

The microfiche card displays a grid of 48 frames of performance test data. Each frame contains a small table or chart with various numerical values and labels, representing different test runs or configurations. The data is organized into a structured grid, with each frame containing a small table or chart. The frames are arranged in 8 rows and 6 columns. The data in the frames is too small to read accurately, but it appears to be a series of performance metrics or test results.

IDENTIFICATION

SEQ 0001

Product Code: AC-8174B-MC
Product Name: CVADABO - ADV11 Performance Test
Date: July 1978
Maintainer: Diagnostic Group

Copyright (C) 1976,1978

Digital Equipment Corporation, Maynard, Mass.

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

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

The following are trademarks of Digital Equipment Corporation:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

1.0 ABSTRACT

This diagnostic has two starting addresses:

- 200 standard tolerances
- 204 restart
- 210 tighter tolerances for the option test area's burn in.

This diagnostic tests the ADV11 with or without the BERG test connector.

When starting the diagnostic, a set of tests is listed and this statement is printed out: "Type the letter and carriage return of the desired test:". The following chart indicates which letter corresponds to which test:

W: The entire Wraparound test (requires BERG test connector)

- a. Analog subtests
- b. Noise test
- c. Interchannel Settling test
- d. Differential Linearity and Relative Accuracy test

C: Calibration test only

P: Print values test only

L: Logic Subtests only

A: Auto test (requires BERG test connector)

- A. Logic subtests
- B. Analog subtests
- C. Noise Test
- D. Interchannel Settling Test
- E. Differential Linearity and Relative Accuracy Test

2.0 REQUIREMENTS

2.1 Equipment

LSI-11 computer with 8K of memory
I/O Terminal
ADV11 Module
VT55 Terminal supported for graphic output
BERG test connector

2.2 Storage

This program uses all 8K of memory and is not 'chainable' on an 8K CPU. The program is 'chainable' on 12K or greater CPU. The program will destroy 'absolute loader' on an 8K CPU, if 'W' or 'A' is selected.

3.0 LOADING PROCEDURE

Procedure for loading normal binary tapes should be followed.

4.0 STARTING PROCEDURE

4.1 Control Switch Settings

Standard PDP-11 Format

SW15=1	Halt on error
SW14=1	Loop on test
SW13=1	Inhibit error typeouts
SW12=1	Halt for VT55 display
SW11=1	Inhibit iterations
SW10=1	Bell on error
SW9 =1	Loop on error
SW8 =1	Loop on test in SWR <7:0>

200 is the starting address of the diagnostic for standard tolerances. 204 is the restart address. 210 is the starting address of the diagnostic for the option test area's burn in test.

5.0 OPERATING PROCEDURE

Start the diagnostic at 200 or 210. The program heading and the list of tests available, will be printed out followed by a message 'Type letter and <CR> for test:'. Then type the letter you want, according to the table listed and depress return. If started at the option test area's starting address, the program will not ask for the test but will run the logic test.

Two control characters, ^A and ^C, are set aside for interrupting a test and transferring control to either the beginning of the diagnostic (^C) or to the beginning of the specific test which was in progress (^A). During the logic tests while a reset is being performed, ^C or ^A will not be executed until after the reset has been completed, therefore continue typing ^C or ^A until it is successful.

For machines without a hardware switch register, 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 type 'XX ADV11's FOUND'. Where XX is the number of ADV11's in octal. If the number is greater than 1, the test will be run successively on each ADV11. The program will run through the analog subtests, the Noise test, the Interchannel Settling test, and the Differential Linearity and Relative Accuracy test. The BERG test connector is required.

If 'C' is typed, the program will run the calibration routine and loop on the test until it is calibrated and a carriage return typed. If a certain ADV11 is to be calibrated, its status register address must be loaded into \$BASE (1250), and its vector address must be loaded into the low byte of \$VECT1 (1244).

If 'P' is typed, the program will run the print values routine and will loop on that test until the operator halts it. If a certain ADV11 is to be tested, its status register address must be loaded into \$BASE (1250), and its vector address must be loaded into the low byte of \$VECT1 (1244).

If 'A' is typed, the program will execute the logic tests, analog tests, noise, settle and differential linearity. At the beginning of the test the program will type 'XX ADV11'S FOUND'. Where XX is the number of ADV11's in octal. If the number is greater than 1 the test will run successively on each ADV11.

If 'L' is typed, the program will execute the logic tests, printing 'END PASS' when it has completed an entire pass. At the beginning of the test the program will type 'XX ADV11'S FOUND'. Where XX is the number of ADV11's in octal. If the number is greater than 1, the test will be run successively on each ADV11.

5.1 Inhibiting Auto-Size Feature

This program will automatically auto-size and test each ADV11 it detects on the system. To inhibit this feature, set bit 15 of location \$ENVM (1214). Also, load location \$BASE (1250) with the ADV11's status register address and the low byte of location \$VECT1 (1244) with the ADV11's vector address.

5.2 End of Pass Typeouts

At end of pass, the following typeout will occur:

```
'ENDPASS GOOD UNITS 0000000000000011
```

This indicates that units 1 and 2 have run without failure.

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:

Calibration:	5 conversions/min @110 baud
Print Values:	8 conversions/8 seconds @ 110 baud
Wraparound Test:	7 minutes first pass; 25 minutes for successive passes
Logic Test:	1 minute
Auto Test:	8 minutes first pass, 26 minutes for successive passes

7.2 Status Register and Vector Addresses and Priority

When testing more than one ADV11, the difference in addresses is 4 for bus address and 10 for vector address. These values are in VADR (bus address) (1332) and VVCT (vector address) (1334). The first ADV11's status register address must be in \$BASE (1250), its vector address must be in the low byte of \$VECT1 (1244).

7.3 Switch Register

If a hardware switch register is present and the operator desires to use a software switch register and the ^G feature; it is necessary to load the starting address, set the hardware switch register to all ones (-1), and depress start. The program will then run with the software switch register.

7.4 VT55 Graphic Output

The screen display may be halted for examination by setting bit 12 of the switch register. Then, type 'P' to complete the program's execution.

8.0 RESTRICTIONS

8.1 Testing

The BERG Test Connector 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 two tests of this program check to see if the ADV11 responds to the expected address. If the ADV11 does not respond, a buss error occurs. Also bus errors can occur during the time the program sizes to see how many ADV11's are on your system.

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 Tests

These 23 logic subtests run sequentially without further operator intervention. Its 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 external start enable bit, the done bit, the interrupt enable bit, the overflow 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 interrupt logic and the correct setting of the error flag.

9.2 Calibration Routine

If 'C' is typed, the program will ask for a channel. Type channel number followed by a carriage return. The program will ask you if you want offset or gain. Apply voltage requested to selected channel. Adjust pot requested for 0.00 SB typeout. Type carriage return when adjusted. The last typeout will be checked for 0.00 LSB with a tolerance of 0.04 LSB if outside, the program will ask you to adjust the same pot again.

9.3 Print Values Routine

This test begins when the operator types 'P'. It then loads the channel from the switch register bits 0-7 and does a conversion on that channel. If SWR bit 13 is down (0), it prints out the converted value on the teletype; otherwise, if SWR bit 13 is up (1), it puts the converted value in the display register. The operator may change the channel at any time during the test, however the new values from the new channel will not be printed until the next line of 8 values is printed. The 8 values on each line correspond to only one channel.

9.4 Differential Linearity

This test determine if a change in the input voltage represents a similar change in the resulting converted binary value, by measuring the width of each state correct to 0.01 LSB.

9.5 Settling Test

The purpose of this test is to check that the time needed to settle and correctly report a new input value after switching channels does not exceed the expected amount of time for such a change.

9.6 Noise Test

This test measures the internal short-term repeatability noise within the A/D. RMS noise equals 1 standard deviation of the Gaussian curve, PEAK noise equals 2.3 standard deviation of the Gaussian curve.

9.7 Analog Tests

These 6 subtests check the channels and their output.

