%

IDENTIFICATION

PRODUCT CODE: AC-T429A-MC

PRODUCT NAME: CNAXAA0 AXV11-C/ADV11-C

PRODUCT DATE: DECEMBER, 1982

MAINTAINER: Diagnostic Services/ISS

AUTHOR: DIAG/ISS

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

PDP11/21 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.

10. REVISION HISTORY
    CVAXAA1 DIAGNOSTIC WAS MADE SPECIFIC TO 11/21 PROCESSOR
    BY LOWERING PRIORITY 7 TO 6 AND ASSEMBLED WITH CNMAC1.SML
    AND RENAMED TO CNAXAA0.

```
5675                              ;DEVELOPED USING CNMAC2.SML
5676                              ;CVAXAA DIAGNOSTIC WAS MODIFIED TO RUN ON 11/21 PROCESSOR
5677                              ;BY CHANGING PRIORITY TO 300 INSTEAD OF 340 AND ADDING A CALL
5678                              ;TO CNMAC2.SML. ALSO DEFAULT ADDRESS AND VECTOR WERE CHANGED.
5691                              .TITLE   MAINDEC-11-CNAXA-A
 (1)                              ;*COPYRIGHT (C) 1982
 (1)                              ;*DIGITAL EQUIPMENT CORP.
 (1)                              ;*MAYNARD, MASS. 01754
 (1)                              ;*
 (1)                              ;*PROGRAM BY R.SHOOP
 (1)                              ;*
 (1)                              ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
 (1)                              ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
 (1)                              ;*
5692                              .SBTTL   BASIC DEFINITIONS
 (1)
 (1)                              ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
 (1)          001100             STACK=  1100
 (1)                             .EQUIV   EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
 (1)                             .EQUIV   IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
 (1)
 (1)                              ;*MISCELLANEOUS DEFINITIONS
 (1)          000011             HT=      11             ;;CODE FOR HORIZONTAL TAB
 (1)          000012             LF=      12             ;;CODE FOR LINE FEED
 (1)          000015             CR=      15             ;;CODE FOR CARRIAGE RETURN
 (1)          000200             CRLF=    200            ;;CODE FOR CARRIAGE RETURN-LINE FEED
 (1)          177776             PS=      177776         ;;PROCESSOR STATUS WORD
 (1)                             .EQUIV   PS,PSW
 (1)          177774             STKLMT=  177774         ;;STACK LIMIT REGISTER
 (1)          177772             PIRQ=    177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
 (1)          177570             DSWR=    177570         ;;HARDWARE SWITCH REGISTER
 (1)          177570             DDISP=   177570         ;;HARDWARE DISPLAY REGISTER
 (1)                              ;***** THE FOLLOWING ODT START ADDRESS FOR SBC 11/21 IS ADDED
 (1)          170000             ODTST=   170000
 (1)                              ;*GENERAL PURPOSE REGISTER DEFINITIONS
 (1)          000000             R0=      %0             ;;GENERAL REGISTER
 (1)          000001             R1=      %1             ;;GENERAL REGISTER
 (1)          000002             R2=      %2             ;;GENERAL REGISTER
 (1)          000003             R3=      %3             ;;GENERAL REGISTER
 (1)          000004             R4=      %4             ;;GENERAL REGISTER
 (1)          000005             R5=      %5             ;;GENERAL REGISTER
 (1)          000006             R6=      %6             ;;GENERAL REGISTER
 (1)          000007             R7=      %7             ;;GENERAL REGISTER
 (1)          000006             SP=      %6             ;;STACK POINTER
 (1)          000007             PC=      %7             ;;PROGRAM COUNTER
 (1)
 (1)                              ;*PRIORITY LEVEL DEFINITIONS
 (1)          000000             PR0=     0              ;;PRIORITY LEVEL 0
 (1)          000040             PR1=     40             ;;PRIORITY LEVEL 1
 (1)          000100             PR2=     100            ;;PRIORITY LEVEL 2
 (1)          000140             PR3=     140            ;;PRIORITY LEVEL 3
 (1)          000200             PR4=     200            ;;PRIORITY LEVEL 4
 (1)          000240             PR5=     240            ;;PRIORITY LEVEL 5
 (1)          000300             PR6=     300            ;;PRIORITY LEVEL 6
 (1)          000340             PR7=     340            ;;PRIORITY LEVEL 7
 (1)
```

```
(1)                                    ;*"SWITCH REGISTER" SWITCH DEFINITIONS
(1)               100000               SW15=   100000
(1)               040000               SW14=   40000
(1)               020000               SW13=   20000
(1)               010000               SW12=   10000
(1)               004000               SW11=   4000
(1)               002000               SW10=   2000
(1)               001000               SW09=   1000
(1)               000400               SW08=   400
(1)               000200               SW07=   200
(1)               000100               SW06=   100
(1)               000040               SW05=   40
(1)               000020               SW04=   20
(1)               000010               SW03=   10
(1)               000004               SW02=   4
(1)               000002               SW01=   2
(1)               000001               SW00=   1
(1)                                    .EQUIV  SW09,SW9
(1)                                    .EQUIV  SW08,SW8
(1)                                    .EQUIV  SW07,SW7
(1)                                    .EQUIV  SW06,SW6
(1)                                    .EQUIV  SW05,SW5
(1)                                    .EQUIV  SW04,SW4
(1)                                    .EQUIV  SW03,SW3
(1)                                    .EQUIV  SW02,SW2
(1)                                    .EQUIV  SW01,SW1
(1)                                    .EQUIV  SW00,SW0
(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)                                     ;***** THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED
    (1)             000100                  LKVEC= 100               ;;LINE CLOCK VECTOR
    (1)             000140                  BRKVEC= 140              ;;BREAK VECTOR
    (1)             000240                  PIRQVEC=240              ;;PROGRAM INTERRUPT REQUEST VECTOR
   5693                                     .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>
   5694            175400                   ABASE= 175400
   5695            000220                   AVECT1= 220
   5696            000200                   APRIOR= 200
   5697
   5698
   5699                                     .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                   @#BEGIN0        ;;JUMP TO STARTING ADDRESS OF PROGRAM
   5700     000204  000137  001530          JMP      @#BEGIN2        ;RESTART ADDRESS
   5701
   5702            000100                   .=400
   5703     000100  000104  000340  000002  10 ,340,2                ;''B EVENT'' HANDLER
   5704
   5705            000000                   CHAN00= 00
   5706            000001                   CHAN01= 01
   5707            000002                   CHAN02= 02
   5708            000003                   CHAN03= 03
   5709            000004                   CHAN04= 04
   5710            000005                   CHAN05= 05
   5711            000006                   CHAN06= 06
   5712            000007                   CHAN07= 07
```

```
5713           000010                    CHAN10= 10
5714           000011                    CHAN11= 11
5715           000012                    CHAN12= 12
5716           000013                    CHAN13= 13
5717           000014                    CHAN14= 14
5718           000015                    CHAN15= 15
5719           000016                    CHAN16= 16
5720           000017                    CHAN17= 17
5721
5722           000000                    GAIN00= 00
5723           000004                    GAIN01= 04
5724           000010                    GAIN10= 10
5725           000014                    GAIN11= 14
5726
5727
5728                             .SBTTL  ACT11 HOOKS

 (1)                            ;;********************************************************
 (2)                            ;;HOOKS REQUIRED BY ACT11
 (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
5729           001000            .=1000
5730                             .SBTTL  APT PARAMETER BLOCK

 (1)                            ;;********************************************************
 (2)                            ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
 (1)                            ;;********************************************************
 (2)           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)
```

```
 5731                              .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
   (1)                                                 ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
   (1)  001156     012             $FILLC: .BYTE  12                 ;;INSERT FILL CHARS. AFTER A ''LINE FEED''
   (1)  001157     000             $TPFLG: .BYTE  0                  ;;''TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
   (1)  001160  000000             $TIMES: 0                         ;;MAX. NUMBER OF ITERATIONS
   (1)  001162  000000             $ESCAPE:0                         ;;ESCAPE ON ERROR ADDRESS
   (1)  001164  177607   000377    $BELL:  .ASCIZ <207><377><377> ;;CODE FOR BELL
   (1)  001170     077             $QUES:  .ASCII /?/                ;;QUESTION MARK
   (1)  001171     015             $CRLF:  .ASCII <15>               ;;CARRIAGE RETURN
   (1)  001172  000012             $LF:    .ASCIZ <12>               ;;LINE FEED
   (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
(2)                               ;;ENVIRONMENT MODE BITS
(2)   001216  000000              $SWREG: .WORD    ASWREG   ;;APT SWITCH REGISTER
(2)   001220  000000              $USWR:  .WORD    AUSWR    ;;USER SWITCHES
(2)   001222  000000              $CPUOP: .WORD    ACPUOP   ;;CPU TYPE,OPTIONS
(2)                               ;*                        BITS 15-11=CPU TYPE
(2)                               ;*                            11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                               ;*                            11/70=06,PDQ=07,Q=10
(2)                               ;*                        BIT 10=REAL TIME CLOCK
(2)                               ;*                        BIT  9=FLOATING POINT PROCESSOR
(2)                               ;*                        BIT  8=MEMORY MANAGEMENT
(2)   001224     000              $MAMS1: .BYTE    AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
(2)   001225     000              $MTYP1: .BYTE    AMTYP1   ;;MEM. TYPE,BLK#1
(2)                               ;*                        MEM.TYPE BYTE    --  (HIGH BYTE)
(2)                               ;*                            900 NSEC CORE=001
(2)                               ;*                            300 NSEC BIPOLAR=002
(2)                               ;*                            500 NSEC MOS=003
(2)   001226  000000              $MADR1: .WORD    AMADR1   ;;HIGH ADDRESS,BLK#1
(2)                               ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
(2)   001230     000              $MAMS2: .BYTE    AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
(2)   001231     000              $MTYP2: .BYTE    AMTYP2   ;;MEM.TYPE,BLK#2
(2)   001232  000000              $MADR2: .WORD    AMADR2   ;;MEM.LAST ADDRESS,BLK#2
(2)   001234     000              $MAMS3: .BYTE    AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
(2)   001235     000              $MTYP3: .BYTE    AMTYP3   ;;MEM.TYPE,BLK#3
(2)   001236  000000              $MADR3: .WORD    AMADR3   ;;MEM.LAST ADDRESS,BLK#3
(2)   001240     000              $MAMS4: .BYTE    AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
(2)   001241     000              $MTYP4: .BYTE    AMTYP4   ;;MEM.TYPE,BLK#4
(2)   001242  000000              $MADR4: .WORD    AMADR4   ;;MEM.LAST ADDRESS,BLK#4
(2)   001244  000220              $VECT1: .WORD    AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)   001246  000000              $VECT2: .WORD    AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)   001250  175400              $BASE:  .WORD    ABASE
(2)                               ;;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                       $ERRTB:
   5733
   5734
   5735
   5744                                 ;ITEM    1
   5745    001256   013215                  EM1                                 ;STATUS REG. ERROR
   5746    001260   013335                  DH1                                 ;ERRPC STREG EXPECTED ACTUAL
   5747    001262   013504                  DT1                                 ;$ERRPC, STREG, $GDDAT, $BDDAT
   5748    001264   013544                  DF1
   5749
   5750
   5751                                 ;ITEM    2
   5752    001266   013237                  EM2                                 ;FAILED TO INTERRUPT
   5753    001270   013454                  DH3                                 ;ERRPC STREG ACTUAL
   5754    001272   013534                  DT3                                 ;$ERRPC, STREG,  $BDDAT
   5755    001274   013544                  DF1
   5756
   5757                                 ;ITEM    3
   5758    001276   013263                  EM3                                 ;UNEXPECTED INTERRUPT
   5759    001300   013454                  DH3                                 ;ERRPC STREG
   5760    001302   013534                  DT3                                 ;$ERRPC, STREG
   5761    001304   013544                  DF1
   5762
   5763                                 ;ITEM    4
   5764    001306   013310                  EM4                                 ;ERROR ON A/D CHANNEL
   5765    001310   013375                  DH2                                 ;ERRPC  STREG  CHAN  NOMINAL  TOL  ACTUAL
   5766    001312   013516                  DT2                                 ;$ERRPC,STREG,CHANL,$GDDAT,SPREAD,$BDDAT
   5767    001314   013544                  DF1
```

```
5769
5770                                    .SBTTL        MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
5771   001316  175400           STREG:  ABASE                   ;ADDRESS OF STATUS REGISTER
5772   001320  175401           ADST1:  ABASE+1                 ;UPPER BYTE OF STATUS REG.
5773   001322  175402           ADBUFF: ABASE+2                 ;ADDRESS OF A/D BUFFER
5774   001324  175404           DACA:   ABASE+4                 ;ADDRESS OF D TO A ''A''
5775   001326  175406           DACB:   ABASE+6                 ;ADDRESS OF D TO A 'B'
5776   001330  000220           VECTOR: AVECT1                  ;VECTOR ADDRESS
5777   001332  000222           VECTR1: AVECT1+2
5778   001334  000224           VECTR2: AVECT1+4                ;ERROR VECTOR ADDRESS
5779   001336  000226           VECTR3: AVECT1+6
5780   001340  170420           KWCSR:  170420                  ;CLOCK STATUS/CONTROL REGISTER
5781   001342  170422           KWBPR:  170422                  ;CLOCK PRESET/COUNTER REGISTER
5782   001344  170440           DAC0:   170440                  ;AAV11-C DAC ''A'' ADDRESS
5783   001346  170442           DAC1:   170442                  ;              'B'
5784   001350  170444           DAC2:   170444                  ;              ''C''
5785   001352  170446           DAC3:   170446                  ;              'D'
5786   001354  000020           VWRAP:  20
5787   001356  001000           BARF:   BIT9                    ;DELAY FACTOR
5788   001360  000000           TEMP:   0                       ;WORK AREA
5789   001362  000000           CHANL:  0                       ;CHANNEL VALUE
5790   001364  000000           SPREAD: 0                       ;DEVIATION FROM THE NOMINAL
5791   001366  000000           TC1:    0          ;NON-ZERO, AXV11-C TEST FIXTURE IS INSTALLED
5792   001370  000000           TC2:    0          ;NON-ZERO, AAV11-C TO AXV11-C CABLE IN INSTALLED
5793   001372  000000           ADV11C: 0          ;NON-ZERO, MODULE IS ADV11-C (NO DAC'S ON BOARD)
5794   001374  000000           KWAD:   0          ;NON-ZERO, CLOCK CONNECTED TO RTC IN
5795   001376  000000           KWEX:   0          ;NON-ZERO, JUMPER F2 IS INSTALLED AND CLOCK CONNECTED TO EXT TRIG
5796   001400  000000           MAEX:   0          ;NON-ZERO, JUMPER F2 IS INSTALLED AND MANUAL TRIGGER IS CONNECTED
5797   001402  000000           BTEX:   0          ;NON-ZERO, JUMPER F1 IS INSTALLED
5798
5799   001404                   UNEXP:
 (1)   001404  012737  001420  001162    MOV     #1$,$ESCAPE    ;;ESCAPE TO 1$ ON ERROR
5800   001412  005237  001103           INC     $ERFLG
5801   001416  104003                    ERROR   3
5802   001420  005037  001162    1$:     CLR     $ESCAPE        ;RETURN ESCAPE TO NORMAL
5803   001424  000002                    RTI                    ;UNEXPECTED INTERRUPT
5804
5805                            ;SUBROUTINE TO DELAY AN AMOUNT OF CPU TIME
5806
5807   001426  013700  001356    STALL:  MOV     BARF,R0        ;GET DELAY FACTOR
5808   001432  005300            1$:     DEC     R0             ;DELAY
5809   001434  001376                    BNE     1$
5810   001436  000207                    RTS     PC             ;EXIT
```

```
5812
5813
5814   001440   022776   000001   000000   RETURN: CMP      #1,@0(SP)      ;DOES IT RETURN TO A WAIT?
5815   001446   001002                               BNE      1$             ;NO
5816   001450   062716   000002                      ADD      #2,(SP)        ;BUMP RETURN ADDRESS
5817   001454   000002                      1$:      RTI
5818
5819                                        ;SUBROUTINE TO ASK QUESTIONS OF THE OPERATOR
5820   001456   012537   001470   ASKTA:    MOV      (R5)+,10$      ;GET THE ASCII POINTER
5821   001462   104401   001171             TYPE     ,$CRLF         ;MAKE A FRESH LINE
5822   001466   104401                      TYPE                    ;TELL THE OPERATOR A MESSAGE
5823   001470   011505                      10$:     MSKWAD
5824   001472   104412                      RDLIN
5825   001474   012600                      MOV      (SP)+,R0       ;GET ANSWER
5826   001476   005075   000000             CLR      @(R5)          ;IF ANSWER IS NOT A ''Y'', CLEAR MESSAGE FLAG
5827   001502   042710   000040             BIC      #40,(R0)       ;ENSURE UPPER CASE
5828   001506   122710   000131             CMPB     #'Y,(R0)       ;TEST IF ''Y''
5829   001512   001001                      BNE      1$             ;BR IF NOT
5830   001514   005235                      INC      @(R5)+         ;SET YES FLAG
5831   001516   005725                      1$:      TST      (R5)+          ;BUMP EXIT
5832   001520   000205                      RTS      R5             ;EXIT
```

```
5834
5835                                      .SBTTL        INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
5836   001522  005037  001360     BEGIN0: CLR     TEMP            ;CLEAR RESTART FLAG
5837   001526  000402                     BR      BEGST
5838   001530  005237  001360     BEGIN2: INC     TEMP            ;SET RESTART FLAG
5839   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     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
 (1)   001562  012737  000300  000022    MOV     #PR6,@#IOTVEC+2 ;;LEVEL 6
 (1)   001570  012737  015632  000030    MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
 (1)   001576  012737  000300  000032    MOV     #PR6,@#EMTVEC+2 ;;LEVEL 6
 (1)                              ;;BIT02
 (1)   001604  012737  017444  000034    MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
 (1)   001612  012737  000300  000036    MOV     #PR6,@#TRAPVEC+2;LEVEL 6
 (1)   001620  012737  015174  000024    MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
 (1)   001626  012737  000300  000026    MOV     #PR6,@#PWRVEC+2 ;;LEVEL 6
 (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     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
 (2)   002004                     67$:
5840   002004  012737  005046  016166    MOV     #5046,$TYPE     ;A WAY TO LOWER
5841   002012  012737  012746  016170    MOV     #12746,$TYPE+2  ; PS FOR
5842   002020  012737  016200  016172    MOV     #$TYPE+12,$TYPE+4
5843   002026  012737  000002  016174    MOV     #RTI,$TYPE+6    ;   TTY OUTPUT
5844   002034  004737  013614            JSR     PC,$TKINT       ;INIT THE CONSOLE VECTORS
```

I 2

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 61
CNAXAA.P11   23-DEC-82 14:21   INITIALIZE THE COMMON TAGS   SEQ 0021

```
5846
5847                                    .SBTTL         DIALOGUE TO DETERMINE WHICH TEST TO RUN
5848                                    .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$:  .ASCIZ  <CRLF># CNAXAA  AXV11-C/ADV11-C DIAGNOSTIC  #<CRLF>
  (1)    002174                         68$:
5849    002174  004737  007506                JSR     PC,FIXONE       ;INITIALIZE ADDRESSES
5850    002200  005737  001360         77$:    TST     TEMP            ;ARE WE RESTARTING THE PROGRAM
5851    002204  001062                         BNE     40$             ;BR IF YES
5852    002206  005737  001134                TST     $AUTOB          ;IS IT CHAINED?
5853    002212  001402                        BEQ     1$
5854    002214  000137  007360                JMP     BEGIND  ;RUN ONLY THE LOGIC TEST AND SELECTED WRAPAROUND IF APT/XXDP CHA
5855    002220  004537  001456         1$:     JSR     R5,ASKTA        ;ASK OPERATOR ABOUT DIFFERENT CONFIG.
5856    002224  011505                        MSKWAD                  ;IS KWV11-C CONNECTED TO CLOCK START
5857    002226  001374                        KWAD                    ;
5858    002230  000240                        NOP
5859    002232  005037  001400                CLR     MAEX            ;ENSURE CLEARED FLAG
5860    002236  004537  001456                JSR     R5,ASKTA        ;ASK IF KWV11-C CONNECTED TO EXT. START
5861    002242  011567                        MSKWEX
5862    002244  001376                        KWEX
5863    002246  000403                        BR      2$
5864    002250  000415                        BR      4$              ;IF ANSWER WAS YES, BYPASS NEXT QUESTION
5865    002252  005037  001402                CLR     BTEX            ;ENSURE CLEARED FLAG
5866    002256  004537  001456         2$:     JSR     R5,ASKTA        ;ASK IF MANUAL TRIGGER IS CONNECTED TO EXT. START
5867    002262  011676                        MSMAEX
5868    002264  001400                        MAEX
5869    002266  000401                        BR      3$
5870    002270  000405                        BR      4$
5871    002272  004537  001456         3$:     JSR     R5,ASKTA        ;ASK IF B EVENT IS CONNECTED TO EXT TRIG
5872    002276  012054                        MSBTEX
5873    002300  001402                        BTEX
5874    002302  000240                        NOP
5875    002304  004537  001456         4$:     JSR     R5,ASKTA        ;ASK IF MODULE IS ADV11-C
5876    002310  012147                        MSADV
5877    002312  001372                        ADV11C
5878    002314  000240                        NOP
5879    002316  004537  001456         10$:    JSR     R5,ASKTA        ;ASK IF TEST FIXTURE #1 IS INSTALLED
5880    002322  012176                        MSTC1
5881    002324  001366                        TC1
```

