CRLF              ;;ECHO <CR> AND <LF>
   (1)  014344  123727  001135  000001              CMPB    $INTAG,#1           ;;RE-ENABLE TTY KBD INTERRUPTS?
   (1)  014352  001003                              BNE     15$                 ;;BRANCH IF NOT
   (1)  014354  012777  000100  164562              MOV     #100,@$TKS          ;;RE-ENABLE TTY KBD INTERRUPTS
   (1)  014362  000002                  15$:        RTI                         ;;RETURN
   (1)  014364  004737  016400          16$:        JSR     PC,$TYPEC           ;;ECHO CHAR
   (1)  014370  021627  000060                      CMP     (SP),#60            ;;CHAR < 0?
   (1)  014374  002420                              BLT     18$                 ;;BRANCH IF YES
   (1)  014376  021627  000067                      CMP     (SP),#67            ;;CHAR > 7?
   (1)  014402  003015                              BGT     18$                 ;;BRANCH IF YES
   (1)  014404  042726  000060                      BIC     #60,(SP)+           ;;STRIP-OFF ASCII
   (1)  014410  005766  000002                      TST     2(SP)               ;;IS THIS THE FIRST CHAR
   (1)  014414  001403                              BEQ     17$                 ;;BRANCH IF YES
   (1)  014416  006316                              ASL     (SP)                ;;NO, SHIFT PRESENT
   (1)  014420  006316                              ASL     (SP)                ;;    CHAR OVER TO MAKE
   (1)  014422  006316                              ASL     (SP)                ;;    ROOM FOR NEW ONE.
   (1)  014424  005266  000002          17$:        INC     2(SP)               ;;KEEP COUNT OF CHAR
   (1)  014430  056616  177776                      BIS     -2(SP),(SP)         ;;SET IN NEW CHAR
   (1)  014434  000667                              BR      7$                  ;;GET THE NEXT ONE
   (1)  014436  104401  001170          18$:        TYPE    ,$QUES              ;;TYPE ?<CR><LF>
   (1)  014442  000720                              BR      20$                 ;;SIMULATE CONTROL-U
   (1)                                  .DSABL      LSB
   (1)
   (1)
   (2)                                  ;;*******************************************************************
   (1)                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
   (1)                                  ;*CALL:
   (1)                                  ;*          RDCHR                       ;;GET A CHARACTER FROM THE QUEUE
```

```
(1)                                     ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                     ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                     ;
(1)
(1) 014444  011646                      $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
(1) 014446  016666  000004  000002              MOV     4(SP),2(SP)     ;;THE PS
(1) 014454  005066  000004                      CLR     4(SP)           ;;GET READY FOR A CHARACTER
(2) 014460  005046                              CLR     -(SP)           ;;PUT NEW PS ON STACK
(2) 014462  012746  014470                      MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
(2) 014466  000002                              RTI                     ;;POP NEW PC AND PS
(2) 014470                              64$:
(1) 014470  005737  013546              1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
(1) 014474  001775                              BEQ     1$
(1) 014476  005337  013546                      DEC     $TKCNT          ;;DECREMENT THE COUNTER
(1) 014502  117766  177044  000004              MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
(1) 014510  005237  013552                      INC     $TKQOUT         ;;UPDATE THE POINTER
(1) 014514  023727  013552  013614              CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
(1) 014522  001003                              BNE     2$              ;;BRANCH IF NO
(1) 014524  012737  013554  013552              MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
(1) 014532  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) 014534  010346                      $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1) 014536  005046                              CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1) 014540  012703  014770              1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1) 014544  022703  015030              2$:     CMP     #$TTYIN+32.,R3  ;;BUFFER FULL?
(1) 014550  101456                              BLOS    4$              ;;BR IF YES
(1) 014552  104411                              RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1) 014554  112613                              MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1) 014556  122713  000177              10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1) 014562  001022                              BNE     5$              ;;BR IF NO
(1) 014564  005716                              TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1) 014566  001007                              BNE     6$              ;;BR IF NO
(1) 014570  112737  000134  014766              MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1) 014576  104401  014766                      TYPE    ,9$
(1) 014602  012716  177777                      MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1) 014606  005303                      6$:     DEC     R3              ;;BACKUP BY ONE
(1) 014610  020327  014770                      CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1) 014614  103434                              BLO     4$              ;;BR IF YES
(1) 014616  111337  014766                      MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1) 014622  104401  014766                      TYPE    ,9$             ;;GO TYPE
(1) 014626  000746                              BR      2$              ;;GO READ ANOTHER CHAR.
(1) 014630  005716                      5$:     TST     (SP)            ;;RUBOUT KEY SET?
(1) 014632  001406                              BEQ     7$              ;;BR IF NO
(1) 014634  112737  000134  014766              MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1) 014642  104401  014766                      TYPE    ,9$
(1) 014646  005016                              CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1) 014650  122713  000025              7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1) 014654  001003                              BNE     8$              ;;BR IF NO
(1) 014656  104401  015035                      TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
(1) 014662  000726                              BR      1$              ;;GO START OVER
```

```
    (1)  014664  122713  000022          8$:      CMPB    #22,(R3)          ;;IS CHARACTER A ''^R''?
    (1)  014670  001011                           BNE     3$                ;;BRANCH IF NO
    (1)  014672  105013                           CLRB    (R3)              ;;CLEAR THE CHARACTER
    (1)  014674  104401  001171                   TYPE    ,$CRLF            ;;TYPE A ''CR'' & ''LF''
    (1)  014700  104401  014770                   TYPE    ,$TTYIN           ;;TYPE THE INPUT STRING
    (1)  014704  000717                           BR      2$                ;;GO PICKUP ANOTHER CHACTER
    (1)  014706  104401  001170          4$:      TYPE    ,$QUES            ;;TYPF A '?'
    (1)  014712  000712                           BR      1$                ;;CLEAR THE BUFFER AND LOOP
    (1)  014714  111337  014766          3$:      MOVB    (R3),9$           ;;ECHO THE CHARACTER
    (1)  014720  104401  014766                   TYPE    ,9$
    (1)  014724  122723  000015                   CMPB    #15,(R3)+         ;;CHECK FOR RETURN
    (1)  014730  001305                           BNE     2$                ;;LOOP IF NOT RETURN
    (1)  014732  105063  177777                   CLRB    -1(R3)            ;;CLEAR RETURN (THE 15)
    (1)  014736  104401  001172                   TYPE    ,$LF              ;;TYPE A LINE FEED
    (1)  014742  005726                           TST     (SP)+             ;;CLEAN RUBOUT KEY FROM THE STACK
    (1)  014744  012603                           MOV     (SP)+,R3          ;;RESTORE R3
    (1)  014746  011646                           MOV     (SP),-(SP)        ;;ADJUST THE STACK AND PUT ADDRESS OF THE
    (1)  014750  016666  000004  000002           MOV     4(SP),2(SP)       ;;        FIRST ASCII CHARACTER ON IT
    (1)  014756  012766  014770  000004           MOV     #$TTYIN,4(SP)
    (1)  014764  000002                           RTI                       ;;RETURN
    (1)  014766     000                  9$:      .BYTE   0                 ;;STORAGE FOR ASCII CHAR. TO TYPE
    (1)  014767     000                           .BYTE   0                 ;;TERMINATOR
    (1)  014770  000040                  $TTYIN:  .BLKB   32.               ;;RESERVE 32. BYTES FOR TTY INPUT
    (1)  015030  041536  005015     000  $CNTLC:  .ASCIZ  /^C/<15><12>      ;;CONTROL ''C''
    (1)  015035     136  006525  000012  $CNTLU:  .ASCIZ  /^U/<15><12>      ;;CONTROL ''U''
    (1)  015042  043536  005015     000  $CNTLG:  .ASCIZ  /^G/<15><12>      ;;CONTROL ''G''
    (1)  015047     015  051412  051127  $MSWR:   .ASCIZ  <15><12>/SWR = /
    (1)  015054  036440  000040
    (1)  015060  020040  042516  020127  $MNEW:   .ASCIZ  /  NEW = /
    (1)  015066  020075     000
    (1)          015072                           .EVEN
```