21	BASIC DEFINITIONS
22	OPERATIONAL SWITCH SETTINGS
30	TRAP CATCHER
(1)	STARTING ADDRESS(ES)
34	ACT11 HOOKS
36	APT PARAMETER BLOCK
37	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
75	MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
125	CONTROL A AND C DECODERS
158	INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE
163	INITIALIZE THE COMMON TAGS
168	DETERMINE IF VT55 TYPE TERMINAL IS PRESENT
184	DIALOGUE TO DETERMINE WHICH TEST TO RUN
270	T1 FLOAT A ONE THRU MULTIPLEXER BITS
279	T2 LOAD AND READ BACK ERROR I.E. BIT14
283	T3 LOAD AND READ BACK INTERRUPT ENABLE BIT6
289	T4 LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BITS
294	T5 LOAD AND READ BACK EXTERNAL START ENABLE BIT4
298	T6 LOAD AND READ BACK MAINT. TST BIT2
303	T7 LOAD AND READ BACK ENABLE I.D. BIT3
307	T10 TEST I.D. BIT (BIT 12) CLEARED
315	T11 TEST I.D. BIT (BIT 12) SET
323	T12 LOAD AND READ BACK ERROR FLAG BIT15
327	T13 TEST INIT CLEARS BITS 2-6,8-11,14
337	T14 TEST INIT CLEARS ERROR FLAG
344	T15 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
355	T16 TEST INIT CLEARS DONE FLAG
365	T17 TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
372	T20 TEST ALL '0'S RESULTS USING MAINT. ADTST. BIT
383	T21 TEST ALL '1'S RESULT USING MAINT. ADTST. BIT
395	T22 GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
417	T23 TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET
430	T24 TEST ERROR FLAG SETS IF 2ND CONVERSION ENDS BEFORE READING BUFFER
443	T25 TEST ERROR FLAG SETS IF START 2ND CONV. BEFORE DONE FLAG SETS
468	WRAPAROUND TEST SECTION
470	T26 TEST CHO GROUND
479	T27 TEST CH1 +4.5 VOLT
487	T30 TEST CH2 -4.5 VOLT
494	T31 TEST GROUND ON CHANNELS 4 - 17
506	T32 TEST VERNIER OFFSET DAC ON CHO
519	T33 OFFSET ON CHO
534	T34 TEST RAMP RANGE, CH3
562	T35 NOISE TEST, 1 EDGE
590	T36 INTERCHANNEL SETTling TEST, 1 EDGE
611	T37 DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST
707	PRINT VALUES ROUTINE
742	LOGIC TEST SECTION
751	AUTO TEST
767	WRAPAROUND TEST
776	DETERMINE IF MORE ADV11'S TO BE TESTED
1358	END OF PASS ROUTINE
1360	ASCII MESSAGES
1455	TTY INPUT ROUTINE
1457	READ AN OCTAL NUMBER FROM THE ITY

1459	SCOPE HANDLER ROUTINE
1460	ERROR HANDLER ROUTINE
1461	ERROR MESSAGE TYPEOUT ROUTINE
1463	TYPE ROUTINE
1464	APT COMMUNICATIONS ROUTINE
1466	BINARY TO OCTAL (ASCII) AND TYPE
1468	BINARY TO ASCII AND TYPE ROUTINE
1470	TRAP DECODER
(3)	TRAP TABLE

```
20      .TITLE MAINDEC-11-DVADA-B
(1)      .: *COPYRIGHT (C) 1978
(1)      .: *DIGITAL EQUIPMENT CORP.
(1)      .: *MAYNARD, MASS. 01754
(1)      .: *
(1)      .: *PROGRAM BY GEORGE STEVENS
(1)      .: *
(1)      .: *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)      .: *PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)      .: *
21      .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 HDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)
(1)      .: *GENERAL PURPOSE REGISTER DEFINITIONS
(1)      000000 R0= %0      ;;GENERAL REGISTER
(1)      000001 R1= %1      ;;GENERAL REGISTER
(1)      000002 R2= %2      ;;GENERAL REGISTER
(1)      000003 R3= %3      ;;GENERAL REGISTER
(1)      000004 R4= %4      ;;GENERAL REGISTER
(1)      000005 R5= %5      ;;GENERAL REGISTER
(1)      000006 R6= %6      ;;GENERAL REGISTER
(1)      000007 R7= %7      ;;GENERAL REGISTER
(1)      000006 SP= %6      ;;STACK POINTER
(1)      000007 PC= %7      ;;PROGRAM COUNTER
(1)
(1)      .: *PRIORITY LEVEL DEFINITIONS
(1)      000000 PR0= 0      ;;PRIORITY LEVEL 0
(1)      000040 PR1= 40     ;;PRIORITY LEVEL 1
(1)      000100 PR2= 100    ;;PRIORITY LEVEL 2
(1)      000140 PR3= 140    ;;PRIORITY LEVEL 3
(1)      000200 PR4= 200    ;;PRIORITY LEVEL 4
(1)      000240 PR5= 240    ;;PRIORITY LEVEL 5
(1)      000300 PR6= 300    ;;PRIORITY LEVEL 6
(1)      000340 PR7= 340    ;;PRIORITY LEVEL 7
(1)
(1)      .: *'SWITCH REGISTER' SWITCH DEFINITIONS
(1)      100000 SW15= 100000
(1)      040000 SW14= 40000
(1)      020000 SW13= 20000
(1)      010000 SW12= 10000
```

```

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

```

(1)      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)      100000      BIT15= 100000
(1)      040000      BIT14= 40000
(1)      020000      BIT13= 20000
(1)      010000      BIT12= 10000
(1)      004000      BIT11= 4000
(1)      002000      BIT10= 2000
(1)      001000      BIT09= 1000
(1)      000400      BIT08= 400
(1)      000200      BIT07= 200
(1)      000100      BIT06= 100
(1)      000040      BIT05= 40
(1)      000020      BIT04= 20
(1)      000010      BIT03= 10
(1)      000004      BIT02= 4
(1)      000002      BIT01= 2
(1)      000001      BIT00= 1
(1)      .EQUIV      BIT09,BIT9
(1)      .EQUIV      BIT08,BIT8
(1)      .EQUIV      BIT07,BIT7
(1)      .EQUIV      BIT06,BIT6
(1)      .EQUIV      BIT05,BIT5
(1)      .EQUIV      BIT04,BIT4
(1)      .EQUIV      BIT03,BIT3
(1)      .EQUIV      BIT02,BIT2
(1)      .EQUIV      BIT01,BIT1
(1)      .EQUIV      BIT00,BIT0
  
```

```

(1)      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)      000004      ERRVEC= 4            ;;TIME OUT AND OTHER ERRORS
(1)      000010      RESVEC= 10          ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)      000014      TBITVEC=14         ;;'T' BIT
(1)      000014      TRTVEC= 14         ;;TRACE TRAP
  
```

```

(1)      000014      BPTVEC= 14      ;;BREAKPOINT TRAP (BPT)
(1)      000020      IOTVEC= 20      ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)      000024      PWRVEC= 24      ;;POWER FAIL
(1)      000030      EMTVEC= 30      ;;EMULATOR TRAP (EMT) **ERROR**
(1)      000034      TRAPVEC=34      ;;'TRAP' TRAP
(1)      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
(1)      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
(1)      000240      PIRQVEC=240    ;;PROGRAM INTERRUPT REQUEST VECTOR
22      .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)      :*          12          HALT FOR VT55 DISPLAY
(1)      :*          11          INHIBIT ITERATIONS
(1)      :*          10          BELL ON ERROR
(1)      :*          9          LOOP ON ERROR
(1)      :*          8          LOOP ON TEST IN SWR<7:0>
23      170400      ABASE= 170400
24      100400      AVECT1= 100400
25      000200      APRIOR= 200
26
27      000100      .=100
28 000100 000104 000200 000002      .WORD 104,200,2
29
30      .SBTTL TRAP CATCHER
(1)      .*=0
(1)      000000
(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)      .*=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 001644      JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
31 000204 000137 002262      JMP @#BEG2 ;RESTART ADDRESS
32 000210 000137 001652      JMP @#BEGIN2 ;START ADDRESS FOR OPTION TEST AREA

```