J 2

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 61-1
CNAXAA.P11   23-DEC-82 14:21   GET VALUE FOR SOFTWARE SWITCH REGISTER   SEQ 0022

```
5882  002326  000240                        NOP
5883  002330  004537  001456    11$:  JSR    R5,ASKTA       ;ASK IF TEST CONNECTOR #2 IS INSTALLED
5884  002334  012255                        MSTC2
5885  002336  001370                        TC2
5886  002340  000240                        NOP
5887  002342  000240            12$:  NOP
5888  002344  000240            20$:  NOP
5889  002346  104401  012345    30$:  TYPE,  MSG70          ;TELL THE OPERATOR THE TESTS AVAILABLE
5890  002352  104401  011377    40$:  TYPE   ,MSG71
5891                             ;ROUTINE TO ASK OPERATOR WHAT SUB-SECTION TO EXECUTE
5892  002356  104412            TRYAG: RDLIN
5893  002360  052777  000100 176556   BIS    #100,a$TKS
5894  002366  005046                  CLR    -(SP)          ;CLEAR PSW
5895  002370  012746  002376          MOV    #1$,-(SP)
5896  002374  000002                  RTI
5897  002376  012600            1$:   MOV    (SP)+,R0       ;READ ANSWER
5898  002400  011000                  MOV    (R0),R0        ;GET THE 1ST CHARACTER
5899  002402  042700  177600          BIC    #177600,R0     ;REMOVE EXTRA BITS
5900  002406  012701  002434          MOV    #OKCHAR,R1     ;LOAD POINTER TO GOOD CHARACTER LIST
5901  002412  020021            2$:   CMP    R0,(R1)+       ;CHECK IF VALID CHARACTER
5902  002414  001002                  BNE    3$             ;BR IF NOT
5903  002416  011101                  MOV    (R1),R1        ;GET THE ADDRESS
5904  002420  000111                  JMP    aR1            ;DO THE SELECTED SUB-TEST
5905  002422  005721            3$:   TST    (R1)+          ;BUMP THE POINTER
5906  002424  001372                  BNE    2$             ;BR IF MORE CHARACTERS
5907  002426  104401  011077    6$:   TYPE   ,QUEST
5908  002432  000751                  BR     TRYAG          ;WAIT FOR CHARACTER
5909
5910                             ;TABLE OF VALID MENU CHARACTERS AND STARTING ADDRESS
5911  002434  000141            OKCHAR: 141                 ;LOWER CASE ''A''
5912  002436  007320                    BEGINA
5913  002440  000154                    154                 ;LOWER CASE ''L''
5914  002442  007302                    BEGINL
5915  002444  000167                    167                 ;LOWER CASE ''W''
5916  002446  007342                    BEGINW
5917  002450  000101                    'A
5918  002452  007320                    BEGINA
5919  002454  000114                    'L
5920  002456  007302                    BEGINL
5921  002460  000127                    'W
5922  002462  007342                    BEGINW
5923  002464  000061  006306            '1     ,IOTST1
5924  002470  000062  006462            '2     ,IOTST2
5925  002474  000063  006664            '3,    IOTST3
5926  002500  000064  006772            '4,    IOTST4
5927  002504  000065  007062            '5,    IOTST5
5928  002510  000066  007150            '6,    IOTST6
5929  002514  000067  007216            '7,    IOTST7
5930  002520  000000  000000  000000    0,0,0,0
      002526  000000
```

```
5932
5938   002530                        BEGL:
5939                                  ;;**********************************************************
 (3)                                  ;*TEST 1           ADDRESS THE 4 BUS ADDRESSES OF THE AXV11-C
 (3)                                  ;;**********************************************************
 (2)   002530  012737  002530  001106 TST1:   MOV     #TST1,$LPADR
 (2)
5940   002536  012737  000001  001102         MOV     #$TN-1,$TSTNM    ;LOAD TEST NUMBER
5941   002544  005777  176546                 TST     @STREG           ;ADDRESS A/D STATUS REGISTER
5942   002550  005777  176546                 TST     @ADBUFF          ;ADDRESS A/D DATA BUFFER
5943   002554  005777  176544                 TST     @DACA            ;ADDRESS D TO A "A"
5944   002560  005777  176542                 TST     @DACB            ;ADDRESS D TO A 'B'
5945                                  ;;**********************************************************
 (3)                                  ;*TEST 2           FLOAT A ONE THRU MULTIPLEXER (BITS 11-8)
 (3)                                  ;;**********************************************************
 (2)   002564  000004                 TST2:   SCOPE
 (2)
5946   002566  012737  000400  001124         MOV     #BIT8,$GDDAT     ;LOAD FIRST BIT
5947   002574  104415                 2$:     CHKIT
5948   002576  104001                         ERROR   1                ;FAILED TO LOAD + READ BIT
5949   002600  006337  001124         1$:     ASL     $GDDAT           ;GET NEXT BIT
5950   002604  023727  001124  010000         CMP     $GDDAT,#BIT12    ;FINISHED?
5951   002612  001370                         BNE     2$               ;;NO,GO TO NEXT TEST
5952
5953                                  ;;**********************************************************
 (3)                                  ;*TEST 3           LOAD AND READ BACK ERROR I.E. BIT14
 (3)                                  ;;**********************************************************
 (2)   002614  000004                 TST3:   SCOPE
 (2)
5954   002616  012737  040000  001124         MOV     #BIT14,$GDDAT
5955   002624  104415                         CHKIT
5956   002626  104001                         ERROR   1                ;FAILED TO LOAD + READ ERROR I.E.
5957                                  ;;**********************************************************
 (3)                                  ;*TEST 4           LOAD AND READ BACK INTERRUPT ENABLE BIT6
 (3)                                  ;;**********************************************************
 (2)   002630  000004                 TST4:   SCOPE
 (2)
5958   002632  012777  001404  176470         MOV     #UNEXP,@VECTOR   ;SETUP FOR UNEXPECTED INTERUPT
5959   002640  012737  000100  001124         MOV     #BIT6,$GDDAT     ;LOAD EXPECTED DATA
5960   002646  104415                         CHKIT
5961   002650  104001                         ERROR   1                ;FAILED TO LOAD + READ INTERRUPT ENABLE
5962
5963                                  ;;**********************************************************
 (3)                                  ;*TEST 5           LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BIT5
 (3)                                  ;;**********************************************************
 (2)   002652  000004                 TST5:   SCOPE
 (2)
5964   002654  012737  000040  001124         MOV     #BIT5,$GDDAT     ;LOAD EXPECTED DATA
5965   002662  104415                         CHKIT
5966   002664  104001                         ERROR   1                ;FAILED TO LOAD + READ CLOCK OVERFLOW START ENABLE
5967                                  ;;**********************************************************
 (3)                                  ;*TEST 6           LOAD AND READ BACK EXTERNAL START ENABLE BIT4
 (3)                                  ;;**********************************************************
 (2)   002666  000004                 TST6:   SCOPE
 (2)
5968   002670  012737  000020  001124         MOV     #BIT4,$GDDAT     ;LOAD EXPECTED DATA
```

```
5969   002676   104415                          CHKIT
5970   002700   104001                          ERROR   1                ;FAILED TO LOAD + READ EXT. START ENABLE
```

```
 5972
 5973                                    ;:****************************************************************
  (3)                                    ;*TEST 7        LOAD AND READ BACK GAIN SELECT 0
  (3)                                    ;:****************************************************************
  (2)   002702  000004                   TST7:   SCOPE
  (2)
 5974   002704  012737  000004  001124           MOV     #BIT2,$GDDAT    ;LOAD EXPECTED DATA
 5975   002712  104415                           CHKIT
 5976   002714  104001                           ERROR   1               ;FAILED TO LOAD + READ BACK GAIN SELECT 0
 5977                                    ;:****************************************************************
  (3)                                    ;*TEST 10       LOAD AND READ BACK GAIN SELECT 1
  (3)                                    ;:****************************************************************
  (2)   002716  000004                   TST10:  SCOPE
  (2)
 5978   002720  012737  000010  001124           MOV     #BIT3,$GDDAT    ;LOAD EXPECTED
 5979   002726  104415                           CHKIT
 5980   002730  104001                           ERROR   1               ;FAILED TO LOAD + READ BACK GAIN SELECT 1
 5981
 5982                                    ;:****************************************************************
  (3)                                    ;*TEST 11       LOAD AND READ BACK ERROR FLAG (BIT15)
  (3)                                    ;:****************************************************************
  (2)   002732  000004                   TST11:  SCOPE
  (2)
 5983   002734  012737  100000  001124           MOV     #BIT15,$GDDAT   ;LOAD EXPECTED DATA
 5984   002742  104415                           CHKIT
 5985   002744  104001                           ERROR   1               ;FAILED TO LOAD + READ BACK ERROR FLAG
 5986                                    ;:****************************************************************
  (3)                                    ;*TEST 12       TEST INIT CLEARS BITS 2-6,14
  (3)                                    ;:****************************************************************
  (2)   002746  000004                   TST12:  SCOPE
  (2)
  (1)   002750  012737  000300  001160           MOV     #300,$TIMES     ;;DO 300 ITERATIONS
 5987   002756  005037  001124                   CLR     $GDDAT          ;LOAD EXPECTED DATA
 5988   002762  012777  040174  176326           MOV     #40174,@STREG   ;SET STATUS REGISTER
 5989   002770  000005                           RESET                   ;INITIALIZE
 5990   002772  052777  000100  176144           BIS     #100,@$TKS      ;SET INTRPT. ENABLE
 5991   003000  017737  176312  001126           MOV     @STREG,$BDDAT   ;READ STATUS REGISTER
 5992   003006  001401                           BEQ     TST13           ;;NEXT TEST
 5993   003010  104001                           ERROR   1               ;RESET FAILED TO CLEAR AD ST. REG. BITS
 5994
 5995                                    ;:****************************************************************
  (3)                                    ;*TEST 13       TEST INIT CLEARS ERROR FLAG
  (3)                                    ;:****************************************************************
  (2)   003012  000004                   TST13:  SCOPE
  (2)
  (1)   003014  012737  000300  001160           MOV     #300,$TIMES     ;;DO 300 ITERATIONS
 5996   003022  012777  100000  176266           MOV     #BIT15,@STREG   ;SET BIT 15
 5997   003030  000005                           RESET                   ;ISSUE INIT
 5998   003032  052777  000100  176104           BIS     #100,@$TKS      ;SET INTRPT. EN. FOR KEYBOARD
 5999   003040  104414                           CHECK
 6000   003042  104001                           ERROR   1               ;BUS INIT FAILED TO CLEAR A/D DONE FLAG
 6001                                    ;:****************************************************************
  (3)                                    ;*TEST 14       TEST  DONE FLAG  SETS AND BIT0 CLEARS ON END OF CONV.
  (3)                                    ;:****************************************************************
  (2)   003044  000004                   TST14:  SCOPE
  (2)
```

```
 6002   003046  017700  176250              MOV     @ADBUFF,R0      ;READ DATA
 6003   003052  005277  176240              INC     @STREG          ;START CONVERSION
 6004   003056  012737  000200  001124      MOV     #BIT7,$GDDAT    ;LOAD EXPECTED
 6005   003064  004737  001426              JSR     PC,STALL        ;DELAY AN AMOUNT OF TIME
 6006   003070  042777  100000  176220      BIC     #BIT15,@STREG   ;MASK OUT ERROR BIT
 6007   003076  104414                      CHECK
 6008   003100  104001                      ERROR   1               ;A/D DONE FLAG FAILED TO SET
 6009                                                                ;  OR BITO FAILED TO CLEAR
 6010   003102  017700  176214              MOV     @ADBUFF,R0      ;CLEAR DONE FLAG FOR ITERATIONS
 6011
 6012                            ;;********************************************************************
  (3)                            ;*TEST 15        TEST INIT CLEARS DONE FLAG
  (3)                            ;;********************************************************************
  (2)   003106  000004          TST15:  SCOPE
  (2)
  (1)   003110  012737  000300  001160      MOV     #300,$TIMES     ;;DO 300 ITERATIONS
 6013   003116  005037  001124              CLR     $GDDAT          ;CLEAR EXPECTED
 6014   003122  005277  176170              INC     @STREG          ;START CONVERSION
 6015   003126  105777  176164      2$:     TSTB    @STREG
 6016   003132  100375                      BPL     2$
 6017   003134  000005                      RESET
 6C18   003136  104414                      CHECK
 6019   003140  104001                      ERROR   1               ;DONE FLAG FAILED TO CLEAR
 6020   003142  052777  000100  175774      BIS     #100,@$TKS      ;SET INTRPT. EN. BIT
 6021
 6022                            ;;********************************************************************
  (3)                            ;*TEST 16        TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
  (3)                            ;;********************************************************************
  (2)   003150  000004          TST16:  SCOPE
  (2)
 6023   003152  005277  176140              INC     @STREG          ;SET A/D START CONVERSION BIT
 6024   003156  105777  176134      1$:     TSTB    @STREG          ;WAIT FOR FLAG
 6025   003162  100375                      BPL     1$
 6026   003164  017700  176132              MOV     @ADBUFF,R0      ;READ CONVERTED VALUE
 6027   003170  104414                      CHECK
 6028   003172  104001                      ERROR   1               ;DONE FLAG FAILED TO CLEAR
```

B 3

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 64
CNAXAA.P11   23-DEC-82 14:21   T16   TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE   SEQ 0027

```
6030
6031                                    ;;****************************************************************
  (3)                                   ;*TEST 17        GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
  (3)                                   ;;****************************************************************
  (2)    003174   000004               TST17:  SCOPE
  (2)
6032                                    ;* "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
6033     003206   005046                     CLR     -(SP)           ;RESET PRIORITY
6034     003210   012746   003216            MOV     #3$,-(SP)
6035     003214   000002                     RTI
6036     003216   012777   003272   176104   3$:  MOV     #1$,@VECTOR     ;INTERRUPT VECTOR ADDRESS
6037     003224   012777   000200   176100        MOV     #200,@VECTR1    ;SET UP NEW PSW
6038     003232   012777   000101   176056        MOV     #BIT6!BIT0,@STREG ;SET INTERRUPT ENABLE BIT + START CONVERSION
6039     003240   105777   176052            2$:  TSTB    @STREG          ;WAIT FOR DONE
6040     003244   100375                          BPL     2$              ;FLAG TO SET
6041     003246   017737   176044   001126        MOV     @STREG,$BDDAT   ;READ STATUS REGISTER
6042     003254   012737   000300   001124        MOV     #BIT7!BIT6,$GDDAT ;GOOD DATA
6043     003262   104002                          ERROR   2               ;FAILED TO INTERRUPT ON DONE
6044     003264   004737   010164            JSR     PC,DUMC         ;TYPE COMPLETED
6045     003270   000414                     BR      TST20           ;;BRANCH TO NEXT TEST
6046     003272   022626                1$:  CMP     (SP)+,(SP)+     ;RESET STACK POINTER
6047     003274   012777   001404   176026        MOV     #UNEXP,@VECTOR  ;SET UP FOR UNEXPECTED INTERRUPT
6048     003302   005046                     CLR     -(SP)           ;CLEAR PSW
6049     003304   012746   003312            MOV     #4$,-(SP)
6050     003310   000002                     RTI
6051     003312   004737   010164        4$:  JSR     PC,DUMC         ;TYPE COMPLETED
6052     003316   005777   176000            TST     @ADBUFF         ;CLEAR DONE BIT
6053                                    ;;****************************************************************
  (3)                                   ;*TEST 20        TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET
  (3)                                   ;;****************************************************************
  (2)    003322   000004               TST20:  SCOPE
  (2)
6054                                    ;* "ENTERING TEST 20" TYPED OUT TO TELL YOU THE NEXT
  (1)                                   ;*TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
  (1)                                   ;*THERE IS DANGER THAT THE "Q BUSS" COULD GET 'HUNG' WHILE
  (1)                                   ;*EXECUTING TEST "20".
  (1)    003324   012700   000020            MOV     #20,R0          ;GET TEST NO.
  (1)    003330   004737   010112            JSR     PC,DUMW         ;PRINT MESSAGE
6055     003334   012777   003374   175772   MOV     #1$,@VECTR2     ;SETUP VECTOR ADDRESS
6056     003342   012777   140000   175746   MOV     #BIT15!BIT14,@STREG    ;CAUSE AN INTERRUPT
6057     003350   017737   175742   001126   MOV     @STREG,$BDDAT   ;BAD DATA
6058     003356   012737   140000   001124   MOV     #BIT15!BIT14,$GDDAT   ;GOOD DATA
6059     003364   104002                     ERROR   2
6060     003366   004737   010164            JSR     PC,DUMC         ;TYPE COMPLETED
6061     003372   000753                     BR      TST20
6062     003374   022626                1$:  CMP     (SP)+,(SP)+     ;POP STACK
6063     003376   004737   010164            JSR     PC,DUMC
6064     003402   005077   175710            CLR     @STREG
```

C 3

MAINDEC-11-CNAXA-A   MACY11 30(1046)   23-DEC-82 14:22 PAGE 65   SEQ 0028
CNAXAA.P11   23-DEC-82 14:21   T20   TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET

```
6066
6067          ;;****************************************************************
      (3)     ;*TEST 21          TEST ERROR FLAG SETS IF 2ND CONVERSION IS STARTED WHILE A/D DONE IS SET
      (3)     ;;****************************************************************
      (2)  003406  000004           TST21:  SCOPE
      (2)
6068  003410  012777  000001  175700         MOV    #BIT0,@STREG     ;START CONVERSION
6069  003416  105777  175674    1$:  TSTB   @STREG           ;WAIT FOR
6070  003422  100375                  BPL    1$
6071  003424  012737  100200  001124   MOV   #BIT15!BIT7,$GDDAT ;LOAD EXPECTED VALUE
6072  003432  012777  000001  175656   MOV   #BIT0,@STREG     ;START 2ND CONVERSION
6073  003440  104414                  CHECK
6074  003442  104001                  ERROR  1                ;ERROR FLAG NOT SET WHEN 2ND
6075                               ; CONVERSION WAS STARTED BEFORE READING BUFFER FROM FIRST
6076  003444  017700  175652         MOV    @ADBUFF,R0       ;CLEAR DONE FLAG
6077
6078          ;;****************************************************************
      (3)     ;*TEST 22          TEST CLOCK OVERFLOW STARTS A/D  (IF KWV11-C IS AVAILABLE)
      (3)     ;;****************************************************************
      (2)  003450  000004           TST22:  SCOPE
      (2)
6079  003452  005737  001374         TST    KWAD             ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
6080  003456  001424                  BEQ    TST23            ;;BR IF NO CLOCK THERE
6081  003460  012737  000240  001124   MOV   #BIT7!BIT5,$GDDAT      ;LOAD EXPECTED A/D STATUS
6082  003466  013777  001124  175622   MOV   $GDDAT,@STREG    ;ENABLE THE A/D STATUS REGISTER
6083  003474  012777  177776  175640   MOV   #177776,@KWBPR   ;LOAD KWV11-C CLOCK PRESET REGISTER
6084  003502  012777  000011  175630   MOV   #11,@KWCSR       ;START CLOCK
6085  003510  004737  001426         JSR    PC,STALL         ;DELAY FOR A CLOCK TICK
6086  003514  104414                  CHECK                   ;CHECK A/D STATUS AGAINST EXPECTED
6087  003516  104001                  ERROR  1                ;A/D DONE FAILED TO SET WITH CLOCK STARTS
6088  003520  005777  175576         TST    @ADBUFF          ;CLEAR A/D DONE
6089  003524  005077  175566         CLR    @STREG           ;CLEAR A/D CONTROL
6090
6091          ;;****************************************************************
      (3)     ;*TEST 23          TEST EXTERNAL TRIGGER STARTS A/D (IF KW11-C IS CONNECTED TO EXT START TA
      (3)     ;;****************************************************************
      (2)  003530  000004           TST23:  SCOPE
      (2)
6092  003532  005737  001376         TST    KWEX             ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
6093  003536  001424                  BEQ    TST24            ;;BR IF NO CLOCK THERE
6094  003540  012737  000220  001124   MOV   #BIT7!BIT4,$GDDAT      ;LOAD EXPECTED A/D STATUS
6095  003546  013777  001124  175542   MOV   $GDDAT,@STREG    ;ENABLE THE A/D STATUS REGISTER
6096  003554  012777  177776  175560   MOV   #177776,@KWBPR   ;LOAD KWV11-C CLOCK PRESET REGISTER
6097  003562  012777  000011  175550   MOV   #11,@KWCSR       ;START CLOCK
6098  003570  004737  001426         JSR    PC,STALL         ;DELAY FOR CLOCK TICKS
6099  003574  104414                  CHECK                   ;CHECK A/D STATUS AGAINST EXPECTED
6100  003576  104001                  ERROR  1                ;A/D DONE FAILED TO SET WITH EXTERNAL STARTS
6101  003600  005777  175516         TST    @ADBUFF          ;CLEAR A/D DONE
6102  003604  005077  175506         CLR    @STREG           ;CLEAR A/D CONTROL
6103
```

```
 6105
 6106                                      ;;********************************************************************
  (3)                                      ;*TEST 24        TEST EXTERNAL TRIGGER STARTS A/D (IF MANUAL TRIGGER IS CONNECTED TO EXT
  (3)                                      ;;********************************************************************
  (2)   003610  000004                     TST24:   SCOPE
  (2)
 6107   003612  005737  001400                      TST     MAEX             ;TEST IF OPERATOR SAID MANUAL TRIGGER IS CONNECTED
 6108   003616  001427                               BEQ     TST25            ;;BR IF NO EXT. TRIGGER AVAILABLE
 6109   003620  005737  001202                       TST     $PASS            ;TEST IF FIRST PASS OF PROGRAM
 6110   003624  001024                               BNE     TST25            ;;BR IF NOT FIRST PASS
 6111   003626  012737  000220  001124               MOV     #BIT7!BIT4,$GDDAT        ;LOAD EXPECTED A/D STATUS
 6112   003634  013777  001124  175454               MOV     $GDDAT,@STREG    ;ENABLE THE EXT START SIGNAL
 6113   003642  104401  012016                        TYPE    .MSGNEX          ;TELL OPERATOR TO GENERATE EXT. TRIGGER
 6114   003646  104401  011276                        TYPE    .CRWR            ;TELL OPERATOR ABOUT "RETURN"
 6115   003652  104412                               RDLIN
 6116   003654  012600                               MOV     (SP)+,R0         ;REMOVE ANSWER OFF OF THE STACK
 6117   003656  000240                               NOP
 6118   003660  000240                               NOP
 6119   003662  104414                               CHECK                    ;CHECK A/D STATUS AGAINST EXPECTED
 6120   003664  104001                               ERROR   1                ;A/D DONE FAILED TO SET WITH EXTERNAL START
 6121   003666  005777  175430                       TST     @ADBUFF          ;CLEAR A/D DONE
 6122   003672  005077  175420                       CLR     @STREG           ;CLEAR A/D CONTROL
 6123
 6124                                      ;;********************************************************************
  (3)                                      ;*TEST 25        TEST ERROR FLAG SETS IS START 2ND CONV. BEFORE DONE FLAG SETS (KWV11-C)
  (3)                                      ;;********************************************************************
  (2)   003676  000004                     TST25:   SCOPE
  (2)
 6125   003700  005737  001374                      TST     KWAD             ;TEST IF OPERATOR SAID KWV11-C WAS CONNECTED
 6126   003704  001436                               BEQ     TST26            ;;BR IF NO CLOCK PRESENT
 6127   003706  012737  100240  001124               MOV     #BIT15!BIT7!BIT5,$GDDAT  ;LOAD EXPECTED
 6128   003714  012777  177776  175420               MOV     #-2,@KWBPR       ;LOAD CLOCK PRESET
 6129   003722  012777  000040  175366               MOV     #BIT5,@STREG     ;ENABLE CLOCK START
 6130   003730  017700  175366                       MOV     @ADBUFF,R0       ;ENSURE CLEARED A/D DONE
 6131   003734  012777  000011  175376               MOV     #11,@KWCSR       ;START CLOCK
 6132   003742  105777  175372             1$:       TSTB    @KWCSR           ;WAIT FOR CLOCK READY
 6133   003746  100375                               BPL     1$
 6134   003750  152777  000001  175340               BISB    #BIT0,@STREG     ;CLOCK OVERFLOW SHOULD HAVE STARTED A/D
 6135                                                                         ;TRY TO START IT AGAIN AND GET AN ERROR
 6136   003756  017737  175334  001126               MOV     @STREG,$BDDAT    ;READ A/D STATUS
 6137   003764  023737  001124  001126               CMP     $GDDAT,$BDDAT    ;COMPARE TO EXPECTED
 6138   003772  001401                               BEQ     2$               ;;BR IF SAME
 6139   003774  104001                               ERROR   1                ;ERROR FLAG NOT SET WHEN 2ND CONVERT STARTED
 6140                                                                         ; WHILE FIRST IS IN PROGRESS
 6141   003776  017700  175320             2$:       MOV     @ADBUFF,R0       ;READ AND CLEAR A/D DONE
```