```
 1326                                    .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) 015072 011646                      $RDOCT: MOV     (SP),-(SP)               ;;PROVIDE SPACE FOR THE
  (1) 015074 016666 000004 000002               MOV     4(SP),2(SP)              ;;INPUT NUMBER
  (3) 015102 010046                              MOV     R0,-(SP)                 ;;PUSH R0 ON STACK
  (3) 015104 010146                              MOV     R1,-(SP)                 ;;PUSH R1 ON STACK
  (3) 015106 010246                              MOV     R2,-(SP)                 ;;PUSH R2 ON STACK
  (1) 015110 104412                      1$:     RDLIN                            ;;READ AN ASCIZ LINE
  (1) 015112 012600                              MOV     (SP)+,R0                 ;;GET ADDRESS OF 1ST CHARACTER
  (1) 015114 005001                              CLR     R1                       ;;CLEAR DATA WORD
  (1) 015116 005002                              CLR     R2
  (1) 015120 112046                      2$:     MOVB    (R0)+,-(SP)              ;;PICKUP THIS CHARACTER
  (1) 015122 001412                              BEQ     3$                       ;;IF ZERO GET OUT
  (1) 015124 006301                              ASL     R1                       ;;*2
  (1) 015126 006102                              ROL     R2
  (1) 015130 006301                              ASL     R1                       ;;*4
  (1) 015132 006102                              ROL     R2
  (1) 015134 006301                              ASL     R1                       ;;*8
  (1) 015136 006102                              ROL     R2
  (1) 015140 042716 177770                       BIC     #^C7,(SP)                ;;STRIP THE ASCII JUNK
  (1) 015144 062601                              ADD     (SP)+,R1                 ;;ADD IN THIS DIGIT
  (1) 015146 000764                              BR      2$                       ;;LOOP
  (1) 015150 005726                      3$:     TST     (SP)+                    ;;CLEAN TERMINATOR FROM STACK
  (1) 015152 010166 000012                       MOV     R1,12(SP)                ;;SAVE THE RESULT
  (1) 015156 010237 015172                       MOV     R2,$HIOCT
  (3) 015162 012602                              MOV     (SP)+,R2                 ;;POP STACK INTO R2
  (3) 015164 012601                              MOV     (SP)+,R1                 ;;POP STACK INTO R1
  (3) 015166 012600                              MOV     (SP)+,R0                 ;;POP STACK INTO R0
  (1) 015170 000002                              RTI                              ;;RETURN
  (1) 015172 000000                      $HIOCT: .WORD   0                        ;;HIGH ORDER BITS GO HERE
 1328                                   .SBTTL  POWER DOWN AND UP ROUTINES
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   ;POWER DOWN ROUTINE
  (1)  015174  012737  015334  000024   $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
  (1)  015202  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (3)  015210  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (3)  015212  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (3)  015214  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (3)  015216  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (3)  015220  010446                           MOV     R4,-(SP)        ;;PUSH R4 ON STACK
  (3)  015222  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
  (3)  015224  017746  163710                   MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
  (1)  015230  010637  015340                   MOV     SP,$SAVR6       ;;SAVE SP
  (1)  015234  012737  015246  000024           MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
  (1)  015242  000000                           HALT
  (1)  015244  000776                           BR      .-2             ;;HANG UP
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   ;POWER UP ROUTINE
  (1)  015246  012737  015334  000024   $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
  (1)  015254  013706  015340                   MOV     $SAVR6,SP       ;;GET SP
  (1)  015260  005037  015340                   CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
  (1)  015264  005237  015340           1$:     INC     $SAVR6          ;;WAIT FOR THE INC
  (1)  015270  001375                           BNE     1$              ;;OF  WORD
  (3)  015272  012677  163642                   MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
  (3)  015276  012605                           MOV     (SP)+,R5        ;;POP STACK INTO R5
  (3)  015300  012604                           MOV     (SP)+,R4        ;;POP STACK INTO R4
  (3)  015302  012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
  (3)  015304  012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
  (3)  015306  012601                           MOV     (SP)+,R1        ;;POP STACK INTO R1
  (3)  015310  012600                           MOV     (SP)+,R0        ;;POP STACK INTO R0
  (1)  015312  012737  015174  000024           MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
  (1)  015320  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (1)  015326  104401                           TYPE                    ;REPORT THE POWER FAILURE
  (1)  015330  015342                   $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
  (1)  015332  000002                           RTI
  (1)  015334  000000                   $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
  (1)  015336  000776                           BR      .-2             ;;  BEFORE THE POWER DOWN WAS COMPLETE
  (1)  015340  000000                   $SAVR6: 0                       ;;PUT THE SP HERE
  (1)  015342  005015  047520  042527   $POWER: .ASCIZ  <15><12>'POWER'
  (1)  015350  000122
  (1)                                           .EVEN
```

```
                                          .SBTTL  SCOPE HANDLER ROUTINE
 (1)
 (2)                                      ;;*****************************************************************
 (1)                                      ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 (1)                                      ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 (1)                                      ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
 (1)                                      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 (1)                                      ;*SW14=1          LOOP ON TEST
 (1)                                      ;*SW11=1          INHIBIT ITERATIONS
 (1)                                      ;*SW09=1          LOOP ON ERROR
 (1)                                      ;*SW08=1          LOOP ON TEST IN SWR<7:0>
 (1)                                      ;*CALL
 (1)                                      ;*        SCOPE             ;;SCOPE=IOT
 (1)
 (1)  015352                              $SCOPE:
 (1)  015352  104410                              CKSWR                     ;;TEST  FOR CHANGE IN SOFT-SWR
 (1)  015354  032777  040000  163556      1$:     BIT     #BIT14,@SWR       ;;LOOP ON PRESENT TEST?
 (1)  015362  001114                              BNE     $OVER             ;;YES IF SW14=1
 (1)                                      ;#####START OF CODE FOR THE XOR TESTER#####
 (1)  015364  000416                      $XTSTR: BR      6$                ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
 (1)                                                                        ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
 (1)  015366  013746  000004                      MOV     @#ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 (1)  015372  012737  015412  000004              MOV     #5$,@#ERRVEC      ;;SET FOR TIMEOUT
 (1)  015400  005737  177060                      TST     @#177060          ;;TIME OUT ON XOR?
 (1)  015404  012637  000004                      MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
 (1)  015410  000463                              BR      $SVLAD            ;;GO TO THE NEXT TEST
 (1)  015412  022626                      5$:     CMP     (SP)+,(SP)+       ;;CLEAR THE STACK AFTER A TIME OUT
 (1)  015414  012637  000004                      MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
 (1)  015420  000423                              BR      7$                ;;LOOP ON THE PRESENT TEST
 (1)  015422                              6$:;#####END OF CODE FOR THE XOR TESTER#####
 (1)  015422  032777  000400  163510              BIT     #BIT08,@SWR       ;;LOOP ON SPEC. TEST?
 (1)  015430  001404                              BEQ     2$                ;;BR IF NO
 (1)  015432  127737  163502  001102              CMPB    @SWR,$TSTNM       ;;ON THE RIGHT TEST?    SWR<7:0>
 (1)  015440  001465                              BEQ     $OVER             ;;BR IF YES
 (1)  015442  105737  001103              2$:     TSTB    $ERFLG            ;;HAS AN ERROR OCCURRED?
 (1)  015446  001421                              BEQ     3$                ;;BR IF NO
 (1)  015450  123737  001115  001103              CMPB    $ERMAX,$ERFLG     ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 (1)  015456  101015                              BHI     3$                ;;BR IF NO
 (1)  015460  032777  001000  163452              BIT     #BIT09,@SWR       ;;LOOP ON ERROR?
 (1)  015466  001404                              BEQ     4$                ;;BR IF NO
 (1)  015470  013737  001110  001106      7$:     MOV     $LPERR,$LPADR     ;;SET LOOP ADDRESS TO LAST SCOPE
 (1)  015476  000446                              BR      $OVER
 (1)  015500  105037  001103              4$:     CLRB    $ERFLG            ;;ZERO THE ERROR FLAG
 (1)  015504  005037  001160                      CLR     $TIMES            ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 (1)  015510  000415                              BR      1$                ;;ESCAPE TO THE NEXT TEST
 (1)  015512  032777  004000  163420      3$:     BIT     #BIT11,@SWR       ;;INHIBIT ITERATIONS?
 (1)  015520  001011                              BNE     1$                ;;BR IF YES
 (1)  015522  005737  001202                      TST     $PASS             ;;IF FIRST PASS OF PROGRAM
 (1)  015526  001406                              BEQ     1$                ;;        INHIBIT ITERATIONS
 (1)  015530  005237  001104                      INC     $ICNT             ;;INCREMENT ITERATION COUNT
 (1)  015534  023737  001160  001104              CMP     $TIMES,$ICNT      ;;CHECK THE NUMBER OF ITERATIONS MADE
 (1)  015542  002024                              BGE     $OVER             ;;BR IF MORE ITERATION REQUIRED
 (1)  015544  012737  000001  001104      1$:     MOV     #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
 (1)  015552  013737  015630  001160              MOV     $MXCNT,$TIMES     ;;SET NUMBER OF ITERATIONS TO DO
 (1)  015560  105237  001102              $SVLAD: INCB    $TSTNM            ;;COUNT TEST NUMBERS
 (1)  015564  113737  001102  001200              MOVB    $TSTNM,$TESTN     ;;SET TEST NUMBER IN APT MAILBOX
```

```
  (1)  015572  011637  001106                   MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
  (1)  015576  011637  001110                   MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
  (1)  015602  005037  001162                   CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
  (1)  015606  112737  000001  001115           MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
  (1)  015614  013777  001102  163320  $OVER:   MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
  (1)  015622  013716  001106                   MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
  (1)  015626  000002                           RTI                     ;;FIXES PS
  (1)  015630  003720                  $MXCNT:  2000.                   ;;MAX. NUMBER OF ITERATIONS
 1331                                  .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)  015632                          $ERROR:
  (1)  015632  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
  (1)  015634  105237  001103          7$:      INCB    $ERFLG          ;;SET THE ERROR FLAG
  (1)  015640  001775                           BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
  (1)  015642  013777  001102  163272           MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
  (1)  015650  032777  002000  163262           BIT     #BIT10,@SWR     ;;BELL ON ERROR?
  (1)  015656  001402                           BEQ     1$              ;;NO - SKIP
  (1)  015660  104401  001164                   TYPE    ,$BELL          ;;RING BELL
  (1)  015664  005237  001112          1$:      INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
  (1)  015670  011637  001116                   MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
  (1)  015674  162737  000002  001116           SUB     #2,$ERRPC
  (1)  015702  117737  163210  001114           MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
  (1)  015710  032777  020000  163222           BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
  (1)  015716  001004                           BNE     20$             ;;SKIP TYPEOUTS
  (1)  015720  004737  016032                   JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
  (1)  015724  104401  001171                   TYPE    ,$CRLF
  (1)  015730                          20$:
  (1)  015730  122737  000001  001214           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)  015736  001007                           BNE     2$              ;;NO,SKIP APT ERROR REPORT
  (1)  015740  113737  001114  015752           MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
  (1)  015746  004737  016540                   JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
  (1)  015752     000                  21$:     .BYTE   0
  (1)  015753     000                           .BYTE   0
  (1)  015754  000777                  22$:     BR      22$             ;;APT ERROR LOOP
  (1)  015756  005777  163156          2$:      TST     @SWR            ;;HALT ON ERROR
  (1)  015762  100002                           BPL     3$              ;;SKIP IF CONTINUE
  (1)  015764  000000                           HALT                    ;;HALT ON ERROR!
  (1)  015766  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
  (1)  015770  032777  001000  163142  3$:      BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
  (1)  015776  001402                           BEQ     4$              ;;BR IF NO
  (1)  016000  013716  001110                   MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
  (1)  016004  005737  001162          4$:      TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
  (1)  016010  001402                           BEQ     5$              ;;BR IF NONE
  (1)  016012  013716  001162                   MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
```