```
34 .SBTTL ACT11 HOOKS
(1)
(2) ::*****
(1) ;HOOKS REQUIRED BY ACT11
(1)          000214          $SVPC=.          ;SAVE PC
(1)          000046          .=46
(1) 000046 011764          $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)          000214          .= $SVPC          ;; RESTORE PC
35          001000          .=1000
36 .SBTTL APT PARAMETER BLOCK
(1)
(2) ::*****
(1) ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2) ::*****
(1)          001000          .$X=.          ;;SAVE CURRENT LOCATION
(1)          000024          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200          200          ;;FOR APT START UP
(1)          000044          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000          $APTHDR      ;;POINT TO APT HEADER BLOCK
(1)          001000          .=.$X          ;;RESET LOCATION COUNTER
(2) ::*****
(1) ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1) ;INTERFAE 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 000454          $TSTM: .WORD 300.      ;;RUN TIM OF LONGEST TEST
(1) 001006 000074          $PASTM: .WORD 60.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000454          $UNITM: .WORD 300.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000031          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```



```
37
(1)
(2)
(1)
(1)
(1)
(1)
(1) 001100 001100
(1) 001100 000000
(1) 001102 000
(1) 001103 000
(1) 001104 000000
(1) 001106 000000
(1) 001110 000000
(1) 001112 000000
(1) 001114 000
(1) 001115 001
(1) 001116 000000
(1) 001120 000000
(1) 001122 000000
(1) 001124 000000
(1) 001126 000000
(1) 001130 000000
(1) 001132 000000
(1) 001134 000
(1) 001135 000
(1) 001136 000000
(1) 001140 177570
(1) 001142 177570
(1) 001144 177560
(1) 001146 177562
(1) 001150 177564
(1) 001152 177566
(1) 001154 000
(1) 001155 002
(1) 001156 012
(1) 001157 000
(1) 001160 000000
(1) 001162 000000
(1) 001164 177607 000377
(1) 001170 077
(1) 001171 015
(1) 001172 000012
(2)
(2)
(2)
(3)
(2)
(2) 001174
(2) 001174 000000
(2) 001176 000000
(2) 001200 000000
(2) 001202 000000
(2) 001204 000000
(2) 001206 000000
(2) 001210 000000

.SBTTL COMMON TAGS
*****
*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
*USED IN THE PROGRAM.
.=1100
SCMTAG:          ;; START OF COMMON TAGS
                 .WORD 0
$STSTM: .BYTE 0 ;; CONTAINS THE TEST NUMBER
$ERFLG: .BYTE 0 ;; CONTAINS ERROR FLAG
$ICNT: .WORD 0 ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD 0 ;; CONTAINS SCOPE LOOP ADDRESS
$LPERR: .WORD 0 ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD 0 ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .BYTE 0 ;; CONTAINS ITEM CONTROL BYTE
$ERMAX: .BYTE 1 ;; CONTAINS MAX. ERRORS PER TEST
$ERRPC: .WORD 0 ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD 0 ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD 0 ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT: .WORD 0 ;; CONTAINS 'GOOD' DATA
$BDDAT: .WORD 0 ;; CONTAINS 'BAD' DATA
                 .WORD 0 ;; RESERVED--NOT TO BE USED
                 .WORD 0
$AUTOB: .BYTE 0 ;; AUTOMATIC MODE INDICATOR
$INTAG: .BYTE 0 ;; INTERRUPT MODE INDICATOR
                 .WORD 0
SWR: .WORD DSWR ;; ADDRESS OF SWITCH REGISTER
DISPLAY: .WORD DDISP ;; ADDRESS OF DISPLAY REGISTER
$TKS: 177560 ;; TTY KBD STATUS
$TKB: 177562 ;; TTY KBD BUFFER
$TPS: 177564 ;; TTY PRINTER STATUS REG. ADDRESS
$TPB: 177566 ;; TTY PRINTER BUFFER REG. ADDRESS
$NULL: .BYTE 0 ;; CONTAINS NULL CHARACTER FOR FILLS
$FILLS: .BYTE 2 ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC: .BYTE 12 ;; INSERT FILL CHARS. AFTER A 'LINE FEED'
$TPFLG: .BYTE 0 ;; 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
$TIMES: 0 ;; MAX. NUMBER OF ITERATIONS
$ESCAPE: 0 ;; ESCAPE ON ERROR ADDRESS
$BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
$QUES: .ASCII /?/ ;; QUESTION MARK
$CRLF: .ASCII <15> ;; CARRIAGE RETURN
$LF: .ASCIZ <12> ;; LINE FEED
*****
.SBTTL APT MAILBOX-ETABLE
*****
.EVEN
$MAIL:          ;; APT MAILBOX
$MSGTY: .WORD AMSGTY ;; MESSAGE TYPE CODE
$FATAL: .WORD AFATAL ;; FATAL ERROR NUMBER
$TESTN: .WORD ATESTN ;; TEST NUMBER
$PASS: .WORD APASS ;; PASS COUNT
$DEVCT: .WORD ADEVCT ;; DEVICE COUNT
$UNIT: .WORD AUNIT ;; I/O UNIT NUMBER
$MSGAD: .WORD AMSGAD ;; MESSAGE ADDRESS
```