E 3

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 67
CNAXAA.P11   23-DEC-82 14:21   T25   TEST ERROR FLAG SETS IS START 2ND CONV. BEFORE DONE FLAG SETS (KWV11-C)   SEQ 0030

```
 6143
 6144                                ;;***********************************************************************
  (3)                                ;*TEST 26         TEST "B EVENT" STARTS A/D (IF JUMPER "F2" IS PRESENT)
  (3)                                ;;***********************************************************************
  (2)    004002  000004              TST26:  SCOPE
  (2)
 6145    004004  005737  001402              TST     BTEX              :TEST IF OPERATOR SAID "F2" IS INSTALLED
 6146    004010  001416                      BEQ     TST27             ;;BR IF NOT THERE
 6147    004012  012737  000220  001124      MOV     #BIT7!BIT4,$GDDAT         :LOAD EXPECTED A/D STATUS
 6148    004020  013777  001124  175270      MOV     $GDDAT,@STREG     :ENABLE THE A/D STATUS REGISTER
 6149    004026  004737  001426              JSR     PC,STALL          :DELAY AN AMOUNT OF TIME
 6150    004032  104414                      CHECK                     :CHECK A/D STATUS AGAINST EXPECTED
 6151    004034  104001                      ERROR   1                 :A/D DONE FAILED TO SET WITH "B EVENT"
 6152    004036  005077  175254              CLR     @STREG            :CLEAR A/D CONTROL
 6153    004042  005777  175254              TST     @ADBUFF           :CLEAR A/D DONE
 6154
 6155
 6156                                ;;***********************************************************************
  (3)                                ;*TEST 27         END OF ADV11-C LOGIC TESTS
  (3)                                ;;***********************************************************************
  (2)    004046  000004              TST27:  SCOPE
  (2)
 6157    004050  000207                      RTS     PC                :RETURN TO TEST SECTION
 6158
 6159
 6160                                        .SBTTL
 6161                                        .SBTTL  END OF LOGIC TESTS - SECTION
 6162
 6163
 6164                                ;;SUBROUTINE FOR LOGIC TESTS;;
 6165    004052  013777  001124  175236 TESTIT: MOV   $GDDAT,@STREG     :LOAD EXPECTED VALUE
 6166    004060  017737  175232  001126 TEST:   MOV   @STREG,$BDDAT     :READ ST. REG.
 6167    004066  023737  001124  001126      CMP     $GDDAT,$BDDAT     :COMPARE RESULTS
 6168    004074  001002                      BNE     RETERR            ;;ERROR RETURN
 6169    004076  062716  000002              ADD     #2,(SP)           :BUMP RETURN ADDRESS TO GET AROUND ERROR
 6170    004102  000002              RETERR: RTI
 6171
 6172                                        .SBTTL
 6173                                        .SBTTL  START OF ADV11-C ANALOG WRAPAROUND SECTION
 6174                                        .SBTTL
```

```
6176
6177    004104                                   WRAP:
 (4)                                             ;:******************************************************
 (3)                                             ;*TEST 30        SETUP TO RUN ANALOG WRAPAROUND TEST
 (3)                                             ;:******************************************************
 (2)    004104  012737  000030  001102           TST30:  MOV     #$TN,$TSTNM
 (2)
 (1)    004112  012737  000001  001160                   MOV     #1,$TIMES          ;:DO 1 ITERATION
6178                                             ;LOAD AXV11-C DAC TO MAX OUTPUT VOLTAGE
6179    004120  012777  007777  175176                   MOV     #7777,@DACA        ;LOAD DAC ''A''
6180    004126  012777  007777  175172                   MOV     #7777,@DACB        ;LOAD DAC ''B''
6181    004134  012737  004156  001110                   MOV     #1$,$LPERR         ;LOAD ERROR ADDRESS
6182    004142  012737  004156  001106                   MOV     #1$,$LPADR         ;LOAD LOOP ADDRESS
6183                                             ;DELAY SUFFICIENT TIME TO LET THE DAC'S SETTLE
6184    004150  012700  000002                           MOV     #2,R0              ;LOAD DELAY TIMER
6185    004154  005001                                   CLR     R1                 ;CLEAR DELAY COUNT
6186    004156  005301                           1$:     DEC     R1                 ;DELAY
6187    004160  001376                                   BNE     1$
6188    004162  005300                                   DEC     R0                 ;DELAY
6189    004164  001374                                   BNE     1$
6190
6191                                             ;:******************************************************
 (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
 (2)
 (1)    004170  012737  000001  001160           1$:     MOV     #1,$TIMES          ;:DO 1 ITERATION
6192    004176  005737  001366                           TST     TC1                ;TEST IF TEST FIXTURE IS INSTALLED
6193    004202  001440                                   BEQ     TST32              ;:BR IF NOT
        004204  004537  007710                           JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
6194    004210  000000                                   CHAN00                     ;CHANNEL 0
6195    004212  004537  010046                           JSR     R5,COMPAR          ;COMPARE RESULTS
6196    004216  007777                                   7777
6197    004220  001354                                   VWRAP
6198    004222  104004                                   ERROR   4                  ;ERROR AN A/D CHANNEL 0 - VALUE DID NOT
6199                                                                                ;    EQUAL EXPECTED VALUE
6200
6201    004224  004537  007710                           JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
        004230  000001                                   CHAN01                     ;CHANNEL 1
6202    004232  004537  010046                           JSR     R5,COMPAR          ;COMPARE RESULTS
6203    004236  006000                                   6000                       ;EXPECTED VALUE
6204    004240  001354                                   VWRAP                      ;USING A KNOWN SPREAD
6205    004242  104004                                   ERROR   4                  ;ERROR ON A/D CHANNEL 1 - VALUE DID NOT
6206                                                                                ;    EQUAL EXPECTED
6207
6208    004244  004537  007710                           JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
        004250  000002                                   CHAN02                     ;CHANNEL 2
6209    004252  004537  010046                           JSR     R5,COMPAR          ;COMPARE RESULTS
6210    004256  005000                                   5000                       ;AGAINST THIS VALUE FOR CHANNEL 2
6211    004260  001354                                   VWRAP                      ;USING A KNOWN SPREAD
6212    004262  104004                                   ERROR   4                  ;ERROR ON A/D CHANNEL 2 - VALUE DID NOT
6213                                                                                ;    EQUAL EXPECTED
6214
6215    004264  004537  007710                           JSR     R5,CONVRT          ;GET THE AVERAGE VALUE FOR
        004270  000003                                   CHAN03                     ;CHANNEL 03
6216    004272  004537  010046                           JSR     R5,COMPAR          ;COMPARE RESULTS
6217    004276  004400                                   4400                       ;AGAINST THIS VALUE FOR CHANNEL 3
6218    004300  001354                                   VWRAP                      ;USING A KNOWN SPREAD
6219    004302  104004                                   ERROR   4                  ;ERROR ON A/D CHANNEL 3 - VALUE DID NOT
6220
```

6221   ; EQUAL EXPECTED

```
 6223
 6224                                  ;;*****************************************************************
  (3)                                  ;*TEST 32       COMPARE CHANNEL 0 (F.S.) AGAINST OTHER F.S. CHANNELS (4 AND 10)
  (3)                                  ;;*****************************************************************
  (2)    004304  000004               TST32:  SCOPE
  (2)
  (1)    004306  012737  000001  001160        MOV     #1,$TIMES       ;;DO 1 ITERATION
 6225    004314  005737  001366               TST     TC1             ;TEST IF TEST FIXTURE IS INSTALLED
 6226    004320  001431                       BEQ     TST33           ;;BR IF NOT
 6227    004322  004537  007710               JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 6228    004326  000000                       CHAN00                  ;CHANNEL 0
 6229    004330  013737  001360  004356        MOV     TEMP,4$         ;SAVE CHANNEL 00 CONVERTED VALUE
 6230    004336  013737  001360  004376        MOV     TEMP,10$        ;
 6231
 6232    004344  004537  007710               JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 6233    004350  000004                       CHAN04                  ;CHANNEL 4
 6234    004352  004537  010046               JSR     R5,COMPAR       ;COMPARE RESULTS
 6235    004356  000000               4$:     0                       ;AGAINST THIS VALUE FOR CHANNEL 0
 6236    004360  010236                       V2                      ;USING A SPREAD OF 2 COUNTS
 6237    004362  104004                       ERROR   4               ;ERROR ON A/D CHANNEL 4 - VALUE DID NOT
 6238                                                                  ;  EQUAL VALUE OF CHANNEL 0
 6239
 6240    004364  004537  007710               JSR     R5,CONVRT       ;GET THE AVERAGE VALUE FOR
 6241    004370  000010                       CHAN10                  ;CHANNEL 10
 6242    004372  004537  010046               JSR     R5,COMPAR       ;COMPARE RESULTS
 6243    004376  000000               10$:    0                       ;AGAINST THIS VALUE FOR CHANNEL 0
 6244    004400  010236                       V2                      ;USING A SPREAD OF 2 COUNTS
 6245    004402  104004                       ERROR   4               ;ERROR ON A/D CHANNEL 10 - VALUE DID NOT
 6246                                                                  ;  EQUAL VALUE OF CHANNEL 0
```

I 3

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22   PAGE 70
CNAXAA.P11   23-DEC-82 14:21   T32   COMPARE CHANNEL 0 (F.S.) AGAINST OTHER F.S. CHANNELS (4 AND 10)   SEQ 0034

```
      6248
      6249
      6250                        ;;*********************************************************************
       (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
       (2)
       (1)   004406  012737  000001  001160      MOV     #1,$TIMES        ;;DO 1 ITERATION
      6251   004414  005737  001366              TST     TC1              ;TEST IF TEST FIXTURE IS INSTALLED
      6252   004420  001431                      BEQ     TST34            ;;BR IF NOT
      6253   004422  004537  007710              JSR     R5,CONVRT        ;GET THE AVERAGE VALUE FOR
      6254   004426  000001                      CHAN01                   ;CHANNEL 1
      6255   004430  013737  001360  004456      MOV     TEMP,4$          ;SAVE CHANNEL 1 CONVERTED VALUE
      6256   004436  013737  001360  004476      MOV     TEMP,10$         ;SAVE IT AGAIN
      6257
      6258   004444  004537  007710              JSR     R5,CONVRT        ;GET THE AVERAGE VALUE FOR
      6259   004450  000005                      CHAN05                   ;CHANNEL 5
      6260   004452  004537  010046              JSR     R5,COMPAR        ;COMPARE RESULTS
      6261   004456  000000              4$:     0                        ;AGAINST THIS VALUE FOR CHANNEL 1
      6262   004460  010236                      V2                       ;USING A SPREAD OF 2 COUNTS
      6263   004462  104004                      ERROR   4                ;ERROR ON A/D CHANNEL 5 - VALUE DID NOT
      6264                                                                ;  EQUAL VALUE OF CHANNEL 0
      6265
      6266   004464  004537  007710              JSR     R5,CONVRT        ;GET THE AVERAGE VALUE FOR
      6267   004470  000011                      CHAN11                   ;CHANNEL 11
      6268   004472  004537  010046              JSR     R5,COMPAR        ;COMPARE RESULTS
      6269   004476  000000              10$:    0                        ;AGAINST THIS VALUE FOR CHANNEL 1
      6270   004500  010236                      V2                       ;USING A SPREAD OF 2 COUNTS
      6271   004502  104004                      ERROR   4                ;ERROR ON A/D CHANNEL 11 - VALUE DID NOT
      6272                                                                ;  EQUAL VALUE OF CHANNEL 1
      6273
```

J 3

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 71
CNAXAA.P11   23-DEC-82 14:21   T33   COMPARE CHANNEL 1 (1/2 F.S.) AGAINST OTHER 1/2 F.S. CHANNELS (5 AND 11)   SEQ 0035

```
6275
6276                                    ;;*******************************************************************
  (3)                                   ;*TEST 34        COMPARE CHANNEL 2 (1/4 F.S.) AGAINST OTHER 1/4 F.S. CHANNELS (6 AND 12)
  (3)                                   ;;*******************************************************************
  (2)    004504  000004                 TST34:  SCOPE
  (2)
  (1)    004506  012737  000001  001160         MOV     #1,$TIMES         ;;DO 1 ITERATION
6277     004514  005737  001366                 TST     TC1               ;TEST IF TEST FIXTURE IS INSTALLED
6278     004520  001431                         BEQ     TST35             ;;BR IF NOT
6279     004522  004537  007710                 JSR     R5,CONVRT         ;GET THE AVERAGE VALUE FOR
6280     004526  000002                          CHAN02                  ;CHANNEL 2
6281     004530  013737  001360  004556         MOV     TEMP,4$           ;SAVE CHANNEL 2 CONVERTED VALUE
6282     004536  013737  001360  004576         MOV     TEMP,10$          ;SAVE IT AGAIN
6283
6284     004544  004537  007710                 JSR     R5,CONVRT         ;GET THE AVERAGE VALUE FOR
6285     004550  000006                          CHAN06                  ;CHANNEL 6
6286     004552  004537  010046                 JSR     R5,COMPAR         ;COMPARE RESULTS
6287     004556  000000                 4$:     0                        ;AGAINST THIS VALUE FOR CHANNEL 2D
6288     004560  010236                         V2                       ;USING A SPREAD OF 2 COUNTS
6289     004562  104004                         ERROR   4                ;ERROR ON A/D CHANNEL 6 - VALUE DID NOT
6290                                                                     ;  EQUAL VALUE OF CHANNEL 2
6291
6292     004564  004537  007710                 JSR     R5,CONVRT         ;GET THE AVERAGE VALUE FOR
6293     004570  000012                          CHAN12                  ;CHANNEL 12
6294     004572  004537  010046                 JSR     R5,COMPAR         ;COMPARE RESULTS
6295     004576  000000                 10$:    0                        ;AGAINST THIS VALUE FOR CHANNEL 2
6296     004600  010236                         V2                       ;USING A SPREAD OF 2 COUNTS
6297     004602  104004                         ERROR   4                ;ERROR ON A/D CHANNEL 12 - VALUE DID NOT
6298                                                                     ;  EQUAL VALUE OF CHANNEL 2
6299
6300                                    ;;*******************************************************************
  (3)                                   ;*TEST 35        COMPARE CHANNEL 3 (1/8 F.S.) AGAINST CHANNEL 7 (1/8 F.S.)
  (3)                                   ;;*******************************************************************
  (2)    004604  000004                 TST35:  SCOPE
  (2)
  (1)    004606  012737  000001  001160         MOV     #1,$TIMES         ;;DO 1 ITERATION
6301     004614  005737  001366                 TST     TC1               ;TEST IF TEST FIXTURE IS INSTALLED
6302     004620  001416                         BEQ     TST36             ;;BR IF NOT
6303     004622  004537  007710                 JSR     R5,CONVRT         ;GET THE AVERAGE VALUE FOR
6304     004626  000003                          CHAN03                  ;CHANNEL 3
6305     004630  013737  001360  004650         MOV     TEMP,4$           ;SAVE CHANNEL 3 CONVERTED VALUE
6306
6307     004636  004537  007710                 JSR     R5,CONVRT         ;GET THE AVERAGE VALUE FOR
6308     004642  000007                          CHAN07                  ;CHANNEL 7
6309     004644  004537  010046                 JSR     R5,COMPAR         ;COMPARE RESULTS
6310     004650  000000                 4$:     0                        ;AGAINST THIS VALUE FOR CHANNEL 3
6311     004652  010236                         V2                       ;USING A SPREAD OF 2 COUNTS
6312     004654  104004                         ERROR   4                ;ERROR ON A/D CHANNEL 7 - VALUE DID NOT
6313                                                                     ;  EQUAL VALUE OF CHANNEL 3
6314
```

```
6316
6317                                       ;;****************************************************************
 (3)                                       ;*TEST 36          RELATIVE GAIN TEST USING CHANNEL 3 (1/8 F.S.)
 (3)                                       ;;****************************************************************
 (2)      004656  000004                   TST36:  SCOPE
 (2)
 (1)      004660  012737  000001  001160           MOV     #1,$TIMES        ;;DO 1 ITERATION
6318      004666  005737  001366                    TST     TC1              ;TEST IF AXV11 OR ADV11 CONNECTOR INSTALLED
6319      004672  001454                            BEQ     TST37            ;;BR IF NO CONNECTOR
6320      004674  012737  000000  010044           MOV     #GAIN00,OTHER    ;SELECT GAIN OF 00
6321      004702  004537  007714                    JSR     R5,CONVTR        ;GET THE VALUE OF CHANNEL 03
6322      004706  000003                            CHAN03
6323      004710  004537  010046                    JSR     R5,COMPAR        ;TEST GAIN
6324      004714  004400                            4400             ;EXPECTED VALUE
6325      004716  001354                            VWRAP            ; USING KNOWN SPREAD
6326      004720  104004                            ERROR   4        ;GAIN SELECT OF 00 FAILED TO EQUAL EXPECTED VALUE
6327
6328      004722  012737  000004  010044           MOV     #GAIN01,OTHER    ;SELECT GAIN OF 01
6329      004730  004537  007714                    JSR     R5,CONVTR        ;GET THE VALUE OF CHANNEL 03
6330      004734  000003                            CHAN03
6331      004736  004537  010046                    JSR     R5,COMPAR        ;TEST GAIN 01
6332      004742  005000                            5000             ;EXPECTED VALUE
6333      004744  001354                            VWRAP            ; USING KNOWN SPREAD
6334      004746  104004                            ERROR   4        ;GAIN SELECT OF 01 FAILED TO INCREASE
6335                                               ;    CONVERTED VALUE CORRECTLY
6336      004750  012737  000010  010044           MOV     #GAIN10,OTHER    ;SET GAIN SELECT = 10
6337      004756  004537  007714                    JSR     R5,CONVTR        ;GET VALUE OF CHANNEL 03
6338      004762  000003                            CHAN03
6339      004764  004537  010046                    JSR     R5,COMPAR        ;TEST GAIN 10 VALUE AGAINST 01
6340      004770  006000                            6000             ;EXPECTED VALUE
6341      004772  001354                            VWRAP            ; USING KNOWN SPREAD
6342      004774  104004                            ERROR   4        ;GAIN SELECT OF 10 FAILED TO INCREASE
6343                                               ;    CONVERTED VALUE CORRECTLY
6344      004776  012737  000014  010044           MOV     #GAIN11,OTHER    ;SET GAIN SELECT = 11
6345      005004  004537  007714                    JSR     R5,CONVTR        ;GET VALUE OF CHANNEL 03
6346      005010  000003                            CHAN03
6347      005012  004537  010046                    JSR     R5,COMPAR        ;TEST GAIN 11 VALUE AGAINST 10
6348      005016  007777                            7777             ;EXPECTED VALUE
6349      005020  001354                            VWRAP            ; USING KNOWN SPREAD
6350      005022  104004                            ERROR   4        ;GAIN SELECT OF 11 FAILED TO INCREASE
6351                                               ;    CONVERTED VALUE CORRECTLY
6352
6353                                       ;;****************************************************************
 (3)                                       ;*TEST 37          IF ADV11-C VERIFY CH13 IS AT + F.S.
 (3)                                       ;;****************************************************************
 (2)      005024  000004                   TST37:  SCOPE
 (2)
 (1)      005026  012737  000001  001160           MOV     #1,$TIMES        ;;DO 1 ITERATION
6354      005034  012777  004000  174264           MOV     #4000,@DACB      ;SET DAC "B" TO MIDRANGE
6355      005042  005737  001372                    TST     ADV11C           ;TEST IF ADV11-C
6356      005046  001410                            BEQ     TST40            ;;BR IF NOT ADV11-C
6357      005050  004537  007710                    JSR     R5,CONVRT        ;GET THE CONVERTED VALUE FOR CH13
6358      005054  000013                            CHAN13
6359      005056  004537  010046                    JSR     R5,COMPAR        ;TEST CH13 AGAINST EXPECTED
6360      005062  007777                            7777             ;+ F.S.
6361      005064  010236                            V2
```