```
    (1)  016016                                     5$:
    (1)  016016  022737  010342  000042                     CMP     #$ENDAD,@#42        ;;ACT-11 AUTO-ACCEPT?
    (1)  016024  001001                                     BNE     6$                  ;;BRANCH IF NO
    (1)  016026  000000                                     HALT                        ;;YES
    (1)  016030                                     6$:
    (1)  016030  000002                                     RTI                         ;;RETURN
   1332                                             .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
    (1)
    (2)                                             ;;*****************************************************************
    (1)                                             ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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)  016032                                     $ERRTYP:
    (1)  016032  104401  001171                             TYPE    ,$CRLF              ;;"CARRIAGE RETURN" & "LINE FEED"
    (1)  016036  010046                                     MOV     R0,-(SP)            ;;SAVE R0
    (1)  016040  005000                                     CLR     R0                  ;;PICKUP THE ITEM INDEX
    (1)  016042  153700  001114                             BISB    @#$ITEMB,R0
    (1)  016046  001004                                     BNE     1$                  ;;IF ITEM NUMBER IS ZERO, JUST
    (1)                                                                                 ;;TYPE THE PC OF THE ERROR
    (2)  016050  013746  001116                             MOV     $ERRPC,-(SP)        ;;SAVE $ERRPC FOR TYPEOUT
    (2)                                                                                 ;;ERROR ADDRESS
    (2)  016054  104402                                     TYPOC                       ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
    (1)  016056  000426                                     BR      6$                  ;;GET OUT
    (1)  016060  005300                             1$:     DEC     R0                  ;;ADJUST THE INDEX SO THAT IT WILL
    (1)  016062  006300                                     ASL     R0                  ;;       WORK FOR THE ERROR TABLE
    (1)  016064  006300                                     ASL     R0
    (1)  016066  006300                                     ASL     R0
    (1)  016070  062700  001256                             ADD     #$ERRTB,R0          ;;FORM TABLE POINTER
    (1)  016074  012037  016104                             MOV     (R0)+,2$            ;;PICKUP "ERROR MESSAGE" POINTER
    (1)  016100  001404                                     BEQ     3$                  ;;SKIP TYPEOUT IF NO POINTER
    (1)  016102  104401                                     TYPE                        ;;TYPE THE "ERROR MESSAGE"
    (1)  016104  000000                             2$:     .WORD   0                   ;;"ERROR MESSAGE" POINTER GOES HERE
    (1)  016106  104401  001171                             TYPE    ,$CRLF              ;;"CARRIAGE RETURN" & "LINE FEED"
    (1)  016112  012037  016122                     3$:     MOV     (R0)+,4$            ;;PICKUP "DATA HEADER" POINTER
    (1)  016116  001404                                     BEQ     5$                  ;;SKIP TYPEOUT IF 0
    (1)  016120  104401                                     TYPE                        ;;TYPE THE "DATA HEADER"
    (1)  016122  000000                             4$:     .WORD   0                   ;;"DATA HEADER" POINTER GOES HERE
    (1)  016124  104401  001171                             TYPE    ,$CRLF              ;;"CARRIAGE RETURN" & "LINE FEED"
    (1)  016130  011000                             5$:     MOV     (R0),R0             ;;PICKUP "DATA TABLE" POINTER
    (1)  016132  001004                                     BNE     7$                  ;;GO TYPE THE DATA
    (1)  016134  012600                             6$:     MOV     (SP)+,R0            ;;RESTORE R0
    (1)  016136  104401  001171                             TYPE    ,$CRLF              ;;"CARRIAGE RETURN" & "LINE FEED"
    (1)  016142  000207                                     RTS     PC                  ;;RETURN
    (1)  016144                                     7$:
    (2)  016144  013046                                     MOV     @(R0)+,-(SP)        ;;SAVE @(R0)+ FOR TYPEOUT
    (2)  016146  104402                                     TYPOC                       ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
    (1)  016150  005710                                     TST     (R0)                ;;IS THERE ANOTHER NUMBER?
    (1)  016152  001770                                     BEQ     6$                  ;;BR IF NO
    (1)  016154  104401  016162                             TYPE    ,8$                 ;;TYPE TWO(2) SPACES
    (1)  016160  000771                                     BR      7$                  ;;LOOP
    (1)  016162  020040     000                     8$:     .ASCIZ  /  /                ;;TWO(2) SPACES
    (1)          016166                                     .EVEN
```