(2)	001212	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
(2)	001214		\$ETABLE:		::APT ENVIRONMENT TABLE
(2)	001214	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
(2)	001215	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
(2)	001216	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
(2)	001220	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
(2)	001222	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
(2)			*		BITS 15-11=CPU TYPE
(2)			*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)			*		11/70=06,PDQ=07,Q=10
(2)			*		BIT 10=REAL TIME CLOCK
(2)			*		BIT 9=FLOATING POINT PROCESSOR
(2)			*		BIT 8=MEMORY MANAGEMENT
(2)	001224	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001225	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
(2)			*		MEM.TYPE BYTE -- (HIGH BYTE)
(2)			*		900 NSEC CORE=001
(2)			*		300 NSEC BIPOLAR=002
(2)			*		500 NSEC MOS=003
(2)	001226	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001230	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001231	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
(2)	001232	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001234	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001235	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
(2)	001236	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001240	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001241	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
(2)	001242	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001244	100400	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001246	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001250	170400	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001252	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
(2)	001254	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001256		\$ETEND:		
(2)			.MEXIT		

```

(1) .SBTTL ERROR POINTER TABLE
(1)
(1) :*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) :*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) :*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) :*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) :*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) :* EM ;;POINTS TO THE ERROR MESSAGE
(1) :* DH ;;POINTS TO THE DATA HEADER
(1) :* DT ;;POINTS TO THE DATA
(1) :* DF ;;POINTS TO THE DATA FORMAT
(1)
(1) $ERRTB:
(1) 001256
39
40
41
50 :ITEM 1
51 001256 014267 EM1 ;STATUS REG. ERROR
52 001260 014407 DH1 ;ERRPC STREG EXPECTED ACTUAL
53 001262 014566 DT1 ;$ERRPC, STREG, $GDDAT, $BDDAT
54 001264 014626 DF1
55
56
57 :ITEM 2
58 001266 014311 EM2 ;FAILED TO INTERRUPT
59 001270 014526 DH3 ;ERRPC STREG ACTUAL
60 001272 014616 DT3 ;$ERRPC, STREG, $BDDAT
61 001274 014626 DF1
62
63 :ITEM 3
64 001276 014335 EM3 ;UNEXPECTED INTERRUPT
65 001500 014526 DH3 ;ERRPC STREG
66 001302 014616 DT3 ;$ERRPC, STREG
67 001304 014626 DF1
68
69 :ITEM 4
70 001306 014362 EM4 ;ERROR ON A/D CHANNEL
71 001310 014443 DH2 ;ERRPC STREG CHAN NOMINAL TOL ACTUAL
72 001312 014600 DT2 ;$ERRPC, STREG, CHANL, $GDDAT, SPREAD, $BDDAT
73 001314 014626 DF1
    
```

```
75 .SBTTL MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
76 001316 170400 STREG: ABASE ;ADDRESS OF STATUS REGISTER
77 001320 170401 ADST1: ABASE+1 ;UPPER BYTE OF STATUS REG.
78 001322 170402 ADBUFF: ABASE+2 ;ADDRESS OF A/D BUFFER
79 001324 100400 VECTOR: AVECT1 ;VECTOR ADDRESS
80 001326 000200 BASEBR: APRIOR ;INTERRUPT PRIORITY LEVEL
81 001330 100402 VECTR1: AVECT1+2
82 001332 100404 VECTR2: AVECT1+4 ;ERROR VECTOR ADDRESS
83 001334 100406 VECTR3: AVECT1+6
84 001336 000004 VADR: 4 ;INCREMENT FOR BUS ADDRESS
85 001340 000010 VVCT: 10 ;INCREMENT FOR VECTOR ADDRESS
86 001342 000000 BASECH: 0 ;BASE CHANNEL
87 001344 000060 KBVECT: 60
88 001346 000000 WIDE: 0 ;NO. OF WIDE STATES
89 001350 000000 NARROW: 0 ;NO. OF NARROW STATES
90 001352 000000 FIRST: 0
91 001354 000000 SKIPST: 0 ;NO. OF SKIPPED STATES
92 001356 000000 TEMP: 0 ;WORK AREA
93 001360 000000 CH1: 0 ;FIRST CHANNEL
94 001362 000000 CH2: 0 ;SECOND CHANNEL
95 001364 000000 NBEXT: 0 ;NO. OF ADV11'S TO BE TESTED
96 001366 000000 NMBEXT: 0 ;NO. OF ADV11'S TO BE TESTED
97 001370 000000 DUMMY: 0 ;DUMMY CHANNEL
98 001372 000000 CHANL: 0 ;CHANNEL VALUE
99 001374 000000 TADDR: 0 ;TEST ADDRESS
100 001376 000000 RNA: 0 ;RANDOM
101 001400 000000 RNB: 0 ;NUMBER
102 001402 000000 RNC: 0 ;VALUES
103 001404 000000 RMS: 0 ;RMS NOISE VALUE
104 001406 000000 PEAK: 0 ;PEAK NOISE VALUE
105 001410 000000 FLAG: 0 ;VT55 FLAG
106 001412 000000 SPREAD: 0 ;DEVIATION FROM THE NOMINAL
107 001414 000000 DAC: 0 ;SAR VALUE
108 001416 000000 DELAY: 0 ;TIME DELAY COUNTER
109 001420 000000 EDGE: 0 ;EDGE VALUE
110 001422 000000 BITPNT: 0
111 001424 000000 MIN: 0 ;MIN VALUE
112 001426 000000 WFTST: 0 ;OPTION TEST AREA FLAG
113 001430 000000 MAX: 0 ;MAX VALUE
114 001432 000000 PERCNT: 0 ;PERCENT FOR SAR ROUTINE
115 001434 000000 OUT: 0
116 001436 000000 GUNITS: 0
117 001440 000001 TSTBIT: 1
118
119 001442 UNEXP:
(1) 001442 012737 001456 001162 MOV #1$, $ESCAPE ;:ESCAPE TO 1$ ON ERROR
120 001450 005237 001103 INC $ERFLG
121 001454 104003 ERROR 3
122 001456 005037 001162 1$: CLR $ESCAPE ;:RETURN ESCAPE TO NORMAL
123 001462 000002 RTI ;UNEXPECTED INTERRUPT
```

```

125          .SBTTL      CONTROL A AND C DECODERS
126 001464 010046      ISERV:  MOV      RO,-(SP)      ;SAVE RO
127 001466 017700 177454  MOV      @STKB,RO      ;GET CHARACTER
128 001472 042700 177600  BIC      #177600,RO
129 001476 120027 000003  CMPB    RO,#3          ;IS IT ^C?
130 001502 001010      BNE      1$
131 001504 104401 012056  TYPE    .CMMSG        ;ECHO CHARACTER
132 001510 012706 001100  MOV      #STACK,SP
133 001514 004737 011320  JSR      PC,RST       ;RESET & SET INTRPT. EN.
134 001520 000137 002262  JMP      BEG2
135 001524 120027 000001  1$:  CMPB    RO,#1          ;IS IT ^A?
136 001530 001010      BNE      2$
137 001532 104401 012051  TYPE    .AMSG        ;ECHO CHARACTER
138 001536 012706 001100  MOV      #STACK,SP
139 001542 004737 011320  JSR      PC,RST       ;RESET & SET INTRPT. EN.
140 001546 000177 177622  JMP      @TADDR       ;RETURN TO TEST
141 001552 120027 000007  2$:  CMPB    RO,#7          ;IS IT ^G?
142 001556 001027      BNE      NONE
143 001560 023727 001140 177570  CMP      SWR,#1775,u   ;HARDWARE SWREG?
144 001566 001423      BEQ      NONE
145 001570 104401 012063  TYPE    .GMSG        ;ECHO CHARACTER
146 001574 017746 177340  MOV      @SWR,-(SP)   ;;SAVE @SWR FOR TYPEOUT
(1)          ;;TYPE SWREG
(1) 001600 104403      TYPOS    ;;GO TYPE--OCTAL ASCII
(1) 001602      006      .BYTE    6           ;;TYPE 6 DIGITS
(1) 001603      001      .BYTE    1           ;;TYPE LEADING ZEROS
147 001604 104401 012243  TYPE    .SLASH
148 001610 104410      RDOCT
149 001612 012677 177322  MOV      (SP)+,@SWR   ;READ NEW VALUE
150 001616 012600      POPRO.  MOV      (SP)+,RO ;LOAD NEW SWREG VALUE
151 001620 022776 000001 000000  RETURN: CMP      #1,@0(SP) ;DOES IT RETURN TO A WAIT?
152 001626 001002      BNE      RET2        ;NO
153 001630 062716 000002  RET1:  ADD      #2,(SP) ;BUMP RETURN ADDRESS
154 001634 000002      RET2:  RTI
155 001636 104401 012047  NONE:  TYPE    .QUEST ;TYPE '?'
156 001642 000765      BR      POPRO
    
```

```

158 .SBTTL INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
159 001644 005037 001426 BEGIN: CLR WFTST
160 001650 000403 BR RBEG
161 001652 012737 000001 001426 BEGIN2: MOV #1,WFTST
162 001660 000005 RBEG: RESET
163 .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001662 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
(1) 001666 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 001670 022706 001140 CMP #SWR,R6 ;;DONE?
(1) 001674 001374 BNE -6 ;;LOOP BACK IF NO
(1) 001676 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 001702 012737 015224 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001710 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
(1) 001716 012737 015502 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001724 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
(1) 001732 012737 017072 000034 MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001740 012737 000340 000036 MOV #340,@TRAPVEC+2;LEVEL 7
(1) 001746 013737 011704 011676 MOV SENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
(1) 001754 005037 001160 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001760 005037 001162 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001764 112737 000001 001115 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
(1) 001772 012737 001772 001106 MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002000 012737 002000 001110 MOV #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002006 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 002012 012737 002046 000004 MOV #64,$@ERRVEC ;;SET UP ERROR VECTOR
(2) 002020 012737 177570 001140 MOV #DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 002026 012737 177570 001142 MOV #DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 002034 022777 177777 177076 CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
(2) 002042 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;;AND THE HARDWARE SWR IS NOT = -1
(2) 002044 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
(2) 002046 012716 002054 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
(2) 002052 000002 RTI
(2) 002054 012737 000176 001140 65$: MOV #SWREG,$SWR ;;POINT TO SOFTWARE SWR
(2) 002062 012737 000174 001142 MOV #DISPREG,$DISPLAY
(2) 002070 012637 000004 66$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002074 005037 001202 CLR $PASS ;;CLEAR PASS COUNT
(2) 002100 132737 000200 001215 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 002106 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
(2) 002110 012737 001216 001140 MOV #SWREG,$SWR ;;NO,USE APT SWITCH REGISTER
(2) 002116 67$:
    
```

```
165 002116 005037 001410 CLR FLAG ;CLEAR VT55 FLAG
166 002122 005737 000042 TST @#42 ;IS IT CHAINED?
167 002126 001033 BNE REST1
168 .SBTTL DETERMIN. IF VT55 TYPE TERMINAL IS PRESENT
169 002130 042777 000100 177006 BIC #100,@$TKS
170 002136 104401 013744 TYPE .CO ;TYPE ASCIZ STRING
171 002142 004737 002432 JSR PC,VTFLG ;GET A CHARACTER
172 002146 020027 000033 CMP RO,#33
173 002152 001017 BNE NOVT55 ;NO VT55 PRESENT
174 002154 004737 002432 JSR PC,VTFLG ;GET A CHARACTER
175 002160 020027 000057 CMP RO,#57
176 002164 001012 BNE NOVT55 ;NO VT55 PRESENT
177 002166 004737 002432 JSR PC,VTFLG ;GET A CHARACTER
178 002172 020027 000103 CMP RO,#103
179 002176 001403 BEQ VT55 ;VT55 IS PRESENT
180 002200 020027 000105 CMP RO,#105
181 002204 001002 BNE NOVT55
182 002206 005237 001410 VT55: INC FLAG
```

```

184 .SBTTL DIALOGUE TO DETERMINE WHICH TEST TO RUN
185 NOVT55: TYPE ,HEAD1
186 REST1: RESET
187 JSR PC, FIXONE ;INITIALIZE ADDRESSES
188 MOV KBVECT, RO
189 MOV #ISERV, (RO)+
190 MOV #340, (RO)
191 MOV #62341, RNA ;RANDOM NO, VARIABLES
192 MOV #142315, RNB
193 MOV #127623, RNC
194 BEG2: MOV #STACK, SP ;RESET STACK POINTER INCASE RESTARTED
195 RESET ;RESTART ADDRESS
196 TST @#42 ;IS IT CHAINED?
197 BEQ 1$
198 2$: JMP BEGL ;GO TO LOGIC TESTS
199 TST WFTST ;TEST FOR OPTION TEST
200 BNE 2$ ;;
201 1$: TYPE ,MSG71
202 TRYAG: RDLIN
203 BIS #100, @STKS
204 CLR -(SP) ;CLEAR PSW
205 MOV #1$, -(SP)
206 RTI
207 1$: MOV (SP)+, RO ;READ ANSWER
208 BICB #40, (RO)
209 CMPB (RO), #'A ;IS IT A?
210 BNE 2$ ;;NO, TRY C
211 JMP BEGINA ;GO TO AUTO TEST
212 2$: CMPB (RO), #'C ;IS IT C?
213 BNE 3$ ;;NO, TRY P
214 JMP BEGINC ;GO TO CALIBRATION TEST
215 3$: CMPB (RO), #'P ;IS IT P?
216 BNE 4$ ;;NO, TRY L
217 JMP BEGINP ;GO TO DISPLAY CONVERSIONS TEST
218 4$: CMPB (RO), #'L ;IS IT L?
219 BNE 5$ ;;NO, TRY W
220 JMP BEGL ;GO TO LOGIC TESTS
221 5$: CMPB (RO), #'W ;IS IT W?
222 BNE 6$ ;;NO, TRY AGAIN
223 JMP BEG:NW ;GO TO WRAPAROUND TEST
224 6$: TYPE ,QUEST
225 BR TRYAG ;WAIT FOR CHARACTER