L 3

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 72-1
CNAXAA.P11   23-DEC-82 14:21   T37   IF ADV11-C VERIFY CH13 IS AT + F.S.   SEQ 0037

6362 005066 104004   ERROR   4   ;CH13 WAS NOT PULLED UP TO +F.S.

```
6364
6365                                          .SBTTL
6366                                          .SBTTL   END OF ADV11-C ANALOG WRAPAROUND SECTION
6367                                          .SBTTL
6368                                          .SBTTL   START OF AXV11-C ANALOG WRAPAROUND SECTION
6369                                          .SBTTL
6370
6371                                  ;;*************************************************************
 (3)                                  ;*TEST 40        AXV11-C ANALOG WRAPAROUND TEST (DAC ''A'' TO A/D CHAN 0)
 (3)                                  ;;*************************************************************
 (2)   005070  000004                 TST40:  SCOPE
 (2)
 (1)   005072  012737  000001  001160          MOV     #1,$TIMES           ;;DO 1 ITERATION
6372                                  ;AXV11-C DAC ''A'' CONNECTED TO AXV11-C A/D CHANNEL 0
6373                                  ;AXV11-C TEST FIXTURE IS REQUIRED
6374
6375   005100  005737  001366                 TST     TC1                 ;TEST IF AXV11-C TEST FIXTURE IS PRESENT
6376   005104  001445                         BEQ     TST41               ;;BR IF NO TEST FIXTURE
6377   005106  005737  001372                 TST     ADV11C              ;TEST IF THE MODULE IS A ADV11-C
6378   005112  001042                         BNE     TST41               ;;BR IF NO DAC'S PRESENT
6379   005114  012737  000000  005154          MOV     #0,2$              ;PRIME THE DAC OUTPUT VALUE
6380   005122  013777  005154  174174          MOV     2$,@DACA           ;PRIME THE DAC OUTPUT STAGE
6381   005130  012777  000000  174160          MOV     #0,@STREG          ;INITIILIZE THE A/D STATUS REG
6382   005136  017700  174160                 MOV     @ADBUFF,R0         ;READ A/D VALUE AND CLEAR A/D DONE FLAG
6383   005142  004537  007710         1$:     JSR     R5,CONVRT          ;GET THE VALUE OF CHANNEL 0
6384   005146  000000                         CHAN00
6385   005150  004537  010046                 JSR     R5,COMPAR          ;COMPARE AGAINST EXPECTED D/A VALUE
6386   005154  000000                 2$:     0                          ;EXPECTED
6387   005156  001354                         VWRAP                      ;SPREAD ALLOWED
6388   005160  000413                         BR      3$                 ;CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
6389   005162  062737  000010  005154          ADD     #10,2$             ;UPDATE THE D/A OUTPUT VALUE
6390   005170  013777  005154  174126          MOV     2$,@DACA           ;UPDATE THE D/A OUTPUT VOLTAGE
6391   005176  022737  010000  005154          CMP     #10000,2$          ;TEST IF LAST STEP
6392   005204  001356                         BNE     1$
6393   005206  000401                         BR      4$                 ;;BR TO NEXT TEST
6394   005210  104004                 3$:     ERROR   4                  ;CONVERTED A/D VALUE DID NOT EQUAL EXPECTED VALUE
6395   005212  012777  007777  174104 4$:     MOV     #7777,@DACA        ;LOAD DAC ''A'' TO +F.S.
6396
```

```
6398
6399
6400                                    ;:******************************************************************
  (3)                                   ;*TEST 41        AXV11-C ANALOG WRAPAROUND TEST (DAC 'B' TO A/D CHAN 13)
  (3)                                   ;:******************************************************************
  (2)    005220  000004                 TST41:  SCOPE
  (2)
  (1)    005222  012737  000001  001160         MOV     #1,$TIMES            ;:DO 1 ITERATION
6401                                    ;AXV11-C DAC 'B' CONNECTED TO AXV11-C A/D CHANNEL 13
6402                                    ;AXV11-C TEST CABLE IS REQUIRED
6403
6404    005230  005737  001366                  TST     TC1                  ;TEST IF AXV11-C TEST FIXTURE IS PRESENT
6405    005234  001445                          BEQ     TST42                ;:BR IF NO TEST FIXTURE
6406    005236  005737  001372                  TST     ADV11C               ;TEST IF MODULE IS AN ADV11-C
6407    005242  001042                          BNE     TST42                ;:BR IF NO DAC'A PRESENT
6408    005244  012737  000000  005304          MOV     #0,2$                ;PRIME THE DAC OUTPUT VALUE
6409    005252  013777  005304  174046          MOV     2$,@DACB             ;PRIME THE DAC OUTPUT STAGE
6410    005260  012777  000000  174030          MOV     #0,@STREG            ;INITIILIZE THE A/D STATUS REG
6411    005266  017700  174030                  MOV     @ADBUFF,R0           ;READ A/D VALUE AND CLEAR A/D DONE FLAG
6412    005272  004537  007710         1$:      JSR     R5,CONVRT            ;GET THE VALUE OF CHANNEL 13
6413    005276  000013                          CHAN13
6414    005300  004537  010046                  JSR     R5,COMPAR            ;COMPARE AGAINST EXPECTED D/A VALUE
6415    005304  000000                 2$:      0                           ;EXPECTED
6416    005306  001354                          VWRAP                       ;SPREAD ALLOWED
6417    005310  000413                          BR      3$                   ;CONVERTED VALUE DID NOT EQUAL EXPECTED D/A VALUE
6418    005312  062737  000010  005304          ADD     #10,2$               ;UPDATE THE D/A OUTPUT VALUE
6419    005320  013777  005304  174000          MOV     2$,@DACB             ;UPDATE THE D/A OUTPUT VOLTAGE
6420    005326  022737  010000  005304          CMP     #10000,2$            ;TEST IF LAST STEP
6421    005334  001356                          BNE     1$
6422    005336  000401                          BR      4$                   ;:BR TO NEXT TEST
6423    005340  104004                 3$:      ERROR   4                    ;CONVERTED D/A VALUE DID NOT EQUAL EXPECTED
6424    005342  012777  007777  173756 4$:      MOV     #7777,@DACB          ;SET DAC 'B' TO + F.S.
6425
6426                                            .SBTTL
6427                                            .SBTTL  END OF AXV11-C ANALOG WRAPAROUND SECTION
```

B 4

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 75
CNAXAA.P11   23-DEC-82 14:21   END OF AXV11-C ANALOG WRAPAROUND SECTION

SEQ 0040

```
6429
6430
6431                                    .SBTTL
6432                                    .SBTTL   START OF AXV11-C/ADV11-C NON-WRAPAROUND ANALOG SECTION
6433                                    .SBTTL
6434
6435                          ;:*********************************************************
  (3)                         ;*TEST 42         VERIFY CH14, 15, 16 AND 17 ARE AT +-0 F.S.
  (3)                         ;:*********************************************************
  (2)    005350  000004       TST42:  SCOPE
  (2)
  (1)    005352  012737  000001  001160    MOV   #1,$TIMES           ;;DO 1 ITERATION
6436                         ;AAV11-C TEST CONNECTOR IS NOT REQUIRED (IN FACT WILL ERROR IF PRESENT)
6437
6438    005360  005737  001370          TST   TC2               ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
6439    005364  001045                  BNE   TST43             ;;BR IF TEST CONNECTOR
6440    005366  012777  000000  173722  MOV   #0,@STREG         ;INITIILIZE THE A/D STATUS REG
6441    005374  017700  173722          MOV   @ADBUFF,RO        ;READ A/D VALUE AND CLEAR A/D DONE FLAG
6442    005400  004537  007710          JSR   R5,CONVRT         ;GET THE VALUE OF CHANNEL 14
6443    005404  000014                  CHAN14
6444    005406  004537  010046          JSR   R5,COMPAR         ;COMPARE AGAINST EXPECTED VALUE
6445    005412  004000                  4000              ;EXPECTED
6446    005414  010236                  V2                ;SPREAD ALLOWED
6447    005416  104004                  ERROR  4           ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE
6448
6449    005420  004537  007710          JSR   R5,CONVRT         ;GET THE VALUE OF CHANNEL 15
6450    005424  000015                  CHAN15
6451    005426  004537  010046          JSR   R5,COMPAR         ;COMPARE AGAINST EXPECTED VALUE
6452    005432  004000                  4000
6453    005434  010236                  V2                ;SPREAD ALLOWED
6454    005436  104004                  ERROR  4           ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE
6455
6456    005440  004537  007710          JSR   R5,CONVRT         ;GET THE VALUE OF CHANNEL 16
6457    005444  000016                  CHAN16
6458    005446  004537  010046          JSR   R5,COMPAR         ;COMPARE AGAINST EXPECTED VALUE
6459    005452  004000                  4000
6460    005454  010236                  V2                ;SPREAD ALLOWED
6461    005456  104004                  ERROR  4           ;CONVERTED VALUE DID NOT EQUAL EXPECTED VALUE
6462
6463    005460  004537  007710          JSR   R5,CONVRT         ;GET THE VALUE OF CHANNEL 17
6464    005464  000017                  CHAN17
6465    005466  004537  010046          JSR   R5,COMPAR         ;COMPARE AGAINST EXPECTED VALUE
6466    005472  004000                  4000
6467    005474  010236                  V2                ;SPREAD ALLOWED
6468    005476  104004                  ERROR  4           ;CONVERTED VLAUE DID NOT EQUAL EXPECTED VALUE
6469
```

C 4

MAINDEC-11-CNAXA-A   MACY11 30(1046)   23-DEC-82 14:22 PAGE 76
CNAXAA.P11   23-DEC-82 14:21   T42   VERIFY CH14, 15, 16 AND 17 ARE AT +-0 F.S.

SEQ 0041

```
6471
6472
6473                                        .SBTTL
6474                                        .SBTTL   START OF AAV11-C TO AXV11-C ANALOG WRAPAROUND SECTION
6475                                        .SBTTL
6476
6477                               ;;***************************************************************
       (3)                         ;*TEST 43        AAV11-C ANALOG WRAPAROUND TEST (DAC ''A'' TO A/D CHAN 14)
       (3)                         ;;***************************************************************
       (2)   005500  000004        TST43:  SCOPE
       (2)
       (1)   005502  012737  000001  001160       MOV     #1,$TIMES        ;;DO 1 ITERATION
6478                                        ;AAV11-C TEST CONNECTOR IS REQUIRED
6479
6480   005510  005737  001370                TST     TC2              ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
6481   005514  001452                         BEQ     TST44            ;;BR IF NO TEST CONNECTOR
6482   005516  012737  000000  005562         MOV     #0,2$            ;PRIME THE DAC OUTPUT VALUE
6483   005524  012777  007777  173612         MOV     #7777,@DACO      ;PRIME THE DAC OUTPUT STAGE
6484   005532  012777  000000  173556         MOV     #0,@STREG        ;INITIILIZE THE A/D STATUS REG
6485   005540  017700  173556                 MOV     @ADBUFF,R0       ;READ A/D VALUE AND CLEAR A/D DONE FLAG
6486   005544  000240                          NOP
6487   005546  000240                          NOP
6488
6489   005550  004537  007710         1$:     JSR     R5,CONVRT        ;GET THE VALUE OF CHANNEL 14
6490   005554  000014                          CHAN14
6491   005556  004537  010046                 JSR     R5,COMPAR        ;COMPARE AGAINST EXPECTED D/A VALUE
6492   005562  000000         2$:     0
6493   005564  001354                          VWRAP                   ;SPREAD ALLOWED
6494   005566  000424                          BR      10$              ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
6495   005570  062737  000010  005562         ADD     #10,2$           ;UPDATE THE D/A OUTPUT VALUE
6496   005576  013737  005562  005636         MOV     2$,7$            ;COPY VALUE
6497   005604  005137  005636                 COM     7$               ;INVERT DATA
6498   005610  042737  170000  005636         BIC     #170000,7$       ;REMOVE EXTRA BITS
6499   005616  013777  005636  173520         MOV     7$,@DACO         ;UPDATE THE D/A OUTPUT VOLTAGE
6500   005624  022737  010000  005562         CMP     #10000,2$        ;TEST IF LAST STEP
6501   005632  001346                          BNE     1$
6502   005634  000402                          BR      TST44            ;;BR TO NEXT TEST
6503   005636  000000         7$:     0
6504   005640  104004         10$:    ERROR   4                        ;CONVERTED D/A VALUE DID NOT EQUAL EXPECTED
6505
```

D 4

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 77
CNAXAA.P11   23-DEC-82 14:21   T43   AAV11-C ANALOG WRAPAROUND TEST (DAC ''A'' TO A/D CHAN 14)   SEQ 0042

```
 6507
 6508
 6509                              ;;********************************************************************
  (3)                             ;*TEST 44        AAV11-C ANALOG WRAPAROUND TEST (DAC 'B' TO A/D CHAN 15)
  (3)                             ;;********************************************************************
  (2)   005642  000004           TST44:  SCOPE
  (2)
  (1)   005644  012737  000001  001160    MOV     #1,$TIMES           ;;DO 1 ITERATION
 6510                              ;AAV11-C TEST CONNECTOR IS REQUIRED
 6511
 6512   005652  005737  001370            TST     TC2                 ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
 6513   005656  001450                    BEQ     TST45               ;;BR IF NO TEST CONNECTOR
 6514   005660  012737  000000  005720    MOV     #0,2$               ;PRIME THE DAC OUTPUT VALUE
 6515   005666  012777  007777  173452    MOV     #7777,@DAC1         ;PRIME THE DAC OUTPUT STAGE
 6516   005674  012777  000000  173414    MOV     #0,@STREG           ;INITIILIZE THE A/D STATUS REG
 6517   005702  017700  173414            MOV     @ADBUFF,R0          ;READ A/D VALUE AND CLEAR A/D DONE FLAG
 6518
 6519   005706  004537  007710    1$:     JSR     R5,CONVRT           ;GET THE VALUE OF CHANNEL 15
 6520   005712  000015            CHAN15
 6521   005714  004537  010046            JSR     R5,COMPAR           ;COMPARE AGAINST EXPECTED D/A VALUE
 6522   005720  000000    2$:     0
 6523   005722  001354            VWRAP                               ;SPREAD ALLOWED
 6524   005724  000424            BR      10$                 ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
 6525   005726  062737  000010  005720    ADD     #10,2$              ;UPDATE THE D/A OUTPUT VALUE
 6526   005734  013737  005720  005774    MOV     2$,7$               ;COPY VALUE
 6527   005742  005137  005774            COM     7$                  ;INVERT DATA
 6528   005746  042737  170000  005774    BIC     #170000,7$          ;REMOVE EXTRA BITS
 6529   005754  013777  005774  173364    MOV     7$,@DAC1            ;UPDATE THE D/A OUTPUT VOLTAGE
 6530   005762  022737  010000  005720    CMP     #10000,2$           ;TEST IF LAST STEP
 6531   005770  001346                    BNE     1$
 6532   005772  000402                    BR      TST45               ;;BR TO NEXT TEST
 6533   005774  000000    7$:     0
 6534   005776  104004    10$:    ERROR   4                   ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
 6535
```

```
6537
6538
6539                              ;;****************************************************************
  (3)                             ;*TEST 45        AAV11-C ANALOG WRAPAROUND TEST (DAC "C" TO A/D CHAN 16)
  (3)                             ;;****************************************************************
  (2)   006000 000004             TST45:  SCOPE
  (2)
  (1)   006002 012737 000001 001160       MOV     #1,$TIMES           ;;DO 1 ITERATION
6540                               ;AAV11-C TEST CONNECTOR IS REQUIRED
6541
6542   006010 005737 001370              TST     TC2                 ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
6543   006014 001450                     BEQ     TST46               ;;BR IF NO TEST CONNECTOR
6544   006016 012737 000000 006056       MOV     #0,2$               ;PRIME THE DAC OUTPUT VALUE
6545   006024 012777 007777 173316       MOV     #7777,@DAC2         ;PRIME THE DAC OUTPUT STAGE
6546   006032 012777 000000 173256       MOV     #0,@STREG           ;INITIILIZE THE A/D STATUS REG
6547   006040 017700 173256              MOV     @ADBUFF,R0          ;READ A/D VALUE AND CLEAR A/D DONE FLAG
6548
6549   006044 004537 007710       1$:    JSR     R5,CONVRT           ;GET THE VALUE OF CHANNEL 16
6550   006050 000016                     CHAN16
6551   006052 004537 010046              JSR     R5,COMPAR           ;COMPARE AGAINST EXPECTED D/A VALUE
6552   006056 000000              2$:    0
6553   006060 001354                     VWRAP                       ;SPREAD ALLOWED
6554   006062 000424                     BR      10$                 ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
6555   006064 062737 000010 006056       ADD     #10,2$              ;UPDATE THE D/A OUTPUT VALUE
6556   006072 013737 006056 006132       MOV     2$,7$               ;COPY VALUE
6557   006100 005137 006132              COM     7$                  ;INVERT DATA
6558   006104 042737 170000 006132       BIC     #170000,7$          ;REMOVE EXTRA BITS
6559   006112 013777 006132 173230       MOV     7$,@DAC2            ;UPDATE THE D/A OUTPUT VOLTAGE
6560   006120 022737 010000 006056       CMP     #10000,2$           ;TEST IF LAST STEP
6561   006126 001346                     BNE     1$
6562   006130 000402                     BR      TST46               ;;BR TO NEXT TEST
6563   006132 000000              7$:    0
6564   006134 104004              10$:   ERROR   4                   ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
6565
```