```
 1334                                          .SBTTL  TYPE ROUTINE
  (1)
  (2)                                          ;;*******************************************************************
  (1)                                          ;;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
  (1)                                          ;;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
  (1)                                          ;;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
  (1)                                          ;;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
  (1)                                          ;;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
  (1)                                          ;;*
  (1)                                          ;;*CALL:
  (1)                                          ;;*1) USING A TRAP INSTRUCTION
  (1)                                          ;;*        TYPE     ,MESADR            ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
  (1)                                          ;;*OR
  (1)                                          ;;*        TYPE
  (1)                                          ;;*        MESADR
  (1)                                          ;;*
  (1)
  (1)  016166  105737  001157          $TYPE:  TSTB     $TPFLG             ;;IS THERE A TERMINAL?
  (1)  016172  100002                          BPL      1$                 ;;BR IF YES
  (1)  016174  000000                          HALT                        ;;HALT HERE IF NO TERMINAL
  (1)  016176  000430                          BR       3$                 ;;LEAVE
  (1)  016200  010046                  1$:     MOV      R0,-(SP)           ;;SAVE R0
  (1)  016202  017600  000002                  MOV      @2(SP),R0          ;;GET ADDRESS OF ASCIZ STRING
  (1)  016206  122737  000001  001214          CMPB     #APTENV,$ENV       ;;RUNNING IN APT MODE
  (1)  016214  001011                          BNE      62$                ;;NO,GO CHECK FOR APT CONSOLE
  (1)  016216  132737  000100  001215          BITB     #APTSPOOL,$ENVM    ;;SPOOL MESSAGE TO APT
  (1)  016224  001405                          BEQ      62$                ;;NO,GO CHECK FOR CONSOLE
  (1)  016226  010037  016236                  MOV      R0,61$             ;;SETUP MESSAGE ADDRESS FOR APT
  (1)  016232  004737  016530                  JSR      PC,$ATY3           ;;SPOOL MESSAGE TO APT
  (1)  016236  000000                  61$:    .WORD    0                  ;;MESSAGE ADDRESS
  (1)  016240  132737  000040  001215  62$:    BITB     #APTCSUP,$ENVM     ;;APT CONSOLE SUPPRESSED
  (1)  016246  001003                          BNE      60$                ;;YES,SKIP TYPE OUT
  (1)  016250  112046                  2$:     MOVB     (R0)+,-(SP)        ;;PUSH CHARACTER TO BE TYPED ONTO STACK
  (1)  016252  001005                          BNE      4$                 ;;BR IF IT ISN'T THE TERMINATOR
  (1)  016254  005726                          TST      (SP)+              ;;IF TERMINATOR POP IT OFF THE STACK
  (1)  016256  012600                  60$:    MOV      (SP)+,R0           ;;RESTORE R0
  (1)  016260  062716  000002          3$:     ADD      #2,(SP)            ;;ADJUST RETURN PC
  (1)  016264  000002                          RTI                         ;;RETURN
  (1)  016266  122716  000011          4$:     CMPB     #HT,(SP)           ;;BRANCH IF <HT>
  (1)  016272  001430                          BEQ      8$
  (1)  016274  122716  000200                  CMPB     #CRLF,(SP)         ;;BRANCH IF NOT <CRLF>
  (1)  016300  001006                          BNE      5$
  (1)  016302  005726                          TST      (SP)+              ;;POP  <CR><LF> EQUIV
  (1)  016304  104401                          TYPE                        ;;TYPE A CR AND LF
  (1)  016306  001171                          $CRLF
  (1)  016310  105037  016516                  CLRB     $CHARCNT           ;;CLEAR CHARACTER COUNT
  (1)  016314  000755                          BR       2$                 ;;GET NEXT CHARACTER
  (1)  016316  004737  016400          5$:     JSR      PC,$TYPEC          ;;GO TYPE THIS CHARACTER
  (1)  016322  123726  001156          6$:     CMPB     $FILLC,(SP)+       ;;IS IT TIME FOR FILLER CHARS.?
  (1)  016326  001350                          BNE      2$                 ;;IF NO GO GET NEXT CHAR.
  (1)  016330  013746  001154                  MOV      $NULL,-(SP)        ;;GET # OF FILLER CHARS. NEEDED
  (1)                                                                      ;;AND THE NULL CHAR.
  (1)  016334  105366  000001          7$:     DECB     1(SP)              ;;DOES A NULL NEED TO BE TYPED?
  (1)  016340  002770                          BLT      6$                 ;;BR IF NO--GO POP THE NULL OFF OF STACK
  (1)  016342  004737  016400                  JSR      PC,$TYPEC          ;;GO TYPE A NULL
  (1)  016346  105337  016516                  DECB     $CHARCNT           ;;DO NOT COUNT AS A COUNT
 (1)  016352  000770                                 BR      7$                ;;LOOP
 (1)
 (1)                                        ;HORIZONTAL TAB PROCESSOR
 (1)
 (1)  016354  112716  000040                8$:      MOVB    #' ,(SP)          ;;REPLACE TAB WITH SPACE
 (1)  016360  004737  016400                9$:      JSR     PC,$TYPEC         ;;TYPE A SPACE
 (1)  016364  132737  000007  016516                 BITB    #7,$CHARCNT       ;;BRANCH IF NOT AT
 (1)  016372  001372                                 BNE     9$                ;;TAB STOP
 (1)  016374  005726                                 TST     (SP)+             ;;POP SPACE OFF STACK
 (1)  016376  000724                                 BR      2$                ;;GET NEXT CHARACTER
 (1)  016400                                $TYPEC:
 (1)  016400  105777  162540                         TSTB    @$TKS             ;;CHAR IN KYBD BUFFER?               ;MJD001
 (1)  016404  100022                                 BPL     10$               ;;BR IF NOT                          ;MJD001
 (1)  016406  017746  162534                         MOV     @$TKB,-(SP)       ;;GET CHAR                           ;MJD001
 (1)  016412  042716  177600                         BIC     #177600,(SP)      ;;STRIP EXTRANEOUS BITS              ;MJD001
 (1)  016416  122716  000023                         CMPB    #$XOFF,(SP)       ;;WAS CHAR XOFF                      ;MJD001
 (1)  016422  001012                                 BNE     102$              ;;BR IF NOT                          ;MJD001
 (1)  016424                                101$:                                                                   ;MJD001
 (1)  016424  105777  162514                         TSTB    @$TKS             ;;WAIT FOR CHAR                      ;MJD001
 (1)  016430  100375                                 BPL     101$                                                   ;MJD001
 (1)  016432  117716  162510                         MOVB    @$TKB,(SP)        ;;GET CHAR                           ;MJD001
 (1)  016436  042716  177600                         BIC     #177600,(SP)      ;;STRIP IT                           ;MJD001
 (1)  016442  122716  000021                         CMPB    #$XON,(SP)        ;;WAS IT XON?                        ;MJD001
 (1)  016446  001366                                 BNE     101$              ;;BR IF NOT                          ;MJD001
 (1)  016450                                102$:                                                                   ;MJD001
 (1)  016450  005726                                 TST     (SP)+             ;;FIX STACK                          ;MJD001
 (1)  016452                                10$:                                                                    ;MJD001
 (1)  016452  105777  162472                         TSTB    @$TPS             ;;WAIT UNTIL PRINTER IS READY
 (1)  016456  100375                                 BPL     10$                                                    ;MJD001
 (1)  016460  116677  000002  162464                 MOVB    2(SP),@$TPB       ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 (1)  016466  122766  000015  000002                 CMPB    #CR,2(SP)         ;;IS CHARACTER A CARRIAGE RETURN?
 (1)  016474  001003                                 BNE     1$                ;;BRANCH IF NO
 (1)  016476  105037  016516                         CLRB    $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
 (1)  016502  000406                                 BR      $TYPEX            ;;EXIT
 (1)  016504  122766  000012  000002        1$:      CMPB    #LF,2(SP)         ;;IS CHARACTER A LINE FEED?
 (1)  016512  001402                                 BEQ     $TYPEX            ;;BRANCH IF YES
 (1)  016514  105227                                 INCB    (PC)+             ;;COUNT THE CHARACTER
 (1)  016516  000000                        $CHARCNT:.WORD   0                 ;;CHARACTER COUNT STORAGE
 (1)  016520  000207                        $TYPEX: RTS      PC
 (1)
1335                                        .SBTTL  APT COMMUNICATIONS ROUTINE
 (1)
 (2)                                        ;;****************************************************************
 (1)  016522  112737  000001  016766        $ATY1:   MOVB    #1,$FFLG          ;;TO REPORT FATAL ERROR
 (1)  016530  112737  000001  016764        $ATY3:   MOVB    #1,$MFLG          ;;TO TYPE A MESSAGE
 (1)  016536  000403                                 BR      $ATYC
 (1)  016540  112737  000001  016766        $ATY4:   MOVB    #1,$FFLG          ;;TO ONLY REPORT FATAL ERROR
 (1)  016546                                $ATYC:
 (3)  016546  010046                                 MOV     R0,-(SP)          ;;PUSH R0 ON STACK
 (3)  016550  010146                                 MOV     R1,-(SP)          ;;PUSH R1 ON STACK
 (1)  016552  105737  016764                         TSTB    $MFLG             ;;SHOULD TYPE A MESSAGE?
 (1)  016556  001450                                 BEQ     5$                ;;IF NOT:  BR
 (1)  016560  122737  000001  001214                 CMPB    #APTENV,$ENV      ;;OPERATING UNDER APT?
 (1)  016566  001031                                 BNE     3$                ;;IF NOT:  BR
 (1)  016570  132737  000100  001215                 BITB    #APTSPOOL,$ENVM   ;;SHOULD SPOOL MESSAGES?
 (1)  016576  001425                                 BEQ     3$                ;;IF NOT:  BR
```

```
    (1)  016600  017600  000004                      MOV     @4(SP),R0           ;;GET MESSAGE ADDR.
    (1)  016604  062766  000002  000004              ADD     #2,4(SP)                      ;;BUMP RETURN ADDR.
    (1)  016612  005737  001174          1$:         TST     $MSGTYPE            ;;SEE IF DONE W/ LAST XMISSION?
    (1)  016616  001375                              BNE     1$                  ;;IF NOT:  WAIT
    (1)  016620  010037  001210                      MOV     R0,$MSGAD           ;;PUT ADDR IN MAILBOX
    (1)  016624  105720                  2$:         TSTB    (R0)+               ;;FIND END OF MESSAGE
    (1)  016626  001376                              BNE     2$
    (1)  016630  163700  001210                      SUB     $MSGAD,R0           ;;SUB START OF MESSAGE
    (1)  016634  006200                              ASR     R0                  ;;GET MESSAGE LNGTH IN WORDS
    (1)  016636  010037  001212                      MOV     R0,$MSGLGT          ;;PUT LENGTH IN MAILBOX
    (1)  016642  012737  000004  001174              MOV     #4,$MSGTYPE         ;;TELL APT TO TAKE MSG.
    (1)  016650  000413                              BR      5$
    (1)  016652  017637  000004  016676  3$:         MOV     @4(SP),4$           ;;PUT MSG ADDR IN JSR LINKAGE
    (1)  016660  062766  000002  000004              ADD     #2,4(SP)                      ;;BUMP RETURN ADDRESS
    (3)  016666  013746  177776                      MOV     177776,-(SP)       ;;PUSH 177776 ON STACK
    (1)  016672  004737  016166                      JSR     PC,$TYPE            ;;CALL TYPE MACRO
    (1)  016676  000000                  4$:         .WORD   0
    (1)  016700                          5$:
    (1)  016700  105737  016766          10$:        TSTB    $FFLG               ;;SHOULD REPORT FATAL ERROR?
    (1)  016704  001416                              BEQ     12$                 ;;IF NOT:  BR
    (1)  016706  005737  001214                      TST     $ENV                ;;RUNNING UNDER APT?
    (1)  016712  001413                              BEQ     12$                 ;;IF NOT:  BR
    (1)  016714  005737  001174          11$:        TST     $MSGTYPE            ;;FINISHED LAST MESSAGE?
    (1)  016720  001375                              BNE     11$                 ;;IF NOT:  WAIT
    (1)  016722  017637  000004  001176              MOV     @4(SP),$FATAL       ;;GET ERROR #
    (1)  016730  062766  000002  000004              ADD     #2,4(SP)                      ;;BUMP RETURN ADDR.
    (1)  016736  005237  001174                      INC     $MSGTYPE            ;;TELL APT TO TAKE ERROR
    (1)  016742  105037  016766          12$:        CLRB    $FFLG               ;;CLEAR FATAL FLAG
    (1)  016746  105037  016765                      CLRB    $LFLG               ;;CLEAR LOG FLAG
    (1)  016752  105037  016764                      CLRB    $MFLG               ;;CLEAR MESSAGE FLAG
    (3)  016756  012601                              MOV     (SP)+,R1            ;;POP STACK INTO R1
    (3)  016760  012600                              MOV     (SP)+,R0            ;;POP STACK INTO R0
    (1)  016762  000207                              RTS     PC                  ;;RETURN
    (1)  016764     000                  $MFLG:      .BYTE   0                   ;;MESSG. FLAG
    (1)  016765     000                  $LFLG:      .BYTE   0                   ;;LOG FLAG
    (1)  016766     000                  $FFLG:      .BYTE   0                   ;;FATAL FLAG
    (1)          016770                              .EVEN
    (1)          000200                  APTSIZE=200
    (1)          000001                  APTENV=001
    (1)          000100                  APTSPOOL=100
    (1)          000040                  APTCSUP=040
```