```



```

227 002432 005000          VTFLG: CLR      RO          ;TEST FOR PRESENCE
228 002434 105777 176504 1$:  TSTB   @STKS      ;OF VT55
229 002440 100404          BMI     2$          ;:VT55 RESPONDS WITH <33><57>[<103> OR <105>]
230 002442 005300          DEC     RO          ;
231 002444 001373          BNE    1$          ;
232 002446 005726          TST   (SP)+        ;POP A WORD OFF STACK
233 002450 000660          BR     NOVTS5      ;:NO VT55 PRESENT
234 002452 017700 176470 2$:  MOV    @STKB,RO
235 002456 042700 177600  BIC   #177600,RO  ;TEST VT55 CODE
236 002462 000207          RTS    PC
237
238 002464 005037 001202  TESTAD: CLR    $PASS      ;CLEAR PASS COUNT
239 002470 005037 001436  CLR    GUNITS      ;CLEAR UNIT ERROR BITS
240 002474 012737 000001 001440  MOV    #1,TSTBIT   ;INITIALIZE MODULE ERROR TEST BIT
241 002502 012737 000001 001356  MOV    #1,TEMP     ;SET UP FOR ONLY ONE A/D
242 002510 105737 001215  TSTB  $ENVM        ;TESTING ONLY ONE A/D?
243 002514 100411          BMI    3$          ;:YES
244 002516 012737 000004 001356  MOV    #4,TEMP     ;SET UP MAX NO OF A/D'S
245 002524 005737 001426  TST   WFTST       ;IS IT IN OPTION TEST
246 002530 001403          BEQ   3$          ;:NOT IN OPTION TEST
247 002532 012737 000020 001356  MOV    #16,TEMP    ;SET UP OPTION MAX NO OF A/D'S
248 002540 013737 001250 001126 3$:  MOV    $BASE,$BDDAT ;SETUP TO TEST FOR ADV11'S
249 002546 013746 000004  MOV    @ERRVEC,-(SP) ;SAVE ERRVEC
250 002552 012737 002624 000004  MOV    #2$,ERRVEC  ;SET UP FOR TIME OUT ERROR
251 002560 005037 001364          CLR    NBEXT      ;CLEAR ADV11 COUNTER
252 002564 005777 176336          TST   @SBDDAT     ;ADDRESS ADV11
253 002570 005237 001364          INC   NBEXT      ;INCREMENT ADV11 COUNTER
254 002574 053737 001440 001436  BIS   TSTBIT,GUNITS ;SET A/D BIT UNDER TEST
255 002602 006337 001440          ASL   TSTBIT     ;SET TEST BIT FOR NEXT UNIT
256 002606 005337 001356          DEC   TEMP       ;REACHED MAX?
257 002612 001405          BEQ   4$          ;:REACHED MAX NO OF A/D'S
258 002614 063737 001336 001126  ADD   VADR,$BDDAT ;GET NEXT ADV11
259 002622 000760          BR    1$          ;:TRY NEXT ADV11
260 002624 022626          2$:  CMP    (SP)+,(SP)+ ;POP 2 WORDS OFF STACK
261 002626          4$:
(1) 002626 013746 001364  MOV    NBEXT,-(SP) ;:SAVE NBEXT FOR TYPEOUT
(1)
(1) 002632 104403          TYPOS
(1) 002634 002          .BYTE 2          ;:TYPE NUMBER OF ADV11'S
(1) 002635 000          .BYTE 0          ;:GO TYPE--OCTAL ASCII
(1) 002635 000          .BYTE 0          ;:TYPE 2 DIGIT(S)
262 002636 104401 013121          TYPE ,MSG50      ;:SUPPRESS LEADING ZEROS
263 002642 005337 001364          DEC   NBEXT      ;ADJUST ADV11 COUNT
264 002646 013737 001364 001366  MOV    NBEXT,NMBEXT ;KEEP COUNT OF NUMBER
265 002654 012637 000004  MOV    (SP)+,ERRVEC ;RESTORE ERRVEC
266 002660 012737 000001 001440  MOV    #1,TSTBIT   ;INITIALIZE MODULE ERROR TEST BIT
267 002666 000207          RTS    PC
  
```

```
269 002670 BEGINL:
270 *****
(3) *TEST 1 FLOAT A ONE THRU MULTIPLEXER BITS
(3) *****
(2) 002670 012737 002670 001106 TST1: MOV #TST1,$LPADR
271 002676 012737 002670 001110 MOV #TST1,$LPERR
272 002704 012737 000400 001124 MOV #BIT8,$GDDAT ;LOAD FIRST BIT
273 002712 104412 2$: CHKIT
274 002714 104001 ERROR 1 ;FAILED TO LOAD + READ BIT
275 002716 006337 001124 1$: ASL $GDDAT ;GET NEXT BIT
276 002722 023727 001124 010000 CMP $GDDAT,#BIT12 ;FINISHED?
277 002730 001370 BNE 2$ ;:NO,GO TO NEXT TEST
278
279 *****
(3) *TEST 2 LOAD AND READ BACK ERROR I.E. BIT14
(3) *****
(2) 002732 000004 TST2: SCOPE
280 002734 012737 040000 001124 MOV #BIT14,$GDDAT
281 002742 104412 CHKIT
282 002744 104001 ERROR 1 ;FAILED TO LOAD + READ ERROR I.E.
283
(3) *TEST 3 LOAD AND READ BACK INTERRUPT ENABLE BIT6
(3) *****
(2) 002746 000004 TST3: SCOPE
284 002750 012737 001442 176346 MOV #UNEXP,@VECTOR ;SETUP FOR UNEXPECTED INTERUPT
285 002756 012737 000100 001124 MOV #BIT6,$GDDAT ;LOAD EXPECTED DATA
286 002764 104412 CHKIT
287 002766 104001 ERROR 1 ;FAILED TO LOAD + READ INTERRUPT ENABLE
288
(3) *TEST 4 LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BITS
(3) *****
(2) 002770 000004 TST4: SCOPE
290 002772 012737 000040 001124 MOV #BIT5,$GDDAT ;LOAD EXPECTED DATA
291 003000 104412 CHKIT
292 003002 104001 ERROR 1 ;FAILED TO LOAD + READ CLOCK OVERFLOW START ENAB
293
(3) *TEST 5 LOAD AND READ BACK EXTERNAL START ENABLE BIT4
(3) *****
(2) 003004 000004 TST5: SCOPE
295 003006 012737 000020 001124 MOV #BIT4,$GDDAT ;LOAD EXPECTED DATA
296 003014 104412 CHKIT
297 003016 104001 ERROR 1 ;FAILED TO LOAD + READ EXT. START ENABLE
298
(3) *TEST 6 LOAD AND READ BACK MAINT. TST BIT2
(3) *****
(2) 003020 000004 TST6: SCOPE
299 003022 012737 000004 001124 MOV #BIT2,$GDDAT
300 003030 104412 CHKIT
301 003032 104001 ERROR 1 ;FAILED TO LOAD + READ BACK MAINT. TST
```

```

303      ::*****
(3)      ::*TEST 7      LOAD AND READ BACK ENABLE I.D. BIT3
(3)      ::*****
(2) 003034 000004 000010 001124 TST7: SCOPE
304 003036 012737          MOV      #BIT3,$GDDAT
305 003044 104412          CHKIT
306 003046 104001          ERROR 1          ;FAILED TO LOAD + READ ENABLE I.D. BIT
307      ::*****
(3)      ::*TEST 10     TEST I.D. BIT (BIT 12) CLEARED
(3)      ::*****
(2) 003050 000004 000001 176236 TST10: SCOPE
308 003052 012777          MOV      #1,@STREG          ;CLEAR I.D. ENABLE
309 003060 105777          TSTB     @STREG          ;WAIT FOR CONVERSION
310 003064 100375          BPL      1$              ;;CONVERSION IS NOT DONE YET
311 003066 032777          BIT      #BIT12,@ADBUFF ;IS I.D. BIT CLEARED?
312 003074 001401          BEQ      TST11          ;;YES - GOTO NEXT TEST
313 003076 104001          ERROR 1
314
315      ::*****
(3)      ::*TEST 11     TEST I.D. BIT (BIT 12) SET
(3)      ::*****
(2) 003100 000004 000011 176206 TST11: SCOPE
316 003102 012777          MOV      #BIT3!BIT0,@STREG ;SET I.D. ENABLE BIT
317 003110 105777          TSTB     @STREG          ;WAIT FOR CONVERSION
318 003114 100375          BPL      1$              ;;CONVERSION IS NOT DONE YET
319 003116 032777          BIT      #BIT12,@ADBUFF ;IS I.D. BIT SET?
320 003124 001001          BNE     TST12          ;;YES - GOTO NEXT TEST
321 003126 104001          ERROR 1
322
323      ::*****
(3)      ::*TEST 12     LOAD AND READ BACK ERROR FLAG BIT15
(3)      ::*****
(2) 003130 000004 100600 001124 TST12: SCOPE
324 003132 012737          MOV      #BIT15,$GDDAT ;LOAD EXPECTED DATA
325 003140 104412          CHKIT
326 003142 104001          ERROR 1          ;FAILED TO LOAD + READ ERROR FLAG
327      ::*****
(3)      ::*TEST 13     TEST INIT CLEARS BITS 2-6,8-11,14
(3)      ::*****
(2) 003144 000004 000300 001160 TST13: SCOPE
(1) 003146 012737          MOV      #300,$TIMES      ;;DO 300 ITERATIONS
328 003154 005037          CLR      $GDDAT          ;LOAD EXPECTED DATA
329 003160 012777          MOV      #47574,@STREG   ;SET STATUS REGISTER
330 003166 000005          RESET                    ;INITIALIZE
331 003170 052777          BIS      #100,@STKS      ;SET INTRPT. ENABLE
332 003176 017737          MOV      @STREG,$BDDAT   ;READ STATUS REGISTER
333 003204 001401          BEQ      TST14          ;;NEXT TEST
334 003206 104001          ERROR 1          ;RESET FAILED TO CLEAR AD ST. REG. BITS
335
  