F 4

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 79
CNAXAA.P11   23-DEC-82 14:21   T45   AAV11-C ANALOG WRAPAROUND TEST (DAC ''C'' TO A/D CHAN 16)   SEQ 0044

```
 6567
 6568
 6569                              ;;*************************************************************
  (3)                             ;*TEST 46         AAV11-C ANALOG WRAPAROUND TEST (DAC 'D' TO A/D CHAN 17)
  (3)                             ;;*************************************************************
  (2)   006136  000004           TST46:  SCOPE
  (2)
  (1)   006140  012737  000001  001160     MOV     #1,$TIMES              ;;DO 1 ITERATION
 6570                             ;AAV11-C TEST CONNECTOR IS REQUIRED
 6571   006146  005737  001370            TST     TC2                    ;TEST IF AAV11-C TEST CONNECTOR IS PRESENT
 6572   006152  001450                    BEQ     TST47                  ;;BR IF NO TEST CONNECTOR
 6573   006154  012737  000000  006214    MOV     #0,2$                  ;PRIME THE DAC OUTPUT VALUE
 6574   006162  012777  007777  173162    MOV     #7777,@DAC3            ;PRIME THE DAC OUTPUT STAGE
 6575   006170  012777  000000  173120    MOV     #0,@STREG              ;INITIILIZE THE A/D STATUS REG
 6576   006176  017700  173120            MOV     @ADBUFF,R0             ;READ A/D VALUE AND CLEAR A/D DONE FLAG
 6577
 6578   006202  004537  007710    1$:     JSR     R5,CONVRT              ;GET THE VALUE OF CHANNEL 17
 6579   006206  000017                    CHAN17
 6580   006210  004537  010046            JSR     R5,COMPAR              ;COMPARE AGAINST EXPECTED D/A VALUE
 6581   006214  000000    2$:     0
 6582   006216  001354                    VWRAP                          ;SPREAD ALLOWED
 6583   006220  000424                    BR      10$                    ;CONVERTED VLAUE DID NOT EQUAL EXPECTED D/A VALUE
 6584   006222  062737  000010  006214    ADD     #10,2$                 ;UPDATE THE D/A OUTPUT VALUE
 6585   006230  013737  006214  006270    MOV     2$,7$                  ;COPY DATA
 6586   006236  005137  006270            COM     7$                     ;INVERT DATA
 6587   006242  042737  170000  006270    BIC     #170000,7$             ;REMOVE EXTRA BITS
 6588   006250  013777  006270  173074    MOV     7$,@DAC3               ;UPDATE THE D/A OUTPUT VOLTAGE
 6589   006256  022737  010000  006214    CMP     #10000,2$              ;TEST IF LAST STEP
 6590   006264  001346                    BNE     1$
 6591   006266  000402                    BR      TST47                  ;;BR TO NEXT TEST
 6592   006270  000000    7$:     0
 6593   006272  104004   10$:     ERROR   4                      ;CONVERTED D/A VALUE NOT EQUAL TO EXPECTED
 6594                             ;;*************************************************************
  (3)                             ;*TEST 47        END OF AAV11-C TO AXV11-C ANALOG WRAPAROUND
  (3)                             ;;*************************************************************
  (2)   006274  000004           TST47:  SCOPE
  (2)
  (1)   006276  012737  000001  001160     MOV     #1,$TIMES              ;;DO 1 ITERATION
 6595   006304  000207                    RTS     PC                     ;EXIT AND RETURN TO CALLING ROUTINE
 6603
```

```
6605
6606                                          .SBTTL  I/O SUB-SECTION "1"     REPORT THE CONVERTED A/D VALUES
6607
6608   006306  005077  173004       IOTST1: CLR     @STREG                  ;CLEAR STATUS REGISTER
6609   006312  104401  010376               TYPE    .MSIO1                  ;TYPE OUT HEADING
6610   006316  005046                        CLR     -(SP)                   ;CLEAR PSW
6611   006320  012746  006326               MOV     #77$,-(SP)
6612   006324  000002                        RTI
6613   006326  104401  011122       77$:    TYPE    .CCHAN                  ;ASK OPERATOR FOR CHANNEL
6614   006332  104413                        RDOCT
6615   006334  012637  006422               MOV     (SP)+,10$               ;GET ANSWER
6616   006340  042737  177760  006422        BIC     #177760,10$             ;REMOVE EXTRA BITS
6617   006346  104401  011162               TYPE    .GCHAN                  ;ASK OPERATOR FOR GAIN
6618   006352  104413                        RDOCT
6619   006354  012637  010044               MOV     (SP)+,OTHER             ;GET ANSWER
6620   006360  006137  010044               ROL     OTHER                   ;MOVE TO BITS
6621   006364  006137  010044               ROL     OTHER                   ;2 + 3
6622   006370  042737  177763  010044        BIC     #177763,OTHER           ;REMOVE ANY UNWANTED BITS
6623   006376  104401  011067       1$:     TYPE    .CH
6624   006402  013746  006422               MOV     10$,-(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
6625   006412  012702  000010       2$:     MOV     #10,R2                  ;TYPEOUT COUNTER
6626   006416  004537  007714       3$:     JSR     R5,CONVTR               ;GET AN AVERAGED VALUE FOR THIS CHANNEL
6627   006422  000000              10$:     0
6628   006424  104401  011072       4$:     TYPE    .SPACE
6629   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
6630   006440  012701  010000               MOV     #10000,R1
6631   006444  005301              5$:     DEC     R1
6632   006446  001376                        BNE     5$
6633   006450  005302                        DEC     R2                      ;DECREMENT THE COUNTER
6634   006452  001361                        BNE     3$                      ;NO CARRIAGE RETURN
6635   006454  104401  001171               TYPE    .$CRLF                  ;CARRIAGE RETURN
6636   006460  000746                        BR      1$                      ;REPEAT CONVERSION
```

```
6638
6639                                              .SBTTL  I/O SUB-SECTION ''2''    SCANNING CHANNELS AND GAIN SELECT - SECTION
6640
6641    006462  104401  010454        IOTST2: TYPE    ,MSIO2          ;TELL OPERATOR THE SECTION NAME
6642
6643    006466  005002                        CLR     R2              ;INITIALIZE THE CHANNEL SCANNER
6644    006470  005003                        CLR     R3              ;INITIALIZE THE GAIN SELECT VALUE
6645
6646    006472  104401  001171        1$:     TYPE    ,$CRLF          ;MAKE A FRESH OUTPUT LINE
6647    006476  012704  000007                MOV     #7,R4           ;LOAD LINE WIDTH COUNTER
6648
6649    006502  104401  011067                TYPE    ,CH             ;SHOW ''CH'' TEXT
6650
6651    006506  010246                        MOV     R2,-(SP)        ;LOAD THE CHANNEL CODE
6652    006510  104403                        TYPOS
6653    006512     002     001                .BYTE   2,1
6654
6655    006514  104401  011114                TYPE    ,ADOT           ;SEPERATE CH FROM GS
6656
6657    006520  112737  000060  011116        MOVB    #'0,AZERO       ;LOAD ASCII 0
6658    006526  132703  000010                BITB    #10,R3          ;TEST IF GS1 = 1
6659    006532  001402                        BEQ     2$              ;BR IF NOT SET
6660    006534  105237  011116                INCB    AZERO           ;MAKE IT A ONE
6661    006540  104401  011116        2$:     TYPE    ,AZERO          ;REPORT GS1 STATUS
6662
6663    006544  112737  000060  011116        MOVB    #'0,AZERO       ;LOAD ASCII 0
6664    006552  132703  000004                BITB    #4,R3           ;TEST IF GS0 = 1
6665    006556  001402                        BEQ     3$              ;BR IF NOT SET
6666    006560  105237  011116                INCB    AZERO           ;MAKE IT A ONE
6667    006564  104401  011116        3$:     TYPE    ,AZERO          ;REPORT GS0 STATUS
6668
6669    006570  010200                        MOV     R2,R0           ;GET CURRENT CHANNEL VALUE
6670    006572  000300                        SWAB    R0              ;MOVE TO MUX POSITION
6671    006574  050300                        BIS     R3,R0           ;ADD THE GAIN SELECT BITS
6672    006576  010077  172514                MOV     R0,@STREG       ;SELECT MUX AND GAIN BITS
6673    006602  105277  172510        4$:     INCB    @STREG          ;START CONVERSION
6674    006606  105777  172504        5$:     TSTB    @STREG          ;WAIT FOR A/D DONE
6675    006612  100375                        BPL     5$
6676
6677    006614  104401  011072                TYPE    ,SPACE          ;ENSURE SOME OUTPUT ROOM
6678    006620  017746  172476                MOV     @ADBUFF,-(SP)   ;READ CONVERTED VALUE AND SAVE FOR TYPOUT
6679    006624  104403                        TYPOS
6680    006626     004     001                .BYTE   4,1
6681
6682    006630  105304                        DECB    R4              ;FINISHED A LINE ACROSS THE PAGE
6683    006632  001363                        BNE     4$              ;BR AND CONVERT WITH CURRENT GAIN AND CHANNEL
6684
6685    006634  005202                        INC     R2              ;BUMP CHANNEL VALUE
6686    006636  062703  000004                ADD     #4,R3           ;BUMP GAIN SELECT VALUE
6687    006642  042703  177763                BIC     #177763,R3      ;REMOVE EXTRA BITS
6688    006646  122702  000020                CMPB    #20,R2          ;TEST IS LAST CHANNEL
6689    006652  001307                        BNE     1$              ;BR IF NOT
6690    006654  005002                        CLR     R2              ;INITIALIZE THE CHANNEL
6691    006656  104401  001171                TYPE    ,$CRLF          ;INSERT ANOTHER FRESH OUTPUT LINE
6692    006662  000703                        BR      1$              ;AND DO IT OVER AND OVER AND OVER AGAIN
6693
```

```
6695
6696                                              .SBTTL  I/O SUB-SECTION ''3''      AXV11-C A/D INPUT ECHO TO AXV11-C D/A OUTPUT
6697
6698   006664  104401  010514        IOTST3: TYPE    ,MSIO3          ;TELL OPERATOR THE NAME
6699   006670  104401  011122                TYPE    ,CCHAN          ;ASK OPER. FOR THE CHANNEL
6700   006674  104413                        RDOCT
6701   006676  012637  006744                MOV     (SP)+,10$
6702   006702  042737  177760  006744        BIC     #177760,10$     ;REMOVE EXTRA BITS
6703   006710  104401  011162                TYPE    ,GCHAN          ;ASK OPER FOR THE GAIN SELECT VALUE
6704   006714  104413                        RDOCT
6705   006716  012637  010044                MOV     (SP)+,OTHER     ;GET THE ANSWER
6706   006722  006337  010044                ASL     OTHER           ;MOVE INTO
6707   006726  006337  010044                ASL     OTHER           ;GAIN SELECT POSITION
6708   006732  042737  177763  010044        BIC     #177763,OTHER   ;REMOVE EXTRA BITS
6709
6710   006740  004537  007714        4$:     JSR     R5,CONVTR       ;CONVERT SELECTED CHANNEL AND GAIN
6711   006744  000000        10$:    0
6712
6713   006746  042737  170000  001360        BIC     #170000,TEMP    ;REMOVE EXTRA BITS
6714   006754  013777  001360  172342         MOV     TEMP,@DACA      ;LOAD DAC ''A''
6715   006762  013777  001360  172336         MOV     TEMP,@DACB      ;LOAD DAC ''B''
6716
6717   006770  000763                        BR      4$              ;LOOP BACK AND REPEAT
6718
6719                                              .SBTTL  I/O SUB-SECTION ''4''      AXV11-C D/A RAMPS
6720
6721   006772  104401  010557        IOTST4: TYPE    ,MSIO4          ;TELL OPERATOR THE NAME
6722   006776  012703  000000                MOV     #0,R3           ;LOAD DAC - F.S. VALUE
6723   007002  012704  007777                MOV     #7777,R4        ;LOAD DAC + F.S. VALUE
6724
6725   007006  012705  010000        1$:     MOV     #BIT12,R5       ;LOAD LOOP COUNT
6726   007012  010377  172306        2$:     MOV     R3,@DACA        ;LOAD DAC ''A''
6727   007016  010477  172304                MOV     R4,@DACB        ;LOAD DAC ''B''
6728   007022  005305                        DEC     R5              ;FINISHED ALL BITS ?
6729   007024  001403                        BEQ     3$              ;BR IF DONE
6730   007026  005304                        DEC     R4              ;LOWER DAC ''B'' VALUE
6731   007030  005203                        INC     R3              ;RAISE DAC ''A'' VALUE
6732   007032  000767                        BR      2$              ;DO NEXT COUNT
6733
6734   007034  012705  010000        3$:     MOV     #BIT12,R5       ;LOAD LOOP COUNT
6735   007040  010377  172260        4$:     MOV     R3,@DACA        ;LOAD DAC ''A''
6736   007044  010477  172256                MOV     R4,@DACB        ;LOAD DAC ''B''
6737   007050  005305                        DEC     R5              ;FINISHED ALL BITS ?
6738   007052  001755                        BEQ     1$
6739   007054  005303                        DEC     R3              ;LOWER DAC ''A'' VALUE
6740   007056  005204                        INC     R4              ;RAISE DAC ''B'' VALUE
6741   007060  000767                        BR      4$              ;DO NEXT COUNT
```

```
6743
6744                                     .SBTTL  I/O SUB-SECTION ''5''     AXV11-C D/A CALIBRATION
6745
6746   007062  104401  010632   IOTST5: TYPE    ,MSIO5          ;TELL OPERATOR THE NAME
6747   007066  012703  000000           MOV     #0,R3           ;LOAD DAC - F.S. VALUE
6748   007072  012704  007777           MOV     #7777,R4        ;LOAD DAC + F.S. VALUE
6749   007076  012705  004000           MOV     #4000,R5        ;LOAD 0.0 F.S. VALUE
6750
6751   007102  010377  172216   1$:     MOV     R3,@DACA        ;LOAD DAC ''A'' TO - F.S.
6752   007106  010377  172214           MOV     R3,@DACB        ;LOAD DAC 'B'' TO - F.S.
6753   007112  104412                   RDLIN
6754   007114  012600                   MOV     (SP)+,R0        ;REMOVE CHARACTER
6755   007116  010477  172202           MOV     R4,@DACA        ;LOAD DAC ''A'' TO + F.S.
6756   007122  010477  172200           MOV     R4,@DACB        ;LOAD DAC 'B'' TO + F.S.
6757   007126  104412                   RDLIN
6758   007130  012600                   MOV     (SP)+,R0        ;REMOVE CHARACTER
6759   007132  010577  172166           MOV     R5,@DACA        ;LOAD DAC ''A'' TO MID POINT
6760   007136  010577  172164           MOV     R5,@DACB        ;LOAD DAC 'B'' TO MID POINT
6761   007142  104412                   RDLIN
6762   007144  012600                   MOV     (SP)+,R0        ;REMOVE CHARACTER
6763   007146  000755                   BR      1$
6764
6765                                     .SBTTL  I/O SUB-SECTION ''6''     AXV11-C D/A SQUARE WAVE
6766
6767   007150  104401  010677   IOTST6: TYPE    ,MSIO6          ;TELL OPERATOR THE NAME
6768   007154  012703  000000           MOV     #0,R3           ;LOAD DAC - F.S.
6769   007160  012704  007777           MOV     #7777,R4        ;LOAD DAC + F.S.
6770
6771   007164  010377  172134   1$:     MOV     R3,@DACA        ;LOAD DAC ''A'' TO MIN LEVEL
6772   007170  010377  172132           MOV     R3,@DACB        ;LOAD DAC 'B'' TO MIN LEVEL
6773   007174  004737  001426           JSR     PC,STALL        ;DELAY
6774   007200  010477  172120           MOV     R4,@DACA        ;LOAD DAC ''A'' TO MAX LEVEL
6775   007204  010477  172116           MOV     R4,@DACB        ;LOAD DAC 'B'' TO MAX LEVEL
6776   007210  004737  001426           JSR     PC,STALL        ;DELAY
6777   007214  000763                   BR      1$              ;LOOP BACK AND DO AGAIN
6778
6779                                     .SBTTL  I/O SUB-SECTION ''7''     AXV11-C D/A OUTPUT TO A/D INPUT
6780
6781   007216  104401  010770   IOTST7: TYPE    ,MSIO7          ;TELL OPERATOR THE SUB-SECTION NAME
6782   007222  005003                   CLR     R3              ;INITILIZE THE DAC VALUE
6783   007224  104401  001171   1$:     TYPE    ,$CRLF          ;ENSURE FRESH OUTPUT LINE
6784   007230  012705  000010           MOV     #10,R5          ;LOAD LINE WIDTH COUNTER
6785
6786   007234  105277  172056   2$:     INCB    @STREG          ;START CONVERSION
6787   007240  105777  172052   3$:     TSTB    @STREG          ;WAIT FOR A/D DONE
6788   007244  100375                   BPL     3$
6789   007246  010377  172052           MOV     R3,@DACA        ;LOAD ''DAC A'' OUTPUT VALUE
6790   007252  017746  172044           MOV     @ADBUFF,-(SP)   ;READ AND STORE A/D VALUE
6791   007256  104403                   TYPOS
6792   007260     004     001           .BYTE   4,1
6793   007262  005203                   INC     R3              ;UPDATE TO NEXT D/A VALUE
6794   007264  042703  170000           BIC     #170000,R3      ;ENSURE ONLY 12 BITS LONG
6795   007270  005305                   DEC     R5              ;IS THE WIDTH FINISHED ?
6796   007272  001754                   BEQ     1$              ;BR AND START FRESH OUTPUT LINE
6797   007274  104401  011072           TYPE    ,SPACE          ;ENSURE SOME ROOM
6798   007300  000755                   BR      2$              ;AND DO ANOTHER CONVERSION
```

K 4

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 84
CNAXAA.P11   23-DEC-82 14:21   I/O SUB-SECTION "7"   AXV11-C D/A OUTPUT TO A/D INPUT   SEQ 0049

```
6800
6801
6802                                    .SBTTL
6803                                    .SBTTL  END OF EXTERNAL TESTS SECTION
6804                                    .SBTTL
6805                                    .SBTTL       LOGIC TEST SECTION
6806
6807    007302                  BEGINL:
6808    007302  004737  002530  1$:     JSR     PC,BEGL                 ;LOGIC TESTS
6809    007306  012737  007302  010252          MOV     #1$,AGTST              ;ADDRESS FOR EOP
6810    007314  000137  010254          JMP     $EOP                   ;TYPE END OF PASS
6811
6812                                    .SBTTL       AUTO TEST
6813    007320                  BEGINA:
6814    007320  004737  002530  1$:     JSR     PC,BEGL                ;LOGIC TESTS
6815    007324  004737  004104          JSR     PC,WRAP
6816    007330  012737  007320  010252          MOV     #1$,AGTST              ;ADDRESS FOR EOP
6817    007336  000137  010254          JMP     $EOP                   ;TYPE END OF PASS
6818
6819                                    .SBTTL       WRAPAROUND TEST
6820    007342                  BEGINW:
6821    007342  004737  004104  1$:     JSR     PC,WRAP                ;WRAPAROUND TESTS
6822    007346  012737  007342  010252          MOV     #1$,AGTST
6823    007354  000137  010254          JMP     $EOP                   ;INCREMENTS $PASS
6824
6825                                    .SBTTL       DMT TEST STARTUP
6826    007360  032737  000001  001252  BEGIND: BIT     #BIT0,$DEVM    ;TEST IF KWV11-C CONNECTED TO RTC TRIGGER
6827    007366  001402                  BEQ     1$                     ;BR IF NOT
6828    007370  005237  001374          INC     KWAD                   ;SET KW CONNECTED TO AD RTC TRIG - FLAG
6829    007374  032737  000002  001252  1$:     BIT     #BIT1,$DEVM    ;TEST IF KWV11-C CONNECTED TO EXT TRIG AND "F2"
6830    007402  001402                  BEQ     2$                     ;BR IF NOT
6831    007404  005237  001376          INC     KWEX                   ;SET KW CONNECTED TO AD EXT TRIG - FLAG
6832    007410  032737  000004  001252  2$:     BIT     #BIT2,$DEVM    ;TEST IF TEST FIXTURE CONNECTED
6833    007416  001402                  BEQ     3$                     ;BR IF NOT
6834    007420  005237  001366          INC     TC1                    ;SET TEST FIXTURE PRESENT FLAG
6835    007424  032737  000010  001252  3$:     BIT     #BIT3,$DEVM    ;TEST IF AAV11-C CONNECTED TO TEST FIXTURE
6836    007432  001402                  BEQ     4$                     ;BR IF NOT
6837    007434  005237  001370          INC     TC2                    ;SET AAV11-C ANALOG WRAPAROUND FLAG
6838    007440  032737  000020  001252  4$:     BIT     #BIT4,$DEVM    ;TEST IF BEVENT AND "F1" CONNECTED
6839    007446  001402                  BEQ     5$                     ;BR IF NOT
6840    007450  005237  001402          INC     BTEX                   ;SET BEVENT AND "F1" FLAG
6841    007454  032737  000040  001252  5$:     BIT     #BIT5,$DEVM    ;TEST IF MODULE IS AN "ADV11-C"
6842    007462  001402                  BEQ     6$                     ;BR IF NOT
6843    007464  005237  001372          INC     ADV11C                 ;SET "ADV11-C" FLAG
6844    007470  000240          6$:     NOP
6845    007472  000240                  NOP
6846    007474  000240                  NOP
6847    007476  000240                  NOP
6848    007500  000240                  NOP
6849    007502  000137  007320          JMP     BEGINA                 ;RUN THE "AUTO-MODE" TESTS
```

```
6851
6852                                   .SBTTL  ROUTINE TO INITILIZE THE BUS AND VECTOR ADDRESSES
6853   007506  012737  000006  000004  FIXONE: MOV    #6,@#ERRVEC           ;SET UP ERRVEC
6854   007514  013737  001250  001316          MOV    $BASE,STREG           ;RELOAD INITIAL ADDRESSES
6855   007522  013737  001250  001320          MOV    $BASE,ADST1
6856   007530  013737  001250  001322          MOV    $BASE,ADBUFF
6857   007536  013737  001250  001324          MOV    $BASE,DACA            ;PRIME DAC "A" ADDRESS
6858   007544  013737  001250  001326          MOV    $BASE,DACB            ;         "B"
6859   007552  005237  001320                  INC    ADST1
6860   007556  062737  000002  001322          ADD    #2,ADBUFF
6861   007564  062737  000004  001324          ADD    #4,DACA
6862   007572  062737  000006  001326          ADD    #6,DACB
6863   007600  013737  001244  001330          MOV    $VECT1,VECTOR
6864   007606  042737  170000  001330          BIC    #170000,VECTOR
6865   007614  013737  001330  001332          MOV    VECTOR,VECTR1
6866   007622  062737  000002  001332          ADD    #2,VECTR1
6867   007630  013737  001330  001334          MOV    VECTOR,VECTR2
6868   007636  062737  000004  001334          ADD    #4,VECTR2
6869   007644  013737  001330  001336          MOV    VECTOR,VECTR3
6870   007652  062737  000006  001336          ADD    #6,VECTR3
6871                                   ;;LOAD .+2 AND HALT TRAP CATCH;;
6872   007660  012700  000216                  MOV    #216,R0               ;FILL .+2
6873   007664  012701  000214                  MOV    #214,R1               ;LOAD HALT
6874   007670  010021                  1$:     MOV    R0,(R1)+
6875   007672  005021                          CLR    (R1)+
6876   007674  010100                          MOV    R1,R0
6877   007676  005720                          TST    (R0)+
6878   007700  020027  001002                  CMP    R0,#1002
6879   007704  001371                          BNE    1$
6880   007706  000207                          RTS    PC                    ;TEST NEXT A/D
6881
6882
```