```
 1337                                        .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)  016770  017646  000000                $TYPOS: MOV     @(SP),-(SP)         ;;PICKUP THE MODE
  (1)  016774  116637  000001  017213                MOVB    1(SP),$OFILL        ;;LOAD ZERO FILL SWITCH
  (1)  017002  112637  017215                        MOVB    (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
  (1)  017006  062716  000002                        ADD     #2,(SP)             ;;ADJUST RETURN ADDRESS
  (1)  017012  000406                                BR      $TYPON
  (1)  017014  112737  000001  017213        $TYPOC: MOVB    #1,$OFILL           ;;SET THE ZERO FILL SWITCH
  (1)  017022  112737  000006  017215                MOVB    #6,$OMODE+1         ;;SET FOR SIX(6) DIGITS
  (1)  017030  112737  000005  017212        $TYPON: MOVB    #5,$OCNT            ;;SET THE ITERATION COUNT
  (1)  017036  010346                                MOV     R3,-(SP)            ;;SAVE R3
  (1)  017040  010446                                MOV     R4,-(SP)            ;;SAVE R4
  (1)  017042  010546                                MOV     R5,-(SP)            ;;SAVE R5
  (1)  017044  113704  017215                        MOVB    $OMODE+1,R4         ;;GET THE NUMBER OF DIGITS TO TYPE
  (1)  017050  005404                                NEG     R4
  (1)  017052  062704  000006                        ADD     #6,R4               ;;SUBTRACT IT FOR MAX. ALLOWED
  (1)  017056  110437  017214                        MOVB    R4,$OMODE           ;;SAVE IT FOR USE
  (1)  017062  113704  017213                        MOVB    $OFILL,R4           ;;GET THE ZERO FILL SWITCH
  (1)  017066  016605  000012                        MOV     12(SP),R5           ;;PICKUP THE INPUT NUMBER
  (1)  017072  005003                                CLR     R3                  ;;CLEAR THE OUTPUT WORD
  (1)  017074  006105                        1$:     ROL     R5                  ;;ROTATE MSB INTO "C"
  (1)  017076  000404                                BR      3$                  ;;GO DO MSB
  (1)  017100  006105                        2$:     ROL     R5                  ;;FORM THIS DIGIT
  (1)  017102  006105                                ROL     R5
  (1)  017104  006105                                ROL     R5
  (1)  017106  010503                                MOV     R5,R3
  (1)  017110  006103                        3$:     ROL     R3                  ;;GET LSB OF THIS DIGIT
  (1)  017112  105337  017214                        DECB    $OMODE              ;;TYPE THIS DIGIT?
  (1)  017116  100016                                BPL     7$                  ;;BR IF NO
  (1)  017120  042703  177770                        BIC     #177770,R3          ;;GET RID OF JUNK
  (1)  017124  001002                                BNE     4$                  ;;TEST FOR 0
  (1)  017126  005704                                TST     R4                  ;;SUPPRESS THIS 0?
  (1)  017130  001403                                BEQ     5$                  ;;BR IF YES
```

```
   (1)  017132  005204                    4$:      INC     R4                      ;;DON'T SUPPRESS ANYMORE 0'S
   (1)  017134  052703  000060                     BIS     #'0,R3                  ;;MAKE THIS DIGIT ASCII
   (1)  017140  052703  000040            5$:      BIS     #' ,R3                  ;;MAKE ASCII IF NOT ALREADY
   (1)  017144  110337  017210                     MOVB    R3,8$                   ;;SAVE FOR TYPING
   (1)  017150  104401  017210                     TYPE    ,8$                     ;;GO TYPE THIS DIGIT
   (1)  017154  105337  017212            7$:      DECB    $OCNT                   ;;COUNT BY 1
   (1)  017160  003347                             BGT     2$                      ;;BR IF MORE TO DO
   (1)  017162  002402                             BLT     6$                      ;;BR IF DONE
   (1)  017164  005204                             INC     R4                      ;;INSURE LAST DIGIT ISN'T A BLANK
   (1)  017166  000744                             BR      2$                      ;;GO DO THE LAST DIGIT
   (1)  017170  012605                    6$:      MOV     (SP)+,R5                ;;RESTORE R5
   (1)  017172  012604                             MOV     (SP)+,R4                ;;RESTORE R4
   (1)  017174  012603                             MOV     (SP)+,R3                ;;RESTORE R3
   (1)  017176  016666  000002  000004             MOV     2(SP),4(SP)             ;;SET THE STACK FOR RETURNING
   (1)  017204  012616                             MOV     (SP)+,(SP)
   (1)  017206  000002                             RTI                             ;;RETURN
   (1)  017210     000                    8$:      .BYTE   0                       ;;STORAGE FOR ASCII DIGIT
   (1)  017211     000                             .BYTE   0                       ;;TERMINATOR FOR TYPE ROUTINE
   (1)  017212     000                    $OCNT:   .BYTE   0                       ;;OCTAL DIGIT COUNTER
   (1)  017213     000                    $OFILL:  .BYTE   0                       ;;ZERO FILL SWITCH
   (1)  017214  000000                    $OMODE:  .WORD   0                       ;;NUMBER OF DIGITS TO TYPE
 1338                                     .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)  017216  010146                    $TYPBN:  MOV     R1,-(SP)                ;;SAVE R1 ON THE STACK
   (1)  017220  016601  000006                     MOV     6(SP),R1                ;;GET THE INPUT NUMBER
   (1)  017224  000261                             SEC                             ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
   (1)  017226  112737  000060  017270    1$:      MOVB    #'0,$BIN                ;;SET CHARACTER TO AN ASCII "0".
   (1)  017234  006101                             ROL     R1                      ;;GET THIS BIT
   (1)  017236  001406                             BEQ     2$                      ;;DONE?
   (1)  017240  105537  017270                     ADCB    $BIN                    ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
   (1)  017244  104401  017270                     TYPE    ,$BIN                   ;;GO TYPE THIS BIT
   (1)  017250  000241                             CLC                             ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
   (1)  017252  000765                             BR      1$                      ;;GO DO THE NEXT BIT
   (1)  017254  012601                    2$:      MOV     (SP)+,R1                ;;POP THE STACK INTO R1
   (1)  017256  016666  000002  000004             MOV     2(SP),4(SP)             ;;ADJUST THE STACK
   (1)  017264  012616                             MOV     (SP)+,(SP)
   (1)  017266  000002                             RTI                             ;;RETURN TO USER
   (1)  017270     000     000            $BIN:    .BYTE   0,0                     ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
 1339                                     .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)  017272                               $TYPDS:
    (3)  017272  010046                                MOV     R0,-(SP)          ;;PUSH R0 ON STACK
    (3)  017274  010146                                MOV     R1,-(SP)          ;;PUSH R1 ON STACK
    (3)  017276  010246                                MOV     R2,-(SP)          ;;PUSH R2 ON STACK
    (3)  017300  010346                                MOV     R3,-(SP)          ;;PUSH R3 ON STACK
    (3)  017302  010546                                MOV     R5,-(SP)          ;;PUSH R5 ON STACK
    (1)  017304  012746  020200                        MOV     #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
    (1)  017310  016605  000020                        MOV     20(SP),R5         ;;GET THE INPUT NUMBER
    (1)  017314  100004                                BPL     1$                ;;BR IF INPUT IS POS.
    (1)  017316  005405                                NEG     R5                ;;MAKE THE BINARY NUMBER POS.
    (1)  017320  112766  000055  000001                MOVB    #'-,1(SP)         ;;MAKE THE ASCII NUMBER NEG.
    (1)  017326  005000                       1$:      CLR     R0                ;;ZERO THE CONSTANTS INDEX
    (1)  017330  012703  017506                        MOV     #$DBLK,R3         ;;SETUP THE OUTPUT POINTER
    (1)  017334  112723  000040                        MOVB    #' ,(R3)+         ;;SET THE FIRST CHARACTER TO A BLANK
    (1)  017340  005002                       2$:      CLR     R2                ;;CLEAR THE BCD NUMBER
    (1)  017342  016001  017476                        MOV     $DTBL(R0),R1      ;;GET THE CONSTANT
    (1)  017346  160105                       3$:      SUB     R1,R5             ;;FORM THIS BCD DIGIT
    (1)  017350  002402                                BLT     4$                ;;BR IF DONE
    (1)  017352  005202                                INC     R2                ;;INCREASE THE BCD DIGIT BY 1
    (1)  017354  000774                                BR      3$
    (1)  017356  060105                       4$:      ADD     R1,R5             ;;ADD BACK THE CONSTANT
    (1)  017360  005702                                TST     R2                ;;CHECK IF BCD DIGIT=0
    (1)  017362  001002                                BNE     5$                ;;FALL THROUGH IF 0
    (1)  017364  105716                                TSTB    (SP)              ;;STILL DOING LEADING 0'S?
    (1)  017366  100407                                BMI     7$                ;;BR IF YES
    (1)  017370  106316                       5$:      ASLB    (SP)              ;;MSD?
    (1)  017372  103003                                BCC     6$                ;;BR IF NO
    (1)  017374  116663  000001  177777                MOVB    1(SP),-1(R3)      ;;YES--SET THE SIGN
    (1)  017402  052702  000060               6$:      BIS     #'0,R2            ;;MAKE THE BCD DIGIT ASCII
    (1)  017406  052702  000040               7$:      BIS     #' ,R2            ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
    (1)  017412  110223                                MOVB    R2,(R3)+          ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
    (1)  017414  005720                                TST     (R0)+             ;;JUST INCREMENTING
    (1)  017416  020027  000010                        CMP     R0,#10            ;;CHECK THE TABLE INDEX
    (1)  017422  002746                                BLT     2$                ;;GO DO THE NEXT DIGIT
    (1)  017424  003002                                BGT     8$                ;;GO TO EXIT
    (1)  017426  010502                                MOV     R5,R2             ;;GET THE LSD
    (1)  017430  000764                                BR      6$                ;;GO CHANGE TO ASCII
    (1)  017432  105726                       8$:      TSTB    (SP)+             ;;WAS THE LSD THE FIRST NON-ZERO?
    (1)  017434  100003                                BPL     9$                ;;BR IF NO
    (1)  017436  116663  177777  177776                MOVB    -1(SP),-2(R3)     ;;YES--SET THE SIGN FOR TYPING
    (1)  017444  105013                       9$:      CLRB    (R3)              ;;SET THE TERMINATOR
    (3)  017446  012605                                MOV     (SP)+,R5          ;;POP STACK INTO R5
    (3)  017450  012603                                MOV     (SP)+,R3          ;;POP STACK INTO R3
    (3)  017452  012602                                MOV     (SP)+,R2          ;;POP STACK INTO R2
    (3)  017454  012601                                MOV     (SP)+,R1          ;;POP STACK INTO R1
    (3)  017456  012600                                MOV     (SP)+,R0          ;;POP STACK INTO R0
    (1)  017460  104401  017506                        TYPE    ,$DBLK            ;;NOW TYPE THE NUMBER
    (1)  017464  016666  000002  000004                MOV     2(SP),4(SP)       ;;ADJUST THE STACK
    (1)  017472  012616                                MOV     (SP)+,(SP)
    (1)  017474  000002                                RTI                       ;;RETURN TO USER
    (1)  017476  023420                       $DTBL:   10000.
    (1)  017500  001750                                1000.
    (1)  017502  000144                                100.
    (1)  017504  000012                                10.
    (1)  017506  000004                       $DBLK:   .BLKW   4
```