```

```

337      :*****
(3)      :*TEST 14      TEST INIT CLEARS ERROR FLAG
(3)      :*****
(2) 003210 000004
338 003212 012737 000300 001160 TST14: SCOPE
339 003220 012777 100000 176070      MOV      #300,$TIMES      ;DO 300 ITERATIONS
340 003226 000005      MOV      #BIT15,@STREG      ;SET BIT 15
341 003230 052777 000100 175706      RESET      ;ISSUE INIT
342 003236 104411      BIS      #100,@$TKS      ;SET INTRPT. EN. FOR KEYBOARD
343 003240 104001      CHECK
344      ERROR      1
(3)      :*****
(3)      :*TEST 15      TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
(3)      :*****
(2) 003242 000004
345 003244 012700 001000 TST15: SCOPE
346 003250 005277 176042      MOV      #BIT9,R0      ;STALL TIME COUNTER
347 003254 012737 000200 001124      INC      @STREG      ;START CONVERSION
348 003262 005300      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
349 003264 001376      1$: DEC      R0      ;STALL
350 003266 042777 100000 176022      BNE      1$      ;TIME
351 003274 104411      BIC      #BIT15,@STREG      ;MASK OUT ERROR BIT
352 003276 104001      CHECK
353 003300 017700 176016      ERROR      1      ;A/D DONE FLAG FAILED TO SET;BIT0 FAILED TO CLEAR
      MOV      @ADBUFF,R0      ;CLEAR DONE FLAG FOR ITERATIONS

```

```

355
(3)
(3)
(2) 003304 000004
(1) 003306 012737 000300 001160
356 003314 005037 001124
357 003320 005277 175772
358 003324 105777 175766
359 003330 100375
360 003332 000005
361 003334 104411
362 003336 104001
363 003340 052777 000100 175576
364
365
(3)
(3)
(2) 003346 000004
366 003350 005277 175742
367 003354 105777 175736
368 003360 100375
369 003362 017700 175734
370 003366 104411
371 003370 104001
372
(3)
(3)
(2) 003372 000004
373 003374 005037 001124
374 003400 005037 001372
375 003404 005037 001412
376 003410 012777 000005 175700
377 003416 105777 175674
378 003422 100375
379 003424 017737 175672 001126
380 003432 001401
381 003434 104004
382
383
(3)
(3)
(2) 003436 000004
384 003440 012737 007777 001124
385 003446 012737 000001 001372
386 003454 005037 001412
387 003460 012777 000405 175630
388 003466 105777 175624
389 003472 100375
390 003474 017737 175622 001126
391 003502 023737 001124 001126
392 003510 001401
393 003512 104004

*****
*TEST 16 TEST INIT CLEARS DONE FLAG
*****
TST16: SCOPE
MOV #300,$TIMES ;DO 300 ITERATIONS
CLR $GDDAT ;CLEAR EXPECTED
INC @STREG ;START CONVERSION
2$: TSTB @STREG
BPL 2$
RESET
CHECK
ERROR 1 ;DONE FLAG FAILED TO CLEAR
BIS #100,@STKS ;SET INTRPT. EN. BIT

*****
*TEST 17 TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
*****
TST17: SCOPE
INC @STREG ;SET A/D START CONVERSION BIT
1$: TSTB @STREG ;WAIT FOR FLAG
BPL 1$
MOV @ADBUFF,R0 ;READ CONVERTED VALUE
CHECK
ERROR 1 ;DONE FLAG FAILED TO CLEAR

*****
*TEST 20 TEST ALL '0'S RESULTS USING MAINT. ADTST. BIT
*****
TST20: SCOPE
CLR $GDDAT ;CLEAR EXPECTED VALUE
CLR CHANL ;SET CHANL = 0
CLR SPREAD ;SET SPREAD = 0
MOV #5,@STREG ;CONVERT EVEN CHANNEL WITH MAINT. BIT SET
1$: TSTB @STREG ;WAIT FOR DONE
BPL 1$
MOV @ADBUFF,$BDDAT ;RESULTS TO BDDAT FOR CHECKING
BEQ TST21 ;GOTO NEXT TEST
ERROR 4 ;DID NOT GET ALL '0'S RESULT WITH MAINT. ADTST

*****
*TEST 21 TEST ALL '1'S RESULT USING MAINT. ADTST. BIT
*****
TST21: SCOPE
MOV #777,$GDDAT ;EXPECT ALL '1'S RESULT
MOV #1,CHANL ;SET CHANL = 1
CLR SPREAD ;SET SPREAD = 0
MOV #405,@STREG ;CONVERT ODD CHANNEL WITH MAINT. BIT SET
1$: TSTB @STREG ;WAIT FOR DONE
BPL 1$
MOV @ADBUFF,$BDDAT ;RESULTS TO BDDAT FOR CHECKING
CMP $GDDAT,$BDDAT ;EQUAL?
BEQ TST22 ;GOTO NEXT TEST
ERROR 4 ;DID NOT GET ALL '1'S RESULT WITH MAINT. ADTST
  