M 4

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 86
CNAXAA.P11   23-DEC-82 14:21   ROUTINE TO INITILIZE THE BUS AND VECTOR ADDRESSES   SEQ 0051

```
6884
6885                                  ;;ROUTINE TO AVERAGE 8 CONVERSIONS;;
6886   007710   005037   010044       CONVRT: CLR     OTHER              ;REMOVE EXTRA BITS
6887   007714   012500                CONVTR: MOV     (R5)+,R0           ;GET CHANNEL VALUE
6888   007716   010037   001362               MOV     R0,CHANL
6889   007722   000300                        SWAB    R0
6890   007724   053700   010044··              BIS     OTHER,R0           ;ADD GAIN SELECT IF NEEDED
6891   007730   005037   001360               CLR     TEMP
6892   007734   010077   171356               MOV     R0,@STREG          ;LOAD CHANNEL INTO MIX BITS
6893   007740   012700   010000               MOV     #10000,R0
6894   007744   005300                2$:     DEC     R0
6895   007746   001376                        BNE     2$
6896   007750   012777   001440   171352      MOV     #RETURN,@VECTOR    ;LOAD VECTOR
6897   007756   012700   000010               MOV     #10,R0             ;SET UP COUNTER
6898   007762   152777   000101   171326  1$: BISB    #101,@STREG        ;SET INTRPT. EN., START CONV.
6899   007770   000001                        WAIT                       ;WAIT FOR CONVERSION
6900   007772   017737   171324   010042      MOV     @ADBUFF,77$        ;READ CONVERTED VALUE
6901   010000   042737   170000   010042      BIC     #170000,77$        ;REMOVE HIGH BITS
6902   010006   063737   010042   001360      ADD     77$,TEMP           ;READ BUFFER
6903   010014   005300                        DEC     R0
6904   010016   001361                        BNE     1$                 ;DO 8 TIMES
6905   010020   006237   001360               ASR     TEMP               ;AVERAGE VALUE
6906   010024   006237   001360               ASR     TEMP
6907   010030   006237   001360               ASR     TEMP
6908   010034   005537   001360               ADC     TEMP
6909   010040   000205                        RTS     R5                 ;RETURN
6910   010042   000000                77$:    0
6911   010044   000000                OTHER:  0
6912
6913                                  ;COMPARE $GDDAT AND $BDDAT;;
6914   010046   012537   001124       COMPAR: MOV     (R5)+,$GDDAT       ;GET GOOD DATA
6915   010052   013537   001364               MOV     @(R5)+,SPREAD      ;GET SPREAD
6916   010056   013737   001360   001126      MOV     TEMP,$BDDAT        ;GET BAD(ACTUAL) DATA
6917   010064   013700   001124               MOV     $GDDAT,R0
6918   010070   163700   001126               SUB     $BDDAT,R0          ;GET DIFFERENCE
6919   010074   100001                        BPL     7$
6920   010076   005400                        NEG     R0
6921   010100   020037   001364        7$:    CMP     R0,SPREAD          ;COMPARE IT TO SPREAD
6922   010104   003001                        BGT     10$                ;GO TO ERROR PRINTOUT
6923   010106   005725                        TST     (R5)+              ;BUMP RETURN POINTER AROUND ERROR CALL
6924   010110   000205               10$:     RTS     R5
```

```
6926
6927                                    ;;SUBROUTINE TO TYPE INTRPT. TST MSG.;;
6928  010112  005737  001202    DUMW:   TST     $PASS
6929  010116  001021                    BNE     20$
6930  010120  012737  010162  001110    MOV     #20$,$LPERR
6931  010126  012737  010162  001106    MOV     #20$,$LPADR
6932  010134  104401  011463            TYPE    .METST          ;TYPE ASCIZ STRING
6933  010140  010046                    MOV     R0,-(SP)        ;;SAVE 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
6934  010146  104401  011336            TYPE    .ONAD
6935  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
6936  010162  000207            20$:    RTS     PC
6937
6938  010164  005737  001202    DUMC:   TST     $PASS
6939  010170  001010                    BNE     30$
6940  010172  012737  010212  001110    MOV     #30$,$LPERR
6941  010200  012737  010212  001106    MOV     #30$,$LPADR
6942  010206  104401  011101            TYPE    .DONE
6943  010212  000207            30$:    RTS     PC
6944
6945                                    ;SUBROUTINE TO RESET & SET INTRPT. EN.;
6946  010214  000005            RST:    RESET
6947  010216  052777  000100  170720    BIS     #100,a$TKS
6948  010224  005046                    CLR     -(SP)           ;CLEAR PSW
6949  010226  012746  010234            MOV     #1$,-(SP)
6950  010232  000002                    RTI
6951  010234  000207            1$:     RTS     PC
6952
6953
6954  010236  000002            V2:     2
6955  010240  000012            V12:    12
6956
6957  010242  052777  000100  170674    AGATST: BIS     #100,a$TKS
6958  010250  000137                    JMP     a(PC)+
6959  010252  001522            AGTST:  BEGINO
```

B 5

MAINDEC-11-CNAXA-A   MACY11 30(1046)   23-DEC-82 14:22 PAGE 88   SEQ 0053
CNAXAA.P11   23-DEC-82 14:21   ROUTINE TO INITILIZE THE BUS AND VECTOR ADDRESSES

```
6961
6962
  (1)                                          .SBTTL   END OF PASS ROUTINE
  (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)+,a(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      a#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      a(PC)+              ;;RETURN
  (1)    010354   010242               $RTNAD:  .WORD    AGATST
  (1)
  (1)
  (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
```

```
6964                                            .SBTTL        ASCII MESSAGES
6965
6966    010376  020200  042522  047520   MSIO1:  .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
6967    010454  020200  041523  047101   MSIO2:  .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
6968    010514  020200  027501  020104   MSIO3:  .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
6969    010557     200  047440  052125   MSIO4:  .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
6970    010632  020200  040503  044514   MSIO5:  .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
6971    010677     200  047440  052125   MSIO6:  .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
6972    010770  020200  054101  030526   MSIO7:  .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
6973    011042     136     103     040   CMSG:   .BYTE    136,103,40,40,0          ;CONTROL C ECHO
        011045     040     000
```

```
6974  011047      136    101      040  AMSG:   .BYTE   136,101,40,40,0              ;CONTROL A ECHO
      011052      040    000
6975  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
6976  011067      103  000110          CH:     .ASCIZ  /CH/
6977  011072      040    040      040  SPACE:  .BYTE   40,40,40,40,0
      011075      040    000
6978  011077      077    000          QUEST:  .BYTE   77,0
6979  011101      040  020040   042040  DONE:   .ASCIZ  /   DONE/<15><12>
      011106    047117  006505   000012
6980  011114    000056          ADOT:   .ASCIZ  \.\
6981  011116    000060          AZERO:  .ASCIZ  \0\
6982  011120    000057          SLASH:  .ASCIZ  #/#
6983  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
6984  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
6985  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
6986  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
6987  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
6988  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
6989  011443      015  050012   044522  HEAD5:  .ASCII  <15><12>/PRINT VALUES--/
      011450    052116  053040   046101
```

```
          011456   042525   026523     055
6990      011463      015   020012   047105   METST:  .ASCIZ   <15><12>/ ENTERING TEST /
          011470   042524   044522   043516
          011476   052040   051505   020124
          011504      000
6991      011505      015      012            MSKWAD: .BYTE   15,12
6992      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
6993      011567      015      012            MSKWEX: .BYTE   15,12
6994      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
6995      011676      015      012            MSMAEX: .BYTE   15,12
6996      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   042514   021040
          011774   031106   020042   047111
          012002   052123   046101   042514
          012010   024504   037440   000040
6997      012016      015      012            MSGNEX: .BYTE   15,12
6998      012020   042507   042516   040522           .ASCIZ   \GENERATE ONE TRIGGER SIGNAL\
          012026   042524   047440   042516
          012034   052040   044522   043507
          012042   051105   051440   043511
          012050   040516   000114
6999      012054      015      012            MSBTEX: .BYTE   15,12
7000      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
7001       012147    200   051511  052040   MSADV:  .ASCIZ  <200>\IS THIS AN ADV11-C ? \
           012154  044510  020123  047101
           012162  040440  053104  030461
           012170  041455  037440  000040
7002       012176    015     012             MSTC1:  .BYTE   15,12
7003       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
7004       012255    015     012             MSTC2:  .BYTE   15,12
7005       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
7006       012345    015     012             MSG70:  .BYTE   15,12
7007       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
7008       012442  005015  035114  046040            .ASCII  <15><12>/L: LOGIC TESTS ONLY/
           012450  043517  041511  052040
           012456  051505  051524  047440
           012464  046116    131
7009       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
7010       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
```

```
7011  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
7012  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
7013  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
7014  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
7015  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
7016  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
7017  013137     015  020012  000040   HEAD2: .ASCIZ  <15><12>/ /
7018  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
7019  013215     123  040524  052524   EM1:   .ASCIZ  /STATUS REG. ERROR/
      013222  020123  042522  027107
      013230  042440  051122  051117
      013236     000
7020  013237     106  044501  042514   EM2:   .ASCIZ  /FAILED TO INTERRUPT/
      013244  020104  047524  044440
      013252  052116  051105  052522
```

```
         013260   052120      000
7021     013263      125   042516   050130   EM3:    .ASCIZ   /UNEXPECTED INTERRUPT/
         013270   041505   042524   020104
         013276   047111   042524   051122
         013304   050125   000124
7022     013310   051105   047522   020122   EM4:    .ASCIZ   #ERROR ON A/D CHANNEL#
         013316   047117   040440   042057
         013324   041440   040510   047116
         013332   046105      000
7023     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
7024     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    46101   051440   051120
         013440   040505   020104   040440
         013446   052103   040525   000114
7025     013454   051105   050122   020103   DH3:    .ASCIZ   /ERRPC    STREG    ACTUAL/
         013462   020040   052123   042522
         013470   020107   020040   040440
         013476   052103   040525   000114
7026                                                 .EVEN
7027
7028     013504   001116   001316   001124   DT1:    $ERRPC, STREG, $GDDAT, $BDDAT,0
         013512   001126   000000
7029     013516   001116   001316   001362   DT2:    $ERRPC,STREG,CHANL,$GDDAT,SPREAD,$BDDAT,0
         013524   001124   001364   001126
         013532   000000
7030     013534   001116   001316   001126   DT3:    $ERRPC,STREG,$BDDAT,0
         013542   000000
7031     013544   000000                     DF1:    0
```

```
7033
7034                                    .SBTTL  TTY INPUT ROUTINE
 (1)
 (2)                                    ;;****************************************************************
 (1)
 (1)
 (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    a$TKS            ;;DISABLE TTY KEYBOARD INTERRUPTS
    (1)  013770  005726                    TST    (SP)+            ;;CLEAN CHAR OFF STACK
    (1)  013772  105777  165146    31$:    TSTB   a$TKS            ;;WAIT FOR A CHAR
    (1)  013776  100375                    BPL    31$              ;;LOOP UNTIL ITS THERE
    (1)  014000  117746  165142            MOVB   a$TKB,-(SP)      ;;GET THE CHARACTER
    (1)  014004  042716  177600            BIC    #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
    (1)  014010  022627  000021            CMP    (SP)+,#21        ;;IS IT A CONTROL-Q?
    (1)  014014  001366                    BNE    31$              ;;BRANCH IF NO
    (1)  014016  012777  000100  165120    MOV    #100,a$TKS       ;;REENABLE TTY KEYBOARD INTERRUPTS
    (1)  014024  000002                    RTI                     ;;RETURN
    (1)  014026  005237  013546    32$:    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$:     MOVB   (SP)+,a$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   a$TKS            ;;IS A CHAR WAITING?
    (1)  014116  100121                    BPL    15$              ;;IF NOT, EXIT
    (1)  014120  117746  165022            MOVB   a$TKB,-(SP)      ;;YES
    (1)  014124  042716  177600            BIC    #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
    (1)  014130  021627  000007            CMP    (SP),#7          ;;IS IT A CONTROL-G?
    (1)  014134  001300                    BNE    2$               ;;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,$TKINT        ;;FLUSH THE TTY INPUT QUEUE
    (1)  014154  005077  164764            CLR    a$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$:    CLR    -(SP)              ;;CLEAR COUNTER
(1)   014212  005046                      CLR    -(SP)
(1)                                       ;;THE NEW SWR
(1)   014214  105777  164724      7$:     TSTB   a$TKS              ;;CHAR THERE?
(1)   014220  100375                      BPL    7$                 ;;IF NOT TRY AGAIN
(1)
(1)   014222  117746  164720              MOVB   a$TKB,-(SP)        ;;PICK UP CHAR
(1)   014226  042716  177600              BIC    #^C177,(SP)        ;;MAKE IT 7-BIT ASCII
(1)
(1)   014232  021627  000003              CMP    (SP),#3            ;;IS IT A CONTROL-C?
(1)   014236  001015                      BNE    9$                 ;;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,a$TKS         ;;ALLOW TTY KEYBOARD INTERRUPTS
(1)   014266  000137  001530      8$:     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),aSWR         ;;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,a$TKS         ;;RE-ENABLE TTY KBD INTERRUPTS
(1)   014362  000002      15$:    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)
(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
 (1)
 (1)
 (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 O'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    #$TTYIN+32.,R3
 (1)                                      ;;BUFFER FULL?
 (1)    014550  101456                     BLOS   4$                    ;;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         ;;TYPE 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
```

```
7036
7037                              .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
 (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)                              ;*BITS ARE IN $HIOCT                                ;;HIGH ORDER
 (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
```

```
7039
7040                                    .SBTTL  POWER DOWN AND UP ROUTINES
 (1)
 (2)                            ;;*********************************************************************
 (1)                            ;POWER DOWN ROUTINE
 (1)    015174   012737   015334   000024   $PWRDN:  MOV     #$ILLUP,a#PWRVEC ;;SET FOR FAST UP
 (1)    015202   012737   000300   000026            MOV     #PR6,a#PWRVEC+2 ;;PRIO:6
 (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     aSWR,-(SP)       ;;PUSH aSWR ON STACK
 (1)    015230   010637   015340                     MOV     SP,$SAVR6        ;;SAVE SP
 (1)    015234   012737   015246   000024            MOV     #$PWRUP,a#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,a#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)+,aSWR       ;;POP STACK INTO aSWR
 (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,a#PWRVEC ;;SET UP THE POWER DOWN VECTOR
 (1)    015320   012737   000300   000026            MOV     #PR6,a#PWRVEC+2 ;;PRIO:6
 (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
```

C 6

MAINDEC-11-CNAXA-A   MACY11 30(1046)   23-DEC-82 14:22 PAGE 93   SEQ 0067
CNAXAA.P11   23-DEC-82 14:21   POWER DOWN AND UP ROUTINES

```
7042
7043
 (1)                                    .SBTTL   SCOPE HANDLER ROUTINESTARS
 (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                          $SCOPE:
 (1)    015352   104410                         CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
 (1)    015352   032777   040000  163556 1$:    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
7044                          .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  016466           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$:
     (1)   016030  000002                          RTI                      ;;RETURN
    7045                                    .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
     (1)
     (2)                                    ;;*************************************************************
     (1)                                    ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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$:    MOV    (SP)+,R0         ;;RESTORE R0
     (1)
     (1)   016136  104401  001171                   TYPE   ,$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
```

```
 7047
 7048                                  .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  016456      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  016444      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  016444                    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,$TYPEC           ;;TYPE A SPACE
   (1)   016364  132737  000007  016444            BITB    #7,$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  105777  162544       $TYPEC:      TSTB    @$TPS               ;;WAIT UNTIL PRINTER IS READY
   (1)   016404  100375                            BPL     $TYPEC
   (1)   016406  116677  000002  162536            MOVB    2(SP),@$TPB         ;;LOAD CHAR TO BE TYPED INTO DATA REG.
   (1)
   (1)   016414  122766  000015  000002  CMPB      #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
   (1)   016422  001003                            BNE     1$                  ;;BRANCH IF NO
   (1)   016424  105037  016444                    CLRB    $CHARCNT            ;;YES--CLEAR CHARACTER COUNT
   (1)   016430  000406                            BR      $TYPEX              ;;EXIT
   (1)   016432  122766  000012  000002  1$:       CMPB    #LF,2(SP)           ;;IS CHARACTER A LINE FEED?
   (1)   016440  001402                            BEQ     $TYPEX              ;;BRANCH IF YES
   (1)   016442  105227                            INCB    (PC)+               ;;COUNT THE CHARACTER
   (1)   016444  000000       $CHARCNT:.WORD       0                  ;;CHARACTER COUNT STORAGE
   (1)   016446  000207       $TYPEX: RTS          PC
   (1)
   (1)
   (1)
  7049                          .SBTTL  APT COMMUNICATIONS ROUTINE
   (1)
   (2)                              ;;***************************************************************
   (1)   016450  112737  000001  016714  $ATY1:    MOVB    #1,$FFLG           ;;TO REPORT FATAL ERROR
   (1)   016456  112737  000001  016712  $ATY3:    MOVB    #1,$MFLG           ;;TO TYPE A MESSAGE
   (1)   016464  000403                            BR      $ATYC
   (1)   016466  112737  000001  016714  $ATY4:    MOVB    #1,$FFLG           ;;TO ONLY REPORT FATAL ERROR
   (1)   016474                         $ATYC:
   (3)   016474  010046                            MOV     R0,-(SP)            ;;PUSH R0 ON STACK
   (3)   016476  010146                            MOV     R1,-(SP)            ;;PUSH R1 ON STACK
   (1)   016500  105737  016712                    TSTB    $MFLG               ;;SHOULD TYPE A MESSAGE?
   (1)   016504  001450                            BEQ     5$                  ;;IF NOT:  BR
   (1)   016506  122737  000001  001214            CMPB    #APTENV,$ENV        ;;OPERATING UNDER APT?
   (1)   016514  001031                            BNE     3$                  ;;IF NOT:  BR
   (1)   016516  132737  000100  001215            BITB    #APTSPOOL,$ENVM     ;;SHOULD SPOOL MESSAGES?
   (1)   016524  001425                            BEQ     3$                  ;;IF NOT:  BR
   (1)   016526  017600  000004                    MOV     @4(SP),R0           ;;GET MESSAGE ADDR.
   (1)   016532  062766  000002  000004            ADD     #2,4(SP)            ;;BUMP RETURN ADDR.
   (1)   016540  005737  001174       1$:          TST     $MSGTYPE            ;;SEE IF DONE W/ LAST XMISSION?
   (1)   016544  001375                            BNE     1$                  ;;IF NOT:  WAIT
   (1)   016546  010037  001210                    MOV     R0,$MSGAD
   (1)                                              ;;PUT ADDR IN MAILBOX
   (1)   016552  105720                  2$:       TSTB    (R0)+               ;;FIND END OF MESSAGE
   (1)   016554  001376                            BNE     2$
   (1)   016556  163700  001210                    SUB     $MSGAD,R0           ;;SUB START OF MESSAGE
   (1)   016562  006200                            ASR     R0                  ;;GET MESSAGE LNGTH IN WORDS
   (1)   016564  010037  001212                    MOV     R0,$MSGLGT          ;;PUT LENGTH IN MAILBOX
   (1)   016570  012737  000004  001174            MOV     #4,$MSGTYPE         ;;TELL APT TO TAKE MSG.
   (1)   016576  000413                            BR      5$
```

```
(1)  016600  017637  000004  016624   3$:     MOV    @4(SP),4$           ;;PUT MSG ADDR IN JSR LINKAGE
(1)  016606  062766  000002  000004           ADD    #2,4(SP)                    ;;BUMP RETURN ADDRESS
(3)  016614  013746  177776                    MOV    177776,-(SP)        ;;PUSH 177776 ON STACK
(1)  016620  004737  016166                    JSR    PC,$TYPE            ;;CALL TYPE MACRO
(1)  016624  000000                    4$:     .WORD  0
(1)  016626                            5$:
(1)  016626  105757  016714          10$:      TSTB   $FFLG               ;;SHOULD REPORT FATAL ERROR?
(1)  016632  001416                            BEQ    12$                 ;;IF NOT:  BR
(1)  016634  005737  001214                    TST    $ENV                ;;RUNNING UNDER APT?
(1)  016640  001413                            BEQ    12$                 ;;IF NOT:  BR
(1)  016642  005737  001174          11$:      TST    $MSGTYPE            ;;FINISHED LAST MESSAGE?
(1)  016646  001375                            BNE    11$                 ;;IF NOT:  WAIT
(1)  016650  017637  000004  001176            MOV    @4(SP),$FATAL       ;;GET ERROR #
(1)  016656  062766  000002  000004            ADD    #2,4(SP)                    ;;BUMP RETURN ADDR.
(1)  016664  005237  001174                    INC    $MSGTYPE            ;;TELL APT TO TAKE ERROR
(1)  016670  105037  016714          12$:      CLRB   $FFLG               ;;CLEAR FATAL FLAG
(1)  016674  105037  016713                    CLRB   $LFLG               ;;CLEAR LOG FLAG
(1)  016700  105037  016712                    CLRB   $MFLG               ;;CLEAR MESSAGE FLAG
(3)  016704  012601                            MOV    (SP)+,R1            ;;POP STACK INTO R1
(3)  016706  012600                            MOV    (SP)+,R0            ;;POP STACK INTO R0
(1)  016710  000207                            RTS    PC                  ;;RETURN
(1)  016712    000                    $MFLG:   .BYTE  0                   ;;MESSG. FLAG
(1)  016713    000                    $LFLG:   .BYTE  0
(1)                                                                       ;;LOG FLAG
(1)  016714    000                    $FFLG:   .BYTE  0                   ;;FATAL FLAG
(1)          016716                            .EVEN
(1)          000200                    APTSIZE=200
(1)          000001                    APTENV=001
(1)          000100                    APTSPOOL=100
(1)          000040                    APTCSUP=040
```