```
 1341                                      .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)  017516  010046                      $TRAP:  MOV     R0,-(SP)         ;;SAVE R0
  (1)  017520  016600  000002                      MOV     2(SP),R0         ;;GET TRAP ADDRESS
  (1)  017524  005740                              TST     -(R0)            ;;BACKUP BY 2
  (1)  017526  111000                              MOVB    (R0),R0          ;;GET RIGHT BYTE OF TRAP
  (1)  017530  006300                              ASL     R0               ;;POSITION FOR INDEXING
  (1)  017532  016000  017552                      MOV     $TRPAD(R0),R0    ;;INDEX TO TABLE
  (1)  017536  000200                              RTS     R0               ;;GO TO ROUTINE
  (1)
  (1)
  (1)                                      ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
  (1)
  (1)  017540  011646                      $TRAP2: MOV     (SP),-(SP)       ;;MOVE THE PC DOWN
  (1)  017542  016666  000004  000002              MOV     4(SP),2(SP)      ;;MOVE THE PSW DOWN
  (1)  017550  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)  017552  017540                      $TRPAD: .WORD   $TRAP2
  (3)  017554  016166                              $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
  (3)  017556  017014                              $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
  (3)  017560  016770                              $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
  (3)  017562  017030                              $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
  (3)  017564  017272                              $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
  (3)  017566  017216                              $TYPBN  ;;CALL=TYPBN     TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
  (1)
  (3)  017570  014172                              $GTSWR  ;;CALL=GTSWR     TRAP+7(104407)  GET SOFT-SWR SETTING
  (1)
  (3)  017572  014102                              $CKSWR  ;;CALL=CKSWR     TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
  (3)  017574  014444                              $RDCHR  ;;CALL=RDCHR     TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
  (3)  017576  014534                              $RDLIN  ;;CALL=RDLIN     TRAP+12(104412) TTY TYPEIN STRING ROUTINE
  (3)  017600  015072                              $RDOCT  ;;CALL=RDOCT     TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
 1342  017602  004060                              TEST    ;;CALL=CHECK     TRAP+14(104414)
 1343  017604  004052                              TESTIT  ;;CALL=CHKIT     TRAP+15(104415)
 1344                                      .EVEN
 1345  017606  000240                              NOP                  ;JUST TO FIND THE LAST LOCATION OF THE PROGRAM
 1346          000001                      .END
```

```
ABASE = 170400             17#       54       93       94       95       96       97
ACDW1 = 000000             54
ACDW2 = 000000             54
ACPUOP= 000000             54
ADBUFF  001322             95#      260      319      327      343      368      391      403      416      435      444      455      466
                          689       717      746      789      820      849      877      977     1087     1151*    1155*    1194
ADDW0 = 000000             54
ADDW1 = 000000             54
ADDW10= 000000             54
ADDW11= 000000             54
ADDW12= 000000             54
ADDW13= 000000             54
ADDW14= 000000             54
ADDW15= 000000             54
ADDW2 = 000000             54
ADDW3 = 000000             54
ADDW4 = 000000             54
ADDW5 = 000000             54
ADDW6 = 000000             54
ADDW7 = 000000             54
ADDW8 = 000000             54
ADDW9 = 000000             54
ADEVCT= 000000             54
ADEVM = 000000             54
ADOT    011114            954      1271#
ADST1   001320             94#     1150*    1154*
ADV11C  001372            115#      196      663      684      712     1139*
AENV  = 000000             54
AENVM = 000000             54
AFATAL= 000000             54
AGATST  010242           1250#     1254
ACTST   010252           1105*     1112*    1118*    1252#
AMADR1= 000000             54
AMADR2= 000000             54
AMADR3= 000000             54
AMADR4= 000000             54
AMAMS1= 000000             54
AMAMS2= 000000             54
AMAMS3= 000000             54
AMAMS4= 000000             54
AMSG    011047           1265#
AMSGAD= 000000             54
AMSGLG= 000000             54
AMSGTY= 000000             54
AMTYP1= 000000             54
AMTYP2= 000000             54
AMTYP3= 000000             54
AMTYP4= 000000             54
APASS = 000000             54
APRIOR= 000200             19#       54
APTCSU= 000040           1334      1335#
APTENV= 000001           1331      1334     1335#
APTSIZ= 000200            159      1335#
APTSPO= 000100           1334      1335#
ASKTA   001456            141#      174      179      185      190      194      198      202
ASWREG= 000000             54
```

```
ATESTN= 000000          54
AUNIT = 000000          54
AUSWR = 000000          54
AVECT1= 000400          18#      54      98      99     100     101
AVECT2= 000000          54
AZERO   011116         956*     959*    960     962*    965*    966    1272#
BARF    001356         109#     129
BEGINA  007320         231      237    1109#   1145
BEGIND  007360         173     1122#
BEGINL  007302         233      239    1103#
BEGINW  007342         235      241    1116#
BEGIN0  001522          22      156#   1252
BEGIN2  001530          23      158#   1324
BEGL    002530         256#    1104    1110
BEGST   001534         157      159#
BIT0  = 000001          15#     354     383     387     448    1122
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#    1330
BIT09 = 001000          15#    1330    1331
BIT1  = 000002          15#    1125
BIT10 = 002000          15#    1331
BIT11 = 004000          15#    1330
BIT12 = 010000          15#     268    1023    1032
BIT13 = 020000          15#    1331
BIT14 = 040000          15#     272     372     374    1330
BIT15 = 100000          15#     300     313     323     372     374     386     441
BIT2  = 000004          15#     291    1128
BIT3  = 000010          15#     295    1131
BIT4  = 000020          15#     286     409     425     460    1134
BIT5  = 000040          15#     282     396     441     443    1137
BIT6  = 000100          15#     277     354     358
BIT7  = 000200          15#     321     358     386     396     409     425     441     460
BIT8  = 000400          15#     264
BIT9  = 001000          15#     109
BPTVEC= 000014          15#
BTEX    001402         119#     184*    192     458    1136*
CCHAN   011122         913      997    1274#
CH      011067         923      948    1267#
CHANL   001362         111#    1182*   1320
CHAN00= 000000          28#     507     539     691
CHAN01= 000001          29#     514     564
CHAN02= 000002          30#     521     589
CHAN03= 000003          31#     528     613     630     638     646     654
CHAN04= 000004          32#     544
CHAN05= 000005          33#     569
CHAN06= 000006          34#     594
CHAN07= 000007          35#     617
CHAN10= 000010          36#     552
CHAN11= 000011          37#     577
```