```

```
395 ::*****  
(3) :*TEST 22 GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION  
(3) :*****  
(2) 003514 000004 TST22: SCOPE  
396 :* 'ENTERING TEST 22' 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 UNIBUS COULD GET 'HUNG' WHILE  
(1) :*EXECUTING TEST '22'.  
(1) 003516 012700 000022 MOV #22,R0 ;GET TEST NO.  
(1) 003522 004737 011216 JSR PC,DUMW ;PRINT MESSAGE  
397 003526 005046 CLR -(SP) ;RESET PRIORITY  
398 003530 012746 003536 MOV #3$,-(SP)  
399 003534 000002 RTI  
400 003536 012777 003612 175560 3$: MOV #1$,@VECTOR ;INTERRUPT VECTOR ADDRESS  
401 003544 012777 000200 175556 MOV #200,@VECTR1 ;SET UP NEW PSW  
402 003552 012777 000101 175536 MOV #BIT6!BIT0,@STREG ;SET INTERRUPT ENABLE BIT + START CONVERSION  
403 003560 105777 175532 2$: TSTB @STREG ;WAIT FOR DONE  
404 003564 100375 BPL 2$ ;FLAG TO SET  
405 003566 017737 175524 001126 MOV @STREG,$BDDAT ;READ STATUS REGISTER  
406 003574 012737 000300 001124 MOV #BIT7!BIT6,$GDDAT ;GOOD DATA  
407 003602 104002 ERROR 2 ;FAILED TO INTERRUPT ON DONE  
408 003604 004737 011270 JSR PC,DUMC ;TYPE COMPLETED  
409 003610 000414 BR TST23 ;BRANCH TO NEXT TEST  
410 003612 022626 1$: CMP (SP)+,(SP)+ ;RESET STACK POINTER  
411 003614 012777 001442 175502 MOV #UNEXP,@VECTOR ;SET UP FOR UNEXPECTED INTERRUPT  
412 003622 005046 CLR -(SP) ;CLEAR PSW  
413 003624 012746 003632 MOV #4$,-(SP)  
414 003630 000002 RTI  
415 003632 004737 011270 4$: JSR PC,DUMC ;TYPE COMPLETED  
416 003636 005777 175460 TST @ADBUFF ;CLEAR DONE BIT  
417 ::*****  
(3) :*TEST 23 TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET  
(3) :*****  
(2) 003642 000004 TST23: SCOPE  
418 :* 'ENTERING TEST 23' TYPED OUT TO TELL YOU THE NEXT  
(1) :*TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.  
(1) :*THERE IS DANGER THAT THE UNIBUS COULD GET 'HUNG' WHILE  
(1) :*EXECUTING TEST '23'.  
(1) 003644 012700 000023 MOV #23,R0 ;GET TEST NO.  
(1) 003650 004737 011216 JSR PC,DUMW ;PRINT MESSAGE  
419 003654 012777 003714 175450 MOV #1$,@VECTR2 ;SETUP VECTOR ADDRESS  
420 003662 012777 140000 175426 MOV #BIT15!BIT14,@STREG ;CAUSE AN INTERRUPT  
421 003670 017737 175422 001126 MOV @STREG,$BDDAT ;BAD DATA  
422 003676 012737 140000 001124 MOV #BIT15!BIT14,$GDDAT ;GOOD DATA  
423 003704 104002 ERROR 2  
424 003706 004737 011270 JSR PC,DUMC ;TYPE COMPLETED  
425 003712 000627 BR TST20  
426 003714 022626 1$: CMP (SP)+,(SP)+ ;POP STACK  
427 003716 004737 011270 JSR PC,DUMC  
428 003722 005077 175370 CLR @STREG
```

```

430      ::*****
(3)      ::*TEST 24      TEST ERROR FLAG SETS IF 2ND CONVERSION ENDS BEFORE READING BUFFER
(3)      ::*****
(2) 003726 000004 TST24: SCOPE
431 003730 012777 000001 175360 MOV #BIT0,@STREG ;START CONVERSION
432 003736 105777 175354 1$: TSTB @STREG ;WAIT FOR
433 003742 100375 BPL 1$
434 003744 012737 100200 001124 2$: MOV #BIT15!BIT7,$GDDAT ;LOAD EXPECTED VALUE
435 003752 012777 000001 175336 MOV #BIT0,@STREG ;START 2ND CONVERSION
436 003760 012700 001000 MOV #BIT9,R0 ;WAIT FOR 2ND
437 003764 005300 3$: DEC R0 ;CONVERSION TO END
438 003766 001376 BNE 3$
439 003770 104411 4$: CHECK
440 003772 104001 ERROR 1 ;ERROR FLAG NOT SET WHEN 2ND
441 ; CONVERT ENDS BEFORE READ BUFFER FROM FIRST
442 003774 017700 175322 MOV @ADBUFF,R0 ;CLEAR DONE FLAG
443      ::*****
(3)      ::*TEST 25      TEST ERROR FLAG SETS IF START 2ND CONV. BEFORE DONE FLAG SETS
(3)      ::*****
(2) 004000 000004 TST25: SCOPE
444 004002 012737 100000 001124 MOV #BIT15,$GDDAT ;LOAD EXPECTED DATA
445 004010 012777 000001 175300 MOV #BIT0,@STREG ;START CONVERSION
446 004016 112777 000001 175272 MOVB #BIT0,@STREG ;START NEXT CONVERSION
447 004024 112777 000001 175264 MOVB #BIT0,@STREG ;ONCE AGAIN IN CASE REFRESH INTERVENED
448 004032 017737 175260 001126 MOV @STREG,$BDDAT ;READ STATUS REGISTER
449 004040 042737 077777 001126 BIC #77777,$BDDAT ;MASK OUT BIT 15
450 004046 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULTS
451 004054 001401 BEQ 1$ ;BRANCH OVER ERROR
452 004056 104001 ERROR 1 ;ERROR FLAG NOT SET WHEN 2ND
453 ; CONVERT BEGINS BEFORE FIRST DONE
454 004060 017700 175236 1$: MOV @ADBUFF,R0 ;READ CONVERTED VALUE
455 004064 005077 175226 CLR @STREG ;CLEAR STATUS REGISTER
456 004070 000004 SCOPE
457 004072 000207 RTS PC ;RETURN TO TEST SECTION
458
459
460      ;;SUBROUTINE FOR LOGIC TESTS;;
461 004074 013777 001124 175214 TESTIT: MOV $GDDAT,@STREG ;LOAD EXPECTED VALUE
462 004102 017737 175210 001126 TEST: MOV @STREG,$BDDAT ;READ ST. REG.
463 004110 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULTS
464 004116 001002 BNE RETERR ;;ERROR RETURN
465 004120 062716 000002 ADD #2,(SP) ;BUMP RETURN ADDRESS TO GET AROUND ERROR
466 004124 000002 RETERR: RTI
  
```

468
469 004126
470
(3)
(3)
(2) 004126 012737 000026 001102
(1) 004134 012737 000010 001160
471 004142 012737 004126 001110
472 004150 012737 004126 001106
473 004156 004537 011036
474 004162 000000
475 004164 004537 011150
476 004170 004000
477 004172 011616
478 004174 104004
479
(3)
(3)
(2) 004176 000004
(1) 004200 012737 000010 001160
480 004206 004537 011036
481 004212 000001
482 004214 004537 011150
483 004220 007344
484 004222 011622
485 004224 104004
486
487
(3)
(3)
(2) 004226 000004
(1) 004230 012737 000010 001160
488 004236 004537 011036
489 004242 000002
490 004244 004537 011150
491 004250 000434
492 004252 011622
493 004254 104004
494
(3)
(3)
(2) 004256 000004
(1) 004260 012737 000010 001160
495 004266 012737 000004 004300
496 004274 004537 011036
497 004300 000004
498 004302 004537 011150
499 004306 004000
500 004310 011616
501 004312 104004
502 004314 005237 004300
503 004320 022737 000017 004300
504 004326 001362

```
.SBTTL      WRAPAROUND TEST SECTION
WRAP
*****
*TEST 26    TEST CHO GROUND
*****
TST26:  MOV    #STN,$STNM
        MOV    #10,$TIMES      ;;DO 10 ITERATIONS
        MOV    #TST26,$LPERR
        MOV    #TST26,$LPADR
        JSR    R5,CONVRT      ;CONVERT 8 TIMES
        0
        JSR    R5,COMPAR      ;COMPARE RESULTS
        4000                  ;NOMINAL
        V12                    ;TOLERANCE
        ERROR 4                ;ERROR ON A/D CHANNEL
*****
*TEST 27    TEST CH1 +4.5 VOLT
*****
TST27:  SCOPE
        MOV    #10,$TIMES      ;;DO 10 ITERATIONS
        JSR    R5,CONVRT      ;CONVERT 8 TIMES
        1                    ;CHANNEL 1
        JSR    R5,COMPAR      ;COMPARE RESULTS
        7344                  ;NOMINAL
        V326                  ;TOLERANCE
        ERROR 4                ;ERROR ON A/D CHANNEL
*****
*TEST 30    TEST CH2 -4.5 VOLT
*****
TST30:  SCOPE
        MOV    #10,$TIMES      ;;DO 10 ITERATIONS
        JSR    R5,CONVRT      ;CONVERT 8 TIMES
        2                    ;CHANNEL 2
        JSR    R5,COMPAR      ;COMPARE RESULTS
        434                   ;NOMINAL
        V326                  ;TOLERANCE
        ERROR 4                ;ERROR ON A/D CHANNEL
*****
*TEST 31    TEST GROUND ON CHANNELS 4 - 17
*****
TST31:  SCOPE
        MOV    #10,$TIMES      ;;DO 10 ITERATIONS
        MOV    #4,2$          ;SET UP FIRST CHANNEL
        JSR    R5,CONVRT      ;CONVERT CHANNEL
        4
        JSR    R5,COMPAR      ;TEST RESULTS
        4000
        V12
        ERROR 4
        INC    2$            ;GET NEXT CHANNEL
        CMP    #17,2$        ;DONE?
        BNE   1$            ;;NO
```



```

506      ::*****
(3)      ::*TEST 32      TEST VERNIER OFFSET DAC ON CHO
(3)      ::*****
(2) 004330 000004      TST32: SCOPE
(1) 004332 012737 000001 001160      MOV #1,STIMES      ;;DO 1 ITERATION
507 004340 005077 174756      CLR @ADBUFF      ;SET VERNIER DAC = 0
508 004344 004537 011036      JSR R5,CONVRT    ;CONV. CHO, DIRECT VERNIER DAC
509 004350 000000      0
510 004352 013704 001356      MOV TEMP,R4      ;SAVE VALUE IN R4
511 004356 012777 000377 174736 1$: MOV #377,@ADBUFF  ;SET VERNIER DAC = 377
512 004364 004537 011036      JSR R5,CONVRT    ;CONVERT IT
513 004370 000000      0
514 004372 160437 001356      SUB R4,TEMP      ;TEMP=DIFF. BETWEEN VALUE & PREVIOUS
515 004376 004537 011150      JSR R5,COMPAR    ;COMPARE RESULTS
516 004402 000005      5
517 004404 011612      V2
518 004406 104004      ERROR 4
519      ::*****
(3)      ::*TEST 33      OFFSET ON CHO
(3)      ::*****
(2) 004410 000004      TST33: SCOPE
(1) 004412 012737 000001 001160      MOV #1,STIMES      ;;DO 1 ITERATION
520 004420 013737 001342 001372      MOV BASECH,CHANL ;LOAD CHANNEL
521 004426 013737 001342 001370      MOV BASECH,DUMMY ;LOAD DUMMY
522 004434 004737 005160      JSR PC,OFFSET    ;FIND OFFSET
523 004440 104401 013756      TYPE ,MOFFSET    ;TYPE 'OFFSET='
524 004444 004737 005234      JSR PC,TOFF      ;TYPE OFFSET
525 004450 004537 011150      JSR R5,COMPAR    ;IS RESULT WITHIN LIMITS?
526 004454 000000      0
527 004456 011620      VSOD
528 004460 000401      BR OFFERR        ;NO-ERROR
529 004462 000403      BR OFFOK         ;YES-OK
530 004464 104401 012567      OFFERR: TYPE ,ERMSG
531 004470 000402      BR TST34         ;;GO TO NEXT TEST
532 004472 104401 012245      OFFOK: TYPE ,OKMSG
  