I 6

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 95
CNAXAA.P11   23-DEC-82 14:21   APT COMMUNICATIONS ROUTINE   SEQ 0073

```
7051
7052                                    .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) 016716 017646 000000      $TYPOS: MOV     @(SP),-(SP)        ;;PICKUP THE MODE
  (1) 016722 116637 000001 017141       MOVB    1(SP),$OFILL       ;;LOAD ZERO FILL SWITCH
  (1) 016730 112637 017143             MOVB    (SP)+,$OMODE+1     ;;NUMBER OF DIGITS TO TYPE
  (1) 016734 062716 000002             ADD     #2,(SP)            ;;ADJUST RETURN ADDRESS
  (1) 016740 000406                    BR      $TYPON
  (1) 016742 112737 000001 017141 $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
  (1) 016750 112737 000006 017143       MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
  (1) 016756 112737 000005 017140 $TYPON: MOVB    #5,$OCNT       ;;SET THE ITERATION COUNT
  (1) 016764 010346                    MOV     R3,-(SP)           ;;SAVE R3
  (1) 016766 010446                    MOV     R4,-(SP)           ;;SAVE R4
  (1) 016770 010546                    MOV     R5,-(SP)           ;;SAVE R5
  (1) 016772 113704 017143             MOVB    $OMODE+1,R4        ;;GET THE NUMBER OF DIGITS TO TYPE
  (1) 016776 005404                    NEG     R4
  (1) 017000 062704 000006             ADD     #6,R4              ;;SUBTRACT IT FOR MAX. ALLOWED
  (1) 017004 110437 017142             MOVB    R4,$OMODE          ;;SAVE IT FOR USE
  (1) 017010 113704 017141             MOVB    $OFILL,R4          ;;GET THE ZERO FILL SWITCH
  (1) 017014 016605 000012             MOV     12(SP),R5          ;;PICKUP THE INPUT NUMBER
  (1) 017020 005003                    CLR     R3                 ;;CLEAR THE OUTPUT WORD
  (1) 017022 006105              1$:   ROL     R5                 ;;ROTATE MSB INTO "C"
  (1) 017024 000404                    BR      3$                 ;;GO DO MSB
  (1) 017026 006105              2$:   ROL     R5                 ;;FORM THIS DIGIT
  (1) 017030 006105                    ROL     R5
  (1) 017032 006105                    ROL     R5
  (1) 017034 010503                    MOV     R5,R3
  (1) 017036 006103              3$:   ROL     R3                 ;;GET LSB OF THIS DIGIT
  (1) 017040 105337 017142             DECB    $OMODE             ;;TYPE THIS DIGIT?
  (1) 017044 100016                    BPL     7$                 ;;BR IF NO
  (1) 017046 042703 177770             BIC     #177770,R3         ;;GET RID OF JUNK
  (1) 017052 001002                    BNE     4$                 ;;TEST FOR 0
  (1) 017054 005704                    TST     R4                 ;;SUPPRESS THIS 0?
```

```
    (1)   017056  001403                        BEQ     5$              ;;BR IF YES
    (1)   017060  005204             4$:         INC     R4              ;;DON'T SUPPRESS ANYMORE O'S
    (1)   017062  052703  000060                 BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
    (1)   017066  052703  000040     5$:         BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
    (1)   017072  110337  017136                 MOVB    R3,8$           ;;SAVE FOR TYPING
    (1)   017076  104401  017136                 TYPE    .8$             ;;GO TYPE THIS DIGIT
    (1)   017102  105337  017140     7$:         DECB    $OCNT           ;;COUNT BY 1
    (1)   017106  003347                         BGT     2$              ;;BR IF MORE TO DO
    (1)   017110  002402                         BLT     6$              ;;BR IF DONE
    (1)   017112  005204                         INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
    (1)   017114  000744                         BR      2$              ;;GO DO THE LAST DIGIT
    (1)   017116  012605             6$:         MOV     (SP)+,R5        ;;RESTORE R5
    (1)   017120  012604                         MOV     (SP)+,R4        ;;RESTORE R4
    (1)   017122  012603                         MOV     (SP)+,R3        ;;RESTORE R3
    (1)   017124  016666  000002 000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
    (1)   017132  012616                         MOV     (SP)+,(SP)
    (1)   017134  000002                         RTI                     ;;RETURN
    (1)   017136  000                8$:         .BYTE   0               ;;STORAGE FOR ASCII DIGIT
    (1)   017137  000                            .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
    (1)   017140  000                $OCNT:      .BYTE   0               ;;OCTAL DIGIT COUNTER
    (1)   017141  000                $OFILL:     .BYTE   0               ;;ZERO FILL SWITCH
    (1)   017142  000000             $OMODE:     .WORD   0               ;;NUMBER OF DIGITS TO TYPE
7053                                 .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)   017144  010146             $TYPBN: MOV     R1,-(SP)            ;;SAVE R1 ON THE STACK
    (1)   017146  016601  000006             MOV     6(SP),R1            ;;GET THE INPUT NUMBER
    (1)   017152  000261                     SEC                         ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
    (1)   017154  112737  000060 017216 1$:   MOVB    #'0,$BIN           ;;SET CHARACTER TO AN ASCII "0".
    (1)   017162  006101                     ROL     R1                  ;;GET THIS BIT
    (1)   017164  001406                     BEQ     2$                  ;;DONE?
    (1)   017166  105537  017216             ADCB    $BIN                ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
    (1)   017172  104401  017216             TYPE    .$BIN               ;;GO TYPE THIS BIT
    (1)   017176  000241                     CLC                         ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
    (1)   017200  000765                     BR      1$                  ;;GO DO THE NEXT BIT
    (1)   017202  012601             2$:     MOV     (SP)+,R1            ;;POP THE STACK INTO R1
    (1)   017204  016666  000002 000004      MOV     2(SP),4(SP)         ;;ADJUST THE STACK
    (1)   017212  012616                     MOV     (SP)+,(SP)
    (1)   017214  000002                     RTI                         ;;RETURN TO USER
    (1)   017216  000     000        $BIN:   .BYTE   0,0                 ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
7054                                 .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)   017220                      $TYPDS:
  (3)   017220  010046                      MOV     R0,-(SP)            ;;PUSH R0 ON STACK
  (3)   017222  010146                      MOV     R1,-(SP)            ;;PUSH R1 ON STACK
  (3)   017224  010246                      MOV     R2,-(SP)            ;;PUSH R2 ON STACK
  (3)   017226  010346                      MOV     R3,-(SP)            ;;PUSH R3 ON STACK
  (3)   017230  010546                      MOV     R5,-(SP)            ;;PUSH R5 ON STACK
  (1)   017232  012746  020200              MOV     #20200,-(SP)        ;;SET BLANK SWITCH AND SIGN
  (1)   017236  016605  000020              MOV     20(SP),R5           ;;GET THE INPUT NUMBER
  (1)   017242  100004                      BPL     1$                  ;;BR IF INPUT IS POS.
  (1)   017244  005405                      NEG     R5                  ;;MAKE THE BINARY NUMBER POS.
  (1)   017246  112766  000055  000001      MOVB    #'-,1(SP)           ;;MAKE THE ASCII NUMBER NEG.
  (1)   017254  005000              1$:     CLR     R0                  ;;ZERO THE CONSTANTS INDEX
  (1)   017256  012703  017434              MOV     #$DBLK,R3           ;;SETUP THE OUTPUT POINTER
  (1)   017262  112723  000040              MOVB    #' ,(R3)+           ;;SET THE FIRST CHARACTER TO A BLANK
  (1)   017266  005002              2$:     CLR     R2                  ;;CLEAR THE BCD NUMBER
  (1)   017270  016001  017424              MOV     $DTBL(R0),R1        ;;GET THE CONSTANT
  (1)   017274  160105              3$:     SUB     R1,R5               ;;FORM THIS BCD DIGIT
  (1)   017276  002402                      BLT     4$                  ;;BR IF DONE
  (1)   017300  005202                      INC     R2                  ;;INCREASE THE BCD DIGIT BY 1
  (1)   017302  000774                      BR      3$
  (1)   017304  060105              4$:     ADD     R1,R5               ;;ADD BACK THE CONSTANT
  (1)   017306  005702                      TST     R2                  ;;CHECK IF BCD DIGIT=0
  (1)   017310  001002                      BNE     5$                  ;;FALL THROUGH IF 0
  (1)   017312  105716                      TSTB    (SP)                ;;STILL DOING LEADING 0'S?
  (1)   017314  100407                      BMI     7$                  ;;BR IF YES
  (1)   017316  106316              5$:     ASLB    (SP)                ;;MSD?
  (1)   017320  103003                      BCC     6$                  ;;BR IF NO
  (1)   017322  116663  000001  177777      MOVB    1(SP),-1(R3)        ;;YES--SET THE SIGN
  (1)   017330  052702  000060      6$:     BIS     #'0,R2              ;;MAKE THE BCD DIGIT ASCII
  (1)   017334  052702  000040      7$:     BIS     #' ,R2              ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
  (1)   017340  110223                      MOVB    R2,(R3)+            ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
  (1)   017342  005720                      TST     (R0)+               ;;JUST INCREMENTING
  (1)   017344  020027  000010              CMP     R0,#10              ;;CHECK THE TABLE INDEX
  (1)   017350  002746                      BLT     2$                  ;;GO DO THE NEXT DIGIT
  (1)   017352  003002                      BGT     8$                  ;;GO TO EXIT
  (1)   017354  010502                      MOV     R5,R2               ;;GET THE LSD
  (1)   017356  000764                      BR      6$                  ;;GO CHANGE TO ASCII
  (1)   017360  105726              8$:     TSTB    (SP)+               ;;WAS THE LSD THE FIRST NON-ZERO?
  (1)   017362  100003                      BPL     9$                  ;;BR IF NO
  (1)   017364  116663  177777  177776      MOVB    -1(SP),-2(R3)       ;;YES--SET THE SIGN FOR TYPING
  (1)   017372  105013              9$:     CLRB    (R3)                ;;SET THE TERMINATOR
  (3)   017374  012605                      MOV     (SP)+,R5            ;;POP STACK INTO R5
  (3)   017376  012603                      MOV     (SP)+,R3            ;;POP STACK INTO R3
  (3)   017400  012602                      MOV     (SP)+,R2            ;;POP STACK INTO R2
  (3)   017402  012601                      MOV     (SP)+,R1            ;;POP STACK INTO R1
  (3)   017404  012600                      MOV     (SP)+,R0            ;;POP STACK INTO R0
  (1)   017406  104401  017434              TYPE    ,$DBLK              ;;NOW TYPE THE NUMBER
  (1)   017412  016666  000002  000004      MOV     2(SP),4(SP)         ;;ADJUST THE STACK
  (1)   017420  012616                      MOV     (SP)+,(SP)
  (1)   017422  000002                      RTI                         ;;RETURN TO USER
  (1)   017424  023420              $DTBL:  10000.
  (1)   017426  001750                      1000.
  (1)   017430  000144                      100.
  (1)   017432  000012                      10.
```

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 95-3 6
CNAXAA.P11   23-DEC-82 14:21   CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0076

```
(1)  017434  000004                    $DBLK:  .BLKW   4
```

```
7056
7057                                    .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)   017444   010046                  $TRAP:   MOV    R0,-(SP)        ::SAVE R0
 (1)   017446   016600   000002                  MOV    2(SP),R0        ;;GET TRAP ADDRESS
 (1)   017452   005740                           TST    -(R0)           ;;BACKUP BY 2
 (1)   017454   111000                           MOVB   (R0),R0         ;;GET RIGHT BYTE OF TRAP
 (1)   017456   006300                           ASL    R0              ;;POSITION FOR INDEXING
 (1)   017460   016000   017500                  MOV    $TRPAD(R0),R0   ;;INDEX TO TABLE
 (1)   017464   000200                           RTS    R0              ;;GO TO ROUTINE
 (1)
 (1)
 (1)                                    ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
 (1)
 (1)   017466   011646                  $TRAP2:  MOV    (SP),-(SP)      ;;MOVE THE PC DOWN
 (1)   017470   016666   000004 000002          MOV    4(SP),2(SP)     ;;MOVE THE PSW DOWN
 (1)   017476   000002                           RTI                    ;;RESTORE THE PSW
 (3)
 (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)   017500   017466                  $TRPAD:  .WORD  $TRAP2
 (3)   017502   016166                           $TYPE   ;;CALL=TYPE    TRAP+1(104401)  TTY TYPEOUT ROUTINE
 (3)   017504   016742                           $TYPOC  ;;CALL=TYPOC   TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
 (3)   017506   016716                           $TYPOS  ;;CALL=TYPOS   TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
 (3)   017510   016756                           $TYPON  ;;CALL=TYPON   TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
 (3)   017512   017220                           $TYPDS  ;;CALL=TYPDS   TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
 (3)   017514   017144                           $TYPBN  ;;CALL=TYPBN   TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
 (1)
 (3)   017516   014172                           $GTSWR  ;;CALL=GTSWR   TRAP+7(104407)  GET SOFT-SWR SETTING
 (1)
 (3)   017520   014102                           $CKSWR  ;;CALL=CKSWR   TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
 (3)   017522   014444                           $RDCHR  ;;CALL=RDCHR   TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
 (3)   017524   014534                           $RDLIN  ;;CALL=RDLIN   TRAP+12(104412) TTY TYPEIN STRING ROUTINE
 (3)   017526   015072                           $RDOCT  ;;CALL=RDOCT   TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
7058   017530   004060                           TEST    ;;CALL=CHECK   TRAP+14(104414)
7059   017532   004052                           TESTIT  ;;CALL=CHKIT   TRAP+15(104415)
7060                                    .EVEN
7061   017534   000240                           NOP            ;JUST TO FIND THE LAST LOCATION OF THE PROGRAM
7062                                    ;;THE FOLLOWING CALL TO CNMAC2.SML WAS ADDED TO INIT BRKVEC
7063                                    ;;AND LKVEC TO BE SPECIFIC TO 11/21 PROCESSOR.
7064            017536                  POINT=.                 ;SAVE POINTER
 (1)            000100                  .=100
 (1)   000100   017536                  $CLKVEC                 ;LKVEC  HANDLER
 (1)   000102   000300                  300                     ;INTERRUPT HANDLER PRI
 (1)            000140                  .=140                   ;BRKVEC
```

```
 (1)   000140  170000                   170000                  ;ODT START ADDRESS
 (1)   000142  000300                   300                     ;PRIORITY
 (1)            017536                   .=POINT                 ;RESTORE POINTER
 (1)   017536  104401  017544           $CLKVEC:        TYPE,CLKMES
 (1)   017542  000000                                   HALT
 (1)   017544  005015  045514  042526   CLKMES:  .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
 (1)   017552  020103  047111  042524
 (1)   017560  051122  050125  020124
 (1)   017566  020055  044504  041523
 (1)   017574  047117  042516  052103
 (1)   017602  046040  041524  000040
7065           000001                   .END
```

B 7

MAINDEC-11-CNAXA-A   MACY11 30(1046)   23-DEC-82 14:22 PAGE 97   SEQ 0079
CNAXAA.P11   23-DEC-82 14:21   CROSS REFERENCE TABLE -- USER SYMBOLS

```
ABASE = 175400      5694#    5731     5771     5772     5773     5774     5775
ACDW1 = 000000      5731
ACDW2 = 000000      5731
ACPUOP= 000000      5731
ADBUFF  001322      5773#    5942     6002     6010     6026     6052     6076     6088     6101     6121     6130     6141     6153
                    6382     6411     6441     6485     6517     6547     6576     6678     6790     6856*    6860*    6900

ADDW0 = 000000      5731
ADDW1 = 000000      5731
ADDW10= 000000      5731
ADDW11= 000000      5731
ADDW12= 000000      5731
ADDW13= 000000      5731
ADDW14= 000000      5731
ADDW15= 000000      5731
ADDW2 = 000000      5731
ADDW3 = 000000      5731
ADDW4 = 000000      5731
ADDW5 = 000000      5731
ADDW6 = 000000      5731
ADDW7 = 000000      5731
ADDW8 = 000000      5731
ADDW9 = 000000      5731
ADEVCT= 000000      5731
ADEVM = 000000      5731
ADOT    011114      6655     6980#
ADST1   001320      5772#    6855*    6859*
ADV11C  001372      5793#    5877     6355     6377     6406     6843*
AENV  = 000000      5731
AENVM = 000000      5731
AFATAL= 000000      5731
AGATST  010242      6957#    6962
AGTST   010252      6809*    6816*    6822*    6959#
AMADR1= 000000      5731
AMADR2= 000000      5731
AMADR3= 000000      5731
AMADR4= 000000      5731
AMAMS1= 000000      5731
AMAMS2= 000000      5731
AMAMS3= 000000      5731
AMAMS4= 000000      5731
AMSG    011047      6974#
AMSGAD= 000000      5731
AMSGLG= 000000      5731
AMSGTY= 000000      5731
AMTYP1= 000000      5731
AMTYP2= 000000      5731
AMTYP3= 000000      5731
AMTYP4= 000000      5731
APASS = 000000      5731
APRIOR= 000200      5696#    5731
APTCSU= 000040      7048     7049#
APTENV= 000001      7044     7048     7049#
APTSIZ= 000200      5839     7049#
APTSPO= 000100      7048     7049#
ASKTA   001456      5820#    5855     5860     5866     5871     5875     5879     5883
ASWREG= 000000      5731
```

C 7

MAINDEC-11-CNAXA-A   MACY11 30(1046)   23-DEC-82 14:22   PAGE 97-1   SEQ 0080
CNAXAA.P11   23-DEC-82 14:21   CROSS REFERENCE TABLE -- USER SYMBOLS

```
ATESTN= 000000     5731
AUNIT = 000000     5731
AUSWR = 000000     5731
AVECT1= 000220     5695#    5731    5776    5777    5778    5779
AVECT2= 000000     5731
AZERO   011116     6657*    6660*   6661    6663*   6666*   6667    6981#
BARF    001356     5787#    5807
BEGINA  007320     5912     5918    6813#   6849
BEGIND  007360     5854     6826#
BEGINL  007302     5914     5920    6807#
BEGINW  007342     5916     5922    6820#
BEGINO  001522     5699     5836#   6959
BEGIN2  001530     5700     5838#   7034
BEGL    002530     5938#    6808    6814
BEGST   001534     5837     5839#
BIT0  = 000001     5692#    6038    6068    6072    6134    6826
BIT00 = 000001     5692#
BIT01 = 000002     5692#
BIT02 = 000004     5692#
BIT03 = 000010     5692#
BIT04 = 000020     5692#
BIT05 = 000040     5692#
BIT06 = 000100     5692#
BIT07 = 000200     5692#
BIT08 = 000400     5692#    7043
BIT09 = 001000     5692#    7043    7044
BIT1  = 000002     5692#    6829
BIT10 = 002000     5692#    7044
BIT11 = 004000     5692#    7043
BIT12 = 010000     5692#    5950    6725    6734
BIT13 = 020000     5692#    7044
BIT14 = 040000     5692#    5954    6056    6058    7043
BIT15 = 100000     5692#    5983    5996    6006    6056    6058    6071    6127
BIT2  = 000004     5692#    5974    6832
BIT3  = 000010     5692#    5978    6835
BIT4  = 000020     5692#    5968    6094    6111    6147    6838
BIT5  = 000040     5692#    5964    6081    6127    6129    6841
BIT6  = 000100     5692#    5959    6038    6042
BIT7  = 000200     5692#    6004    6042    6071    6081    6094    6111    6127    6147
BIT8  = 000400     5692#    5946
BIT9  = 001000     5692#    5787
BPTVEC= 000014     5692#
BRKVEC= 000140     5692#
BTEX    001402     5797#    5865*   5873    6145    6840*
CCHAN   011122     6613     6699    6983#
CH      011067     6623     6649    6976#
CHANL   001362     5789#    6888*   7029
CHAN00= 000000     5705#    6195    6228    6384
CHAN01= 000001     5706#    6202    6254
CHAN02= 000002     5707#    6209    6280
CHAN03= 000003     5708#    6216    6304    6322    6330    6338    6346
CHAN04= 000004     5709#    6233
CHAN05= 000005     5710#    6259
CHAN06= 000006     5711#    6285
CHAN07= 000007     5712#    6308
CHAN10= 000010     5713#    6241
```

```
CHAN11= 000011          5714#    6267
CHAN12= 000012          5715#    6293
CHAN13= 000013          5716#    6358    6413
CHAN14= 000014          5717#    6443    6490
CHAN15= 000015          5718#    6450    6520
CHAN16= 000016          5719#    6457    6550
CHAN17= 000017          5720#    6464    6579
CHECK = 104414          5999     6007    6018    6027    6073    6086    6099    6119    6150    7058#
CHKIT = 104415          5947     5955    5960    5965    5969    5975    5979    5984    7059#
CKSWR = 104410          7043     7044    7057#
CLKMES  017544          7064#
CMSG    011042          6973#
COMPAR  010046          6196     6203    6210    6217    6234    6242    6260    6268    6286    6294    6309    6323    6331
                        6339     6347    6359    6385    6414    6444    6451    6458    6465    6491    6521    6551    6580
                        6914#
CONVRT  007710          6194     6201    6208    6215    6227    6232    6240    6253    6258    6266    6279    6284    6292
                        6303     6307    6357    6383    6412    6442    6449    6456    6463    6489    6519    6549    6578
                        6886#
CONVTR  007714          6321     6329    6337    6345    6626    6710    6887#
CR    = 000015          5692#    7048
CRLF  = 000200          5692#    5848    7048
CRWR    011276          6114     6986#
DACA    001324          5774#    5943    6179*   6380*   6390*   6395*   6714*   6726*   6735*   6751*   6755*   6759*   6771*
                        6774*    6789*   6857*   6861*
DACB    001326          5775#    5944    6180*   6354*   6409*   6419*   6424*   6715*   6727*   6736*   6752*   6756*   6760*
                        6772*    6775*   6858*   6862*
DAC0    001344          5782#    6483*   6499*
DAC1    001346          5783#    6515*   6529*
DAC2    001350          5784#    6545*   6559*
DAC3    001352          5785*    6574*   6588*
DDISP = 177570          5692#    5731    5839
DF1     013544          5748     5755    5761    5767    7031#
DH1     013335          5746     7023#
DH2     013375          5765     7024#
DH3     013454          5753     5759    7025#
DISPLA  001142          5731#    5839*   7043*   7044*
DISPRE  000174          5699#    5839
DONE    011101          6942     6979#
DSWR  = 177570          5692#    5731    5839
DT1     013504          5747     7028#
DT2     013516          5766     7029#
DT3     013534          5754     5760    7030#
DUMC    010164          6044     6051    6060    6063    6938#
DUMW    010112          6032     6054    6928#
ECHAN   011230          6985#
EMTVEC= 000030          5692#    5839*
EM1     013215          5745     7019#
EM2     013237          5752     7020#
EM3     013263          5758     7021#
EM4     013310          5764     7022#
ERRVEC= 000004          5692#    5839*   6853*   7043*
FIXONE  007506          5849     6853#
GAIN00= 000000          5722#    6320
GAIN01= 000004          5723#    6328
GAIN10= 000010          5724#    6336
GAIN11= 000014          5725#    6344
```

```
GCHAN    011162           6617     6703     6984#
GMSG     011054           6975#
GNS    = ******  U        5699     5848     7057     7058     7059
GTSWR  = 104407           5848     7057#
HEAD2    013144           7018#
HEAD5    011443           6989#
HT     = 000011           5692#    7048
IOTST1   006306           5923     6608#
IOTST2   006462           5924     6641#
IOTST3   006664           5925     6698#
IOTST4   006772           5926     6721#
IOTST5   007062           5927     6746#
IOTST6   007150           5928     6767#
IOTST7   007216           5929     6781#
IOTVEC= 000020            5692#    5839*
KWAD     001374           5794#    5857     6079     6125     6828*
KWBPR    001342           5781#    6083*    6096*    6128*
KWCSR    001340           5780#    6084*    6097*    6131*    6132
KWEX     001376           5795#    5862     6092     6831*
LF     = 000012           5692#    7048
LKVEC  = 000100           5692#
MAEX     001400           5796#    5859*    5868     6107
METST    011463           6932     6990#
MSADV    012147           5876     7001#
MSBTEX   012054           5872     6999#
MSGNEX   012016           6113     6997#
MSG70    012345           5889     7006#
MSG71    011377           5890     6988#
MSIO1    010376           6609     6966#
MSIO2    010454           6641     6967#
MSIO3    010514           6698     6968#
MSIO4    010557           6721     6969#
MSIO5    010632           6746     6970#
MSIO6    010677           6767     6971#
MSIO7    010770           6781     6972#
MSKWAD   011505           5823     5856     6991#
MSKWEX   011567           5861     6993#
MSMAEX   011676           5867     6995#
MSTC1    012176           5880     7002#
MSTC2    012255           5884     7004#
ODTST  = 170000           5692#
OKCHAR   002434           5900     5911#
ONAD     011336           6934     6987#
OTHER    010044           6320*    6328*    6336*    6344*    6619*    6620*    6621*    6622*    6705*    6706*    6707*    6708*    6886*
                          6890     6911#
PIRQ   = 177772           5692#
PIRQVE= 000240            5692#
POINT  = 017536           7064#
PR0    = 000000           5692#
PR1    = 000040           5692#
PR2    = 000100           5692#
PR3    = 000140           5692#
PR4    = 000200           5692#
PR5    = 000240           5692#
PR6    = 000300           5692#    5839     7040
PR7    = 000340           5692#
```

```
PS     = 177776         5692#
PSW    = 177776         5692#
PWRVEC= 000024          5692#   5839*   7040*
QUEST    011077         5907    6978#
RDCHR  = 104411         7034    7057#
RDLIN  = 104412         5824    5892    6115    6753    6757    6761    7037    7057#
RDOCT  = 104413         6614    6618    6700    6704    7057#
RESVEC= 000010          5692#
RETERR   004102         6168    6170#
RETURN   001440         5814#   6896
RST      010214         6946#
SLASH    011120         6982#
SPACE    011072         6628    6677    6797    6977#
SPREAD   001364         5790#   6915*   6921    7029
STACK  = 001100         5692#   5839
STALL    001426         5807#   6005    6085    6098    6149    6773    6776
STKLMT= 177774          5692#
STREG    001316         5771#   5941    5988*   5991    5996*   6003*   6006*   6014*   6015    6023*   6024    6038*   6039
                        6041    6056*   6057    6064*   6068*   6069    6072*   6082*   6089*   6095*   6102*   6112*   6122*
                        6129*   6134*   6136    6148*   6152*   6165*   6166    6381*   6410*   6440*   6484*   6516*   6546*
                        6575*   6608*   6672*   6673*   6674    6786*   6787    6854*   6892*   6898*   6935    7028    7029
                        7030
SWR      001140         5731#   5839*   5848    7034*   7040*   7043    7044
SWREG    000176         5699#   5839    5848    7034
SW0    = 000001         5692#
SW00   = 000001         5692#
SW01   = 000002         5692#
SW02   = 000004         5692#
SW03   = 000010         5692#
SW04   = 000020         5692#
SW05   = 000040         5692#
SW06   = 000100         5692#
SW07   = 000200         5692#
SW08   = 000400         5692#
SW09   = 001000         5692#
SW1    = 000002         5692#
SW10   = 002000         5692#
SW11   = 004000         5692#
SW12   = 010000         5692#
SW13   = 020000         5692#
SW14   = 040000         5692#
SW15   = 100000         5692#
SW2    = 000004         5692#
SW3    = 000010         5692#
SW4    = 000020         5692#
SW5    = 000040         5692#
SW6    = 000100         5692#
SW7    = 000200         5692#
SW8    = 000400         5692#
SW9    = 001000         5692#
TBITVE= 000014          5692#
TC1      001366         5791#   5881    6192    6225    6251    6277    6301    6318*   6375    6404    6834*
TC2      001370         5792#   5885    6438    6480    6512    6542    6571    6837*
TEMP     001360         5788#   5836*   5838*   5850    6229    6230    6255    6256    6281    6282    6305    6629    6713*
                        6714    6715    6891*   6902*   6905*   6906*   6907*   6908*   6916
TEST     004060         6166#   7058
```