```
CHAN12= 000012       38#     602
CHAN13= 000013       39#     666     719
CHAN14= 000014       40#     748     794
CHAN15= 000015       41#     755     823
CHAN16= 000016       42#     762     852
CHAN17= 000017       43#     769     880
CHECK = 104414      316      324     335     344     388     401     414     433     463    1342#
CHKIT = 104415      265      273     278     283     287     292     296     301    1343#
CKSWR = 104410     1330     1331    1341#
CMSG    011042     1264#
COMPAR  010046      508      515     522     529     545     553     570     578     595     603     618     631     639
                    647      655     667     692     720     749     756     763     770     795     824     853     881
                   1208#
CONVRT  007710      506      513     520     527     538     543     551     563     568     576     588     593     601
                    612      616     665     690     718     747     754     761     768     793     822     851     879
                   1180#
CONVTR  007714      629      637     645     653     926    1008    1181#
CR    = 000015       15#    1334
CRLF  = 000200       15#     167    1334
CRWR    011276      428     1277#
DACA    001324       96#     261     491*    687*    697*    702*   1012*   1024*   1033*   1048*   1052*   1056*   1068*
                   1071*    1086*   1152*   1156*
DACB    001326       97#     262     492*    662*    715*    725*    730*   1013*   1025*   1034*   1049*   1053*   1057*
                   1069*    1072*   1153*   1157*
DAC0    001344      104#     787*    803*
DAC1    001346      105#     818*    832*
DAC2    001350      106#     847*    861*
DAC3    001352      107#     875*    889*
DDISP = 177570       15#      54     159
DF1     013544       71       78      84      90    1322#
DH1     013335       69     1314#
DH2     013375       88     1315#
DH3     013454       76       82    1316#
DISPLA  001142       54#     159*   1330*   1331*
DISPRE  000174       22#     159
DONE    011101     1235     1270#
DSWR  = 177570       15#      54     159
DT1     013504       70     1319#
DT2     013516       89     1320#
DT3     013534       77       83    1321#
DUMC    010164      360      367     376     379    1231#
DUMW    010112      348      370    1221#
ECHAN   011230     1276#
EMTVEC= 000030       15#     159*
EM1     013215       68     1310#
EM2     013237       75     1311#
EM3     013263       81     1312#
EM4     013310       87     1313#
ERRVEC= 000004       15#     159*   1148*   1330*
FIXONE  007506      168     1148#
GAIN00= 000000       45#     628
GAIN01= 000004       46#     636
GAIN10= 000010       47#     644
GAIN11= 000014       48#     652
GCHAN   011162      917     1001    1275#
GMSG    011054     1266#
```

```
GNS   = ****** U          22     167    1341    1342    1343
GTSWR = 104407           167    1341#
HEAD2   013144          1309#
HEAD5   011443          1280#
HT    = 000011            15#   1334
IOTST1  006306           242     908#
IOTST2  006462           243     940#
IOTST3  006664           244     996#
IOTST4  006772           245    1019#
IOTST5  007062           246    1043#
IOTST6  007150           247    1064#
IOTST7  007216           248    1078#
IOTVEC= 000020            15#    159*
KWAD    001374           116#    176     394     439    1124*
KWBPR   001342           103#    398*    411*    442*
KWCSR   001340           102#    399*    412*    445*    446
KWEX    001376           117#    181     407    1127*
LF    = 000012            15#   1334
MAEX    001400           118#    178*    187     421
METST   011463          1225    1281#
MSADV   012147           195    1292#
MSBTEX  012054           191    1290#
MSGNEX  012016           427    1288#
MSG70   012345           208    1297#
MSG71   011377           209    1279#
MSIO1   010376           909    1257#
MSIO2   010454           940    1258#
MSIO3   010514           996    1259#
MSIO4   010557          1019    1260#
MSIO5   010632          1043    1261#
MSIO6   010677          1064    1262#
MSIO7   010770          1078    1263#
MSKWAD  011505           144     175    1282#
MSKWEX  011567           180    1284#
MSMAEX  011676           186    1286#
MSTC1   012176           199    1293#
MSTC2   012255           203    1295#
OKCHAR  002434           219     230#
ONAD    011336          1227    1278#
OTHER   010044           628*    636*    644*    652*    919*    920*    921*    922*   1003*   1004*   1005*   1006*   1180*
                        1184    1205#
PIRQ  = 177772            15#
PIRQVE= 000240            15#
PR0   = 000000            15#
PR1   = 000040            15#
PR2   = 000100            15#
PR3   = 000140            15#
PR4   = 000200            15#
PR5   = 000240            15#
PR6   = 000300            15#
PR7   = 000340            15#
PS    = 177776            15#
PSW   = 177776            15#
PWRVEC= 000024            15#    159*   1328*
QUEST   011077           226    1269#
RDCHR = 104411          1324    1341#
```

```
RDLIN = 104412            145     211     429    1050    1054    1058    1326    1341#
RDOCT = 104413            914     918     998    1002    1341#
RESVEC= 000010             15#
RETERR  004102            481     483#
RETURN  001440            135#   1190
RST     010214           1239#
SLASH   011120           1273#
SPACE   011072            928     976    1094    1268#
SPREAD  001364            112#   1209*   1215    1320
STACK = 001100             15#    159
STALL   001426            129#    322     400     413     462    1070    1073
STKLMT= 177774             15#
STREG   001316             93#    259     305*    308     313*    320*    323*    331*    332     340*    341     354*    355
                          357     372*    373     380*    383*    384     387*    397*    404*    410*    417*    426*    436*
                          443*    448*    450     461*    465*    478*    479     688*    716*    745*    788*    819*    848*
                          876*    908*    971*    972*    973    1083*   1084    1149*   1186*   1192*   1228    1319    1320
                         1321
SWR     001140             54#    159*    167    1324*   1328*   1330    1331
SWREG   000176             22#    159     167    1324
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            113#    200     504     536     561     586     610     626     682     710    1130*
TC2     001370            114#    204     743     784     815     844     872    1133*
TEMP    001360            110#    156*    158*    169     540     541     565     566     590     591     614     929    1011*
                         1012    1013    1185*   1196*   1199*   1200*   1201*   1202*   1210
TEST    004060            479#   1342
TESTIT  004052            478#   1343
TKVEC = 000060             15#   1324*
TPVEC = 000064             15#
TRAPVE= 000034             15#    159*
TRTVEC= 000014             15#
```

```
TRYAG   002356               211#      227
TST1    002530               257#
TST10   002716               294#
TST11   002732               299#
TST12   002746               303#
TST13   003012               309       312#
TST14   003044               318#
TST15   003106               329#
TST16   003150               339#
TST17   003174               347#
TST2    002564               263#
TST20   003322               361       369#      377
TST21   003406               382#
TST22   003450               393#
TST23   003530               395       406#
TST24   003610               408       420#
TST25   003676               422       424       438#
TST26   004002               440       457#
TST27   004046               459       469#
TST3    002614               271#
TST30   004104               489#
TST31   004166               503#
TST32   004304               505       535#
TST33   004404               537       560#
TST34   004504               562       585#
TST35   004604               587       609#
TST36   004656               611       625#
TST37   005024               627       661#
TST4    002630               275#
TST40   005070               664       678#
TST41   005220               683       685       706#
TST42   005350               711       713       740#
TST43   005500               744       781#
TST44   005642               785       806       812#
TST45   006000               816       835       841#
TST46   006136               845       864       870#
TST47   006274               873       892       895#
TST5    002652               281#
TST6    002666               285#
TST7    002702               290#
TYPBN = 104406              1341#
TYPDS = 104405              1254      1341#
TYPE  = 104401               142       143       167       208       209       226       427       428       909       913       917       923       928
                             935       940       945       948       954       960       966       976       990       996       997      1001      1019
                            1043      1064      1078      1080      1094      1225      1227      1235      1254      1324      1328      1331      1332
                            1334      1337      1338      1339      1341#
TYPOC = 104402              1324      1332      1341#
TYPON = 104404              1341#
TYPOS = 104403               924       929       951       978      1088      1226      1228      1341#
UNEXP   001404               121#      276       363
VECTOR  001330                98#      276*      352*      363*     1158*     1159*     1160      1162      1164      1190*
VECTR1  001332                99#      353*     1160*     1161*
VECTR2  001334               100#      371*     1162*     1163*
VECTR3  001336               101#     1164*     1165*
VWRAP   001354               108#      510       517       524       531       633       641       649       657       694       722       797       826
                             855       883
```

```
V12      010240      1248#
V2       010236       547      555      572      580      597      605      620      669      751      758      765      772     1247#
WRAP     004104       489#    1111     1117
$APTHD   001000        53#
$ASTAT=  ****** U    1335
$ATYC    016546      1335#
$ATY1    016522      1335#
$ATY3    016530      1334     1335#
$ATY4    016540      1331     1335#
$AUTOB   001134        54#     167*     171     1324
$BASE    001250        54#    1149     1150     1151     1152     1153
$BDADR   001122        54#
$BDDAT   001126        54#     308*     357*     373*     450*     451      479*     480     1210*    1212     1319     1320     1321
$BELL    001164        54#    1324     1331
$BIN     017270      1338#*
$CDW1    001254        54#
$CHARC   016516      1334#*
$CKSWR   014102      1324#    1341
$CMTAG   001100        54#     159
$CM3   = 000000        54#
$CNTLC   015030      1324#
$CNTLG   015042      1324#
$CNTLU   015035      1324#
$CPUOP   001222        54#
$CRLF    001171        54#     142      935      945      990     1080     1324     1331     1332     1334
$DBLK    017506      1339#
$DEVCT   001204        54#
$DEVM    001252        54#    1122     1125     1128     1131     1134     1137
$DOAGN   010352      1254#
$DTBL    017476      1339#
$ENDAD   010342        51      167     1254#    1331
$ENDCT   010310       159     1254#
$ENDMG   010361      1254#
$ENULL   010356      1254#
$ENV     001214        54#     167     1331     1334     1335
$ENVM    001215        54#     159     1334     1335
$EOP     010254      1106     1113     1119     1254#
$EOPCT   010302       159*    1254#
$ERFLG   001103        54#     122*    1330*    1331*
$ERMAX   001115        54#     159*    1330*
$ERROR   015632       159     1331#
$ERRPC   001116        54#    1319     1320     1321     1331*    1332
$ERRTB   001256        54#    1332
$ERRTY   016032      1331     1332#
$ERTTL   001112        54#    1331*
$ESCAP   001162        54#     121*     124*     159*    1330*    1331
$ETABL   001214        54#
$ETEND   001256        53       54#
$FATAL   001176        54#    1335*
$FFLG    016766      1335#*
$FILLC   001156        54#    1334
$FILLS   001155        54#    1334
$GDADR   001120        54#
$GDDAT   001124        54#     264*     267*     268      272*     277*     282*     286*     291*     295*     300*     304*     321*
                      330*     358*     374*     386*     396*     397      409*     410      425*     426      441*     451      460*
                      461      478      480     1208*    1211     1319     1320
```