G 7

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 97-5
CNAXAA.P11   23-DEC-82 14:21   CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0084

```
TESTIT 004052          6165#   7059
TKVEC = 000060         5692#   7034*
TPVEC = 000064         5692#
TRAPVE= 000034         5692#   5839*
TRTVEC= 000014         5692#
TRYAG  002356          5892#   5908
TST1   002530          5939#
TST10  002716          5977#
TST11  002732          5982#
TST12  002746          5986#
TST13  003012          5992    5995#
TST14  003044          6001#
TST15  003106          6012#
TST16  003150          6022#
TST17  003174          6031#
TST2   002564          5945#
TST20  003322          6045    6053#   6061
TST21  003406          6067#
TST22  003450          6078#
TST23  003530          6080    6091#
TST24  003610          6093    6106#
TST25  003676          6108    6110    6124#
TST26  004002          6126    6144#
TST27  004046          6146    6156#
TST3   002614          5953#
TST30  004104          6177#
TST31  004166          6191#
TST32  004304          6193    6224#
TST33  004404          6226    6250#
TST34  004504          6252    6276#
TST35  004604          6278    6300#
TST36  004656          6302    6317#
TST37  005024          6319    6353#
TST4   002630          5957#
TST40  005070          6356    6371#
TST41  005220          6376    6378    6400#
TST42  005350          6405    6407    6435#
TST43  005500          6439    6477#
TST44  005642          6481    6502    6509#
TST45  006000          6513    6532    6539#
TST46  006136          6543    6562    6569#
TST47  006274          6572    6591    6594#
TST5   002652          5963#
TST6   002666          5967#
TST7   002702          5973#
TYPBN = 104406         7057#
TYPDS = 104405         6962    7057#
TYPE  = 104401         5821    5822    5848    5889    5890    5907    6113    6114    6609    6613    6617    6623    6628
                       6635    6641    6646    6649    6655    6661    6667    6677    6691    6698    6699    6703    6721
                       6746    6767    6781    6783    6797    6932    6934    6942    6962    7034    7040    7044    7045
                       7048    7052    7053    7054    7057#   7064
TYPOC = 104402         7034    7045    7057#
TYPON = 104404         7057#
TYPOS = 104403         6624    6629    6652    6679    6791    6933    6935    7057#
UNEXP  001404          5799#   5958    6047
VECTOR 001330          5776#   5958*   6036*   6047*   6863*   6864*   6865    6867    6869    6896*
```

```
VECTR1  001332        5777#   6037*   6865*   6866*
VECTR2  001334        5778#   6055*   6867*   6868*
VECTR3  001336        5779#   6869*   6870*
VWRAP   001354        5786#   6198    6205    6212    6219    6325    6333    6341    6349    6387    6416    6493    6523
                      6553    6582
V12     010240        6955#
V2      010236        6236    6244    6262    6270    6288    6296    6311    6361    6446    6453    6460    6467    6954#
WRAP    004104        6177#   6815    6821
$APTHD  001000        5730#
$ASTAT= ****** U      7049
$ATYC   016474        7049#
$ATY1   016450        7049#
$ATY3   016456        7048    7049#
$ATY4   016466        7044    7049#
$AUTOB  001134        5731#   5848*   5852    7034
$BASE   001250        5731#   6854    6855    6856    6857    6858
$BDADR  001122        5731#
$BDDAT  001126        5731#   5991*   6041*   6057*   6136*   6137    6166*   6167    6916*   6918    7028    7029    7030
$BELL   001164        5731#   7034    7044
$BIN    017216        7053#*
$CDW1   001254        5731#
$CHARC  016444        7048#*
$CKSWR  014102        7034#   7057
$CLKVE  017536        7064#
$CMTAG  001100        5731#   5839
$CM3  = 000000        5731#
$CNTLC  015030        7034#
$CNTLG  015042        7034#
$CNTLU  015035        7034#
$CPUOP  001222        5731#
$CRLF   001171        5731#   5821    6635    6646    6691    6783    7034    7044    7045    7048
$DBLK   017434        7054#
$DEVCT  001204        5731#
$DEVM   001252        5731#   6826    6829    6832    6835    6838    6841
$DOAGN  010352        6962#
$DTBL   017424        7054#
$ENDAD  010342        5728    5848    6962#   7044
$ENDCT  010310        5839    6962#
$ENDMG  010361        6962#
$ENULL  010356        6962#
$ENV    001214        5731#   5848    7044    7048    7049
$ENVM   001215        5731#   5839    7048    7049
$EOP    010254        6810    6817    6823    6962#
$EOPCT  010302        5839*   6962#   7043*   7044*
$ERFLG  001103        5731#   5800*
$ERMAX  001115        5731#   5839*   7043*
$ERROR  015632        5839    7044#
$ERRPC  001116        5731#   7028    7029    7030    7044*   7045
$ERRTB  001256        5731#   7045
$ERRTY  016032        7044    7045#
$ERTTL  001112        5731#   7044*
$ESCAP  001162        5731#   5799*   5802*   5839*   7043*   7044
$ETABL  001214        5731#
$ETEND  001256        5730    5731#
$FATAL  001176        5731#   7049*
$FFLG   016714        7049#*
```

I 7

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 97-7
CNAXAA.P11   23-DEC-82 14:21   CROSS REFERENCE TABLE -- USER SYMBOLS   SEQ 0086

```
$FILLC  001156          5731#   7048
$FILLS  001155          5731#   7048
$GDADR  001120          5731#
$GDDAT  001124          5731#   5946*   5949*   5950    5954*   5959*   5964*   5968*   5974*   5978*   5983*   5987*   6004*
                        6013*   6042*   6058*   6071-   6081*   6082    6094*   6095    6111*   6112    6127*   6137    6147*
                        6148    6165    6167    6914*   6917    7028    7029
$GET42  010332          6962#
$GTSWR  014172          7034#   7057
$HD   = 000000          5691
$HIBTS  001000          5730#
$HIOCT  015172          7037#*
$ICNT   001104          5731#   7043*
$ILLUP  015334          7040#
$INTAG  001135          5731#   7034*
$ITEMB  001114          5731#   7044*   7045
$LF     001172          5731#   7034    7044    7048
$LFLG   016713          7049#*
$LPADR  001106          5731#   5839*   5939*   6182*   6931*   6941*   7043*
$LPERR  001110          5731#   5839*   6181*   6930*   6940*   7043*   7044
$MADR1  001226          5731#
$MADR2  001232          5731#
$MADR3  001236          5731#
$MADR4  001242          5731#
$MAIL   001174          5730    5731#   5839    5848    7043    7044    7048
$MAMS1  001224          5731#
$MAMS2  001230          5731#
$MAMS3  001234          5731#
$MAMS4  001240          5731#
$MBADR  001002          5730#
$MFLG   016712          7049#*
$MNEW   015060          7034#
$MSGAD  001210          5731#   7049*
$MSGLG  001212          5731#   7049*
$MSGTY  001174          5731#   7049*
$MSWR   015047          7034#
$MTYP1  001225          5731#
$MTYP2  001231          5731#
$MTYP3  001235          5731#
$MTYP4  001241          5731#
$MXCNT  015630          7043#
$NULL   001154          5731#   7048
$NWTST= 000001          5939#   5945#   5953#   5957#   5963#   5967#   5973#   5977#   5982#   5986#   5995#   6001#   6012#
                        6022#   6031#   6053#   6067#   6078#   6091#   6106#   6124#   6144#   6156#   6177#   6191#   6224#
                        6250#   6276#   6300#   6317#   6353#   6371#   6400#   6435#   6477#   6509#   6539#   6569#   6594#
$OCNT   017140          7052#*
$OMODE  017142          7052#*
$OVER   015614          7043#
$PASS   001202          5731#   5839*   6109    6928    6938    6962*   7043
$PASTM  001006          5730#
$POWER  015342          7040#
$PWRDN  015174          5839    7040#
$PWRMG  015330          7040#
$PWRUP  015246          7040#
$QUES   001170          5731#   7034    7044    7048
$RDCHR  014444          7034#   7057
$RDDEC= ****** U        7057
```

J 7

MAINDEC-11-CNAXA-A   MACY11 30(1046) 23-DEC-82 14:22 PAGE 97-8
CNAXAA.P11   23-DEC-82 14:21   CROSS REFERENCE TABLE -- USER SYMBOLS   SEQ 0087

```
$RDLIN  014534        7034#  7057
$RDOCT  015072        7037#  7057
$RDSZ = 000040        7034#
$RTNAD  010354        6962#
$R2A  = ****** U       7057
$SAVRE= ****** U       7057
$SAVR6  015340        7040#*
$SCOPE  015352        5839   7043#
$SETUP= 000137        5732#  5839   5848   6962   7034   7043   7044
$STUP = 177777        5732#
$SVLAD  015560        7043#
$SVPC = 000106        5728#
$SWR  = 167400        5683#  5691   5693   5731   5839   5939   5945   5953   5957   5963   5967   5973   5977
                      5982   5986   5995   6001   6012   6022   6031   6053   6067   6078   6091   6106   6124
                      6144   6156   6177   6191   6224   6250   6276   6300   6317   6353   6371   6400   6435
                      6477   6509   6539   6569   6594   6962   7040   7043   7044
$SWREG  001216        5731#  5839
$SWRMK= 000000        5693   7043
$TESTN  001200        5731#  7043*
$TIMES  001160        5731#  5839*  5986*  5995*  6012*  6177*  6191*  6224*  6250*  6276*  6300*  6317*  6353*
                      6371*  6400*  6435*  6477*  6509*  6539*  6569*  6594*  6962*  7043*
$TKB    001146        5731#  7034
$TKCNT  013546        7034#*
$TKINT  013614        5844   7034#
$TKQEN= 013614        7034#
$TKQIN  013550        7034#*
$TKQOU  013552        7034#*
$TKQSR  013554        7034#
$TKS    001144        5731#  5893*  5990*  5998*  6020*  6947*  6957*  7034*
$TKSRV  013664        7034#
$TN   = 000050        5684#  5691   5939#  5940   5945#  5953#  5957#  5963#  5967#  5973#  5977#  5982#  5986#
                      5992   5995#  6001#  6012#  6022#  6031#  6032   6045   6053#  6054   6067#  6078#  6080
                      6091#  6093   6106#  6108   6110   6124#  6126   6144#  6146   6156#  6177#  6191#  6193
                      6224#  6226   6250#  6252   6276#  6278   6300#  6302   6317#  6319   6353#  6356   6371#
                      6376   6378   6400#  6405   6407   6435#  6439   6477#  6481   6502   6509#  6513   6532
                      6539#  6543   6562   6569#  6572   6591   6594#
$TPB    001152        5731#  7048*
$TPFLG  001157        5731#  7048
$TPS    001150        5731#  7048
$TRAP   017444        5839   7057#
$TRAP2  017466        7057#
$TRP  = 000016        7057#  7058#  7059#
$TRPAD  017500        7057#
$TSTM   001004        5730#
$TSTNM  001102        5731#  5940*  6177*  6962*  7043*  7044
$TTYIN  014770        7034#
$TYPBN  017144        7053#  7057
$TYPDS  017220        7054#  7057
$TYPE   016166        5840*  5841*  5842*  5843*  7048#  7049   7057
$TYPEC  016400        7034   7048#
$TYPEX  016446        7048#
$TYPOC  016742        7052#  7057
$TYPON  016756        7052#  7057
$TYPOS  016716        7052#  7057
$UNIT   001206        5731#
$UNITM  001010        5730#
```

```
$USWR   001220          5731#
$VECT1  001244          5731#    6863
$VECT2  001246          5731#
$XTSTR  015364          7043#
$$GET4= 000000          6962#
$OFILL  017141          7052#*
$40CAT= ****** U        7043     7044
.    = 017610           5699#    5702#    5728#    5729#    5730#    5731#    5839    6962    7034#    7040    7043    7044    7045#
                        7048     7049#    7054#    7064#
.$ASTA= ****** U        7049
.$X  = 001000           5730#
```

L 7

MAINDEC-11-CNAXA-A   MACY11 30(1046)   23-DEC-82 14:22 PAGE 98
CNAXAA.P11   23-DEC-82 14:21   CROSS REFERENCE TABLE -- MACRO NAMES   SEQ 0089

```
COMMEN  1566#   5692#
DUMWRN  5736#   6032    6054
ENDCOM  1578#   5692#
ERROR   5692#   5801    5948    5956    5961    5966    5970    5976    5980    5985    5993    6000    6008    6019    6028
        6043    6059    6074    6087    6100    6120    6139    6151    6199    6206    6213    6220    6237    6245    6263
        6271    6289    6297    6312    6326    6334    6342    6350    6362    6394    6423    6447    6454    6461    6468
        6504    6534    6564    6593
ESCAPE  1698#   5692#   5799
GETPRI  1312#   5692#
GETSWR  1771#   5692#   5848#
MULT    4484#   5692#
NEWTST  1626#   5692#   5939    5945    5953    5957    5963    5967    5973    5977    5982    5986    5995    6001    6012
        6022    6031    6053    6067    6078    6091    6106    6124    6144    6156    6177    6191    6224    6250    6276
        6300    6317    6353    6371    6400    6435    6477    6509    6539    6569    6594
POP     2153#   5692#   7037    7040    7049    7054
PUSH    2145#   5692#   7037    7040    7049    7054
REPORT  5463#   5692#
SCOPE   5692#   5945    5953    5957    5963    5967    5973    5977    5982    5986    5995    6001    6012    6022    6031
        6053    6067    6078    6091    6106    6124    6144    6156    6191    6224    6250    6276    6300    6317    6353
        6371    6400    6435    6477    6509    6539    6569    6594
SETPRI  1279#   5692#   7034
SETTRA  7057#   7C58    7059
SETUP   1337#   5692#   5839
SKIP    1733#   5692#   5951    5992    6045    6080    6093    6108    6110    6126    6138    6146    6168    6193    6226
        6252    6278    6302    6319    6356    6376    6378    6393    6405    6407    6422    6439    6481    6502    6513
        6532    6543    6562    6572    6591
SLASH   1517#   5692#
SPACE   5692#
STARS   1485#   5692#   5728    5730    5731    5939    5945    5953    5957    5963    5967    5973    5977    5982    5986
        5995    6001    6012    6022    6031    6053    6067    6078    6091    6106    6124    6156    6177    6191
        6224    6250    6276    6300    6317    6353    6371    6400    6435    6477    6509    6539    6569    6594
        7034    7037    7040    7044    7045    7048    7049    7052    7053    7054    7057              6962
SWRSU   1453#   5692#   5839#
TRMTRP  7057#
TYPBIN  2088#   5692#
TYPDEC  2058#   5692#   6962
TYPNAM  1826#   5692#   5848
TYPNUM  2025#   5692#
TYPOCS  1978#   5692#   6624    6629    6933    6935
TYPOCT  1941#   5692#   7034    7045
TYPTXT  1894#   5692#
$$CMRE  5731#
$$CMTM  5731#
$$ESCA  1711#   5692#
$$NEWT  1662#   5692#   5939    5945    5953    5957    5963    5967    5973    5977    5982    5986    5995    6001    6012
        6022    6031    6053    6067    6078    6091    6106    6124    6144    6156    6177    6191    6224    6250    6276
        6300    6317    6353    6371    6400    6435    6477    6509    6539    6569    6594
$$SET   7057#   7058    7059
$$SETM  5839#
$$SKIP  1746#   5692#   5992    6045    6080    6093    6108    6110    6126    6146    6193    6226    6252    6278    6302
        6319    6356    6376    6378    6405    6407    6439    6481    6502    6513    6532    6543    6562    6572    6591
.EQUAT   189#   5685#   5692
.HEADE    61#   5685#   5691
.INIT   5658#   5692#   7064
.SETUP  1213#   5687#   5732
.SWRHI   104#   5687#   5693
```

```
.SWRLO   5693#
.$ACT1   5064#   5688#   5728
.$APTB   5109#   5688#   5731#
.$APTH   5370#   5688#   5730
.$APTY   5547#   5688#   7049
.$ASTA   5417#
.$CATC    932#   5685#   5699
.$CMTA   1047#   5685#   5731
.$DB2D   4686#
.$DB2O   4812#
.$DIV    4587#
.$EOP    2214#   5685#   6962
.$ERRO   2700#   5685#   7044
.$ERRT   2896#   5687#   7045
.$MULT   4523#
.$PARM   5686#
.$POWE   4229#   5686#   7040
.$RAND   4307#   5688#
.$RDDE   3891#
.$RDOC   3797#   5688#   7037
.$READ   3395#   5686#   7034
.$R2AZ   4958#
.$SAVE   3969#   5686#
.$SB2D   4771#
.$SB2O   4874#
.$SCOP   2454#   5686#   7043
.$SIZE   4361#
.$SPAC   5687#
.$SUPR   4913#
.$SWDO   5687#
.$TRAP   4073#   5687#   7057
.$TYPB   3287#   5686#   7053
.$TYPD   3209#   5688#   7054
.$TYPE   2985#   5687#   7048
.$TYPO   3112#   5686#   7052
.$40CA    972#


. ABS.  017610      000


ERRORS DETECTED:  0

CNAXAA,CNAXAA/CRF/NL:TOC=CNMAC2.SML,CNAXAA.P11
RUN-TIME: 19 15 1 SECONDS
RUN-TIME RATIO: 108/35=3.0
CORE USED:  33K  (66 PAGES)
```