```
$GET42  010332          1254#
$GTSWR  014172          1324#   1341
$HD   = 000000            14
$HIBTS  001000            53#
$HIOCT  015172          1326#*
$ICNT   001104            54#   1330*
$ILLUP  015334          1328#
$INTAG  001135            54#   1324*
$ITEMB  001114            54#   1331*   1332
$LF     001172            54#   1324    1331    1334
$LFLG   016765          1335#*
$LPADR  001106            54#    159*    257*    494*   1224*   1234*   1330*
$LPERR  001110            54#    159*    493*   1223*   1233*   1330*   1331
$MADR1  001226            54#
$MADR2  001232            54#
$MADR3  001236            54#
$MADR4  001242            54#
$MAIL   001174            53      54#    159     167    1330    1331    1334
$MAMS1  001224            54#
$MAMS2  001230            54#
$MAMS3  001234            54#
$MAMS4  001240            54#
$MBADR  001002            53#
$MFLG   016764          1335#*
$MNEW   015060          1324#
$MSGAD  001210            54#   1335*
$MSGLG  001212            54#   1335*
$MSGTY  001174            54#   1335*
$MSWR   015047          1324#
$MTYP1  001225            54#
$MTYP2  001231            54#
$MTYP3  001235            54#
$MTYP4  001241            54#
$MXCNT  015630          1330#
$NULL   001154            54#   1334
$NWTST= 000001           257#    263#    271#    275#    281#    285#    290#    294#    299#    303#    312#    318#    329#
                         339#    347#    369#    382#    393#    406#    420#    438#    457#    469#    489#    503#    535#
                         560#    585#    609#    625#    661#    678#    706#    740#    781#    812#    841#    870#    895#
$OCNT   017212          1337#*
$OMODE  017214          1337#*
$OVER   015614          1330#
$PASS   001202            54#    159*    423    1221    1231    1254*   1330
$PASTM  001006            53#
$POWER  015342          1328#
$PWRDN  015174           159    1328#
$PWRMG  015330          1328#
$PWRUP  015246          1328#
$QUES   001170            54#   1324    1331    1334
$RDCHR  014444          1324#   1341
$RDDEC= ****** U        1341
$RDLIN  014534          1324#   1341
$RDOCT  015072          1326#   1341
$RDSZ = 000040          1324#
$RTNAD  010354          1254#
$R2A  = ****** U        1341
$SAVRE= ****** U        1341
```

```
$SAVR6  015340          1328#*
$SCOPE  015352           159      1330#
$SETUP= 000137            55#      159       167      1254      1324      1330      1331
$STUP = 177777            55#
$SVLAD  015560          1330#
$SVPC = 000106            51#
$SWR  = 167400             6#       14        16        54       159       257       263       271       275       281       285       290       294
                         299       303       312       318       329       339       347       369       382       393       406       420       438
                         457       469       489       503       535       560       585       609       625       661       678       706       740
                         781       812       841       870       895      1254      1328      1330      1331
$SWREG  001216            54#      159
$SWRMK= 000000            16      1330
$TESTN  001200            54#     1330*
$TIMES  001160            54#      159*      303*      312*      329*      489*      503*      535*      560*      585*      609*      625*      661*
                         678*      706*      740*      781*      812*      841*      870*      895*     1254*     1330*
$TKB    001146            54#     1324      1334
$TKCNT  013546          1324#*
$TKINT  013614           164      1324#
$TKQEN= 013614          1324#
$TKQIN  013550          1324#*
$TKQOU  013552          1324#*
$TKQSR  013554          1324#
$TKS    001144            54#      212*      307*      315*      337*     1240*     1250*     1324*     1334
$TKSRV  013664          1324#
$TN   = 000050             7#       14       257#      258       263#      271#      275#      281#      285#      290#      294#      299#      303#
                         309       312#      318#      329#      339#      347#      348       361       369#      370       382#      393#      395
                         406#      408       420#      422       424       438#      440       457#      459       469#      489#      503#      505
                         535#      537       560#      562       585#      587       609#      611       625#      627       661#      664       678#
                         683       685       706#      711       713       740#      744       781#      785       806       812#      816       835
                         841#      845       864       870#      873       892       895#
$TPB    001152            54#     1334*
$TPFLG  001157            54#     1334
$TPS    001150            54#     1334
$TRAP   017516           159      1341#
$TRAP2  017540          1341#
$TRP  = 000016          1341#     1342#     1343#
$TRPAD  017552          1341#
$TSTM   001004            53#
$TSTNM  001102            54#      258*      489*     1254*     1330*     1331
$TTYIN  014770          1324#
$TYPBN  017216          1338#     1341
$TYPDS  017272          1339#     1341
$TYPE   016166           160*      161*      162*      163*     1334#     1335      1341
$TYPEC  016400          1324      1334#
$TYPEX  016520          1334#
$TYPOC  017014          1337#     1341
$TYPON  017030          1337#     1341
$TYPOS  016770          1337#     1341
$UNIT   001206            54#
$UNITM  001010            53#
$USWR   001220            54#
$VECT1  001244            54#     1158
$VECT2  001246            54#
$XOFF = 000023          1334
$XON  = 000021          1334
$XTSTR  015364          1330#
```

```
$$GET4= 000000          1254#
$OFILL  017213          1337#*
$40CAT= ****** U        1330     1331
.       = 017610          22#      25#      51#      52#      53#      54#     159     1254     1324#    1328     1330     1331     1332#
                        1334     1335#    1339#
.$ASTA= ****** U        1335
.$X     = 001000          53#
```

```
COMMEN    15#
DUMWRN    59#   348    370
ENDCOM    15#
ERROR     15#   123    266    274    279    284    288    293    297    302    310    317    325    336    345
         359    375    389    402    415    434    453    464    511    518    525    532    548    556    573
         581    598    606    621    634    642    650    658    670    701    729    752    759    766    773
         808    837    866    894
ESCAPE    15#   121
GETPRI    15#
GETSWR    15#   167#
MULT      15#
NEWTST    15#   257    263    271    275    281    285    290    294    299    303    312    318    329    339
         347    369    382    393    406    420    438    457    469    489    503    535    560    585    609
         625    661    678    706    740    781    812    841    870    895
POP       15#  1326   1328   1335   1339
PUSH      15#  1326   1328   1335   1339
REPORT    15#
SCOPE     15#   263    271    275    281    285    290    294    299    303    312    318    329    339    347
         369    382    393    406    420    438    457    469    503    535    560    585    609    625    661
         678    706    740    781    812    841    870    895
SETPRI    15#  1324
SETTRA  1341#  1342   1343
SETUP     15#   159
SKIP      15#   269    309    361    395    408    422    424    440    452    459    481    505    537    562
         587    611    627    664    683    685    700    711    713    728    744    785    806    816    835
         845    864    873    892
SLASH     15#
SPACE     15#
STARS     15#    51     53     54    257    263    271    275    281    285    290    294    299    303    312
         318    329    339    347    369    382    393    406    420    438    457    469    489    503    535
         560    585    609    625    661    678    706    740    781    812    841    870    895   1254   1324
        1326   1328   1330   1331   1332   1334   1335   1337   1338   1339   1341
SWRSU     15#   159#
TRMTRP  1341#
TYPBIN    15#
TYPDEC    15#  1254
TYPNAM    15#   167
TYPNUM    15#
TYPOCS    15#   924    929   1226   1228
TYPOCT    15#  1324   1332
TYPTXT    15#
$$CMRE    54#
$$CMTM    54#
$$ESCA    15#
$$NEWT    15#   257    263    271    275    281    285    290    294    299    303    312    318    329    339
         347    369    382    393    406    420    438    457    469    489    503    535    560    585    609
         625    661    678    706    740    781    812    841    870    895
$$SET   1341#  1342   1343
$$SETM   159#
$$SKIP    15#   309    361    395    408    422    424    440    459    505    537    562    587    611    627
         664    683    685    711    713    744    785    806    816    835    845    864    873    892
.EQUAT     8#    15
.HEADE     8#    14
.SETUP    10#    55
.SWRHI    10#    16
.SWRLO    16#
```

```
.$ACT1      11#      51
.$APTB      11#      54#
.$APTH      11#      53
.$APTY      11#    1335
.$CATC       8#      22
.$CMTA       8#      54
.$EOP        8#    1254
.$ERRO       8#    1331
.$ERRT      10#    1332
.$PARM       9#
.$POWE       9#    1328
.$RAND      11#
.$RDOC      11#    1326
.$READ       9#    1324
.$SAVE       9#
.$SCOP       9#    1330
.$SPAC      10#
.$SWDO      10#
.$TRAP      10#    1341
.$TYPB       9#    1338
.$TYPD      11#    1339
.$TYPE      10#    1334
.$TYPO       9#    1337

. ABS.  017610      000      CON    RW    ABS    LCL    D

 ERRORS DETECTED:  0

 CVAXAA,CVAXAA/CRF=CVAXAA
 RUN-TIME: 23 10 1 SECONDS
 RUN-TIME RATIO: 955/34=27.3
 CORE USED:  26K  (51 PAGES)
```