```

```

534      ::*****
(3)      :*TEST 34      TEST RAMP RANGE, CH3
(3)      :*****
(2) 004476 000004      TST34: SCOPE
535 004500 012737 000001 001160      MOV #1,$TIMES      ;DO THIS ONCE
536 004506 012703 007777      MOV #7777,R3      ;INIT R3 VALUE
537 004512 005004      CLR R4      ;AND R4
538 004514 012777 001400 174574      MOV #1400,@STREG  ;SETUP FOR CH3
539 004522 012702 047040      MOV #20000.,R2    ;SETUP FOR 20,000 CONVERSIONS
540 004526 105277 174564      1$: INCB @STREG
541 004532 105777 174560      2$: TSTB @STREG
542 004536 100375      BPL 2$
543 004540 027704 174556      CMP @ADBUFF,R4
544 004544 003402      BLE 3$
545 004546 017704 174550      MOV @ADBUFF,R4    ;HIT A NEW HIGH
546 004552 027703 174544      3$: CMP @ADBUFF,R3
547 004556 002002      BGE 4$
548 004560 017703 174536      MOV @ADBUFF,R3    ;HIT A NEW LOW
549 004564 005302      4$: DEC R2
550 004566 001357      BNE 1$
551 004570 010337 001356      MOV R3,TEMP
552 004574 004537 011150      JSR R5,COMPAR
553 004600 000000      O
554 004602 011610      VO
555 004604 104004      ERROR 4 ;RAMP DIDN'T REACH LOW END OF RANGE
556 004606 010437 001356      MOV R4,TEMP
557 004612 004537 011150      JSR R5,COMPAR
558 004616 007777      7777
559 004620 011610      VO
560 004622 104004      ERROR 4 ;RAMP DIDN'T REACH HIGH END OF RANGE
  
```

```
*****
:TEST 35      NOISE TEST, 1 EDGE
*****
TST35: SCOPE
(2) 004624 000004
(1) 004626 012737 000001 001160      MOV #1,$TIMES      ;;DO 1 ITERATION
563 004634 104401 012004              TYPE ,NOIMSG
564 004640 005037 001372              CLR CHANL          ;LOAD CHANNEL 0
565 004644 013737 001372 001370 1$:  MOV CHANL,DUMMY   ;LOAD DUMMY CHANNEL
566 004652 004737 006730              JSR PC,GETEDG     ;GET EDGE VALUE
567 004656 005037 001404              CLR RMS          ;CLEAR RMS VLAUE
568 004662 005037 001406              CLR PEAK         ;CLEAR PEAK VALUE
569 004666 004537 007110              JSR R5,SARSUB    ;DO SAR ROUTINE AT 16%
570 004672 000020
571 004674 063737 001414 001404      ADD DAC,RMS      ;ADD RESULT TO RMS
572 004702 004537 007110              JSR R5,SARSUB    ;DO SAR ROUTINE AT 84%
573 004706 000124
574 004710 163737 001414 001404      SUB DAC,RMS      ;SUBTRACT RESULT FROM RMS
575 004716 004537 007110              JSR R5,SARSUB    ;DO SAR ROUTINE AT 1%
576 004722 000001
577 004724 063737 001414 001406      ADD DAC,PEAK     ;ADD RESULT TO PEAK
578 004732 004537 007110              JSR R5,SARSUB    ;DO SAR ROUTINE AT 99%
579 004736 000143
580 004740 163737 001414 001406      SUB DAC,PEAK     ;SUBTRACT RESULT FROM PEAK
581 004746 012737 000001 007106      MOV #1,EDGFLG
582 004754 004737 010706              JSR PC,TYPRP
583 004760 005237 001372              INC CHANL        ;GET NEXT CHANNEL
584 004764 022737 000003 001372      CMP #3,CHANL     ;CHANNEL 3?
585 004772 001002                      BNE 2$          ;:NO
586 004774 005237 001372              INC CHANL        ;:CHANNEL 3 IS SKIPED
587 005000 022737 000017 001372 2$:  CMP #17,CHANL   ;:DONE?
588 005006 001316                      BNE 1$          ;:NO
```

```
590 .....  
(3) *TEST 36 INTERCHANNEL SETTLING TEST, 1 EDGE  
(3) .....  
(2) 005010 000004 TST36: SCOPE  
(1) 005012 012737 000001 001160 MOV #1,STIMES ;;DO 1 ITERATION  
591 005020 104401 012023 TYPE ,SETMSG ;TYPE 'SETTLING TEST'  
592 005024 012737 000001 001360 MOV #1,CH1 ;DO TEST BETWEEN CHANNEL 1 AND 2  
593 005032 012737 000002 001362 MOV #2,CH2  
594 005040 013737 001362 001372 1$: MOV CH2,CHANL  
595 005046 004737 006730 JSR PC,GETEDG ;GET EDGE VALUES  
596 005052 005002 CLR R2  
597 005054 004737 006666 JSR PC,SET1A ;SCALING = .02 LSB  
598 005060 004737 006666 JSR PC,SET1A ;MAKE IT .01 LSB  
599 005064 100001 BPL 2$  
600 005066 005402 NEG R2 ;MAKE IT POSITIVE  
601 005070 010204 2$: MOV R2,R4  
602 005072 012737 000001 007106 MOV #1,EDGFLG  
603 005100 004737 006536 JSR PC,TYPSET ;TYPE SETTLING INFORMATION  
604 005104 022737 000002 001360 CMP #2,CH1 ;DONE?  
605 005112 001410 BEQ TST37 ;:YES  
606 005114 013702 001360 MOV CH1,R2 ;SETTLE THE OTHER WAY  
607 005120 013737 001362 001360 MOV CH2,CH1  
608 005126 010237 001362 MOV R2,CH2  
609 005132 000742 BR 1$ ;:  
610 005134 3$:  
611 .....  
(3) *TEST 37 DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST  
(3) .....  
(2) 005134 000004 TST37: SCOPE  
(1) 005136 012737 000001 001160 MOV #1,STIMES ;;DO 1 ITERATION  
612 005144 005737 001202 TST SPASS ;FIRST TIME-SKIP DIFLIN  
613 005150 001402 BEQ LEND  
614 005152 004737 007310 JSR PC,DIFLIN  
615 005156 000207 LEND: RTS ;RETURN TO TEST SECTION  
616  
617 005160 012737 004001 001420 OFFSET: MOV #4001,EDGE ;4000,4001 EDGE  
618 005166 004537 007110 JSR R5,SARSUB  
619 005172 000062 SO.  
620 005174 013737 001414 001356 MOV DAC,TEMP  
621 005202 012737 004000 001420 MOV #4000,EDGE ;3777,4000 EDGE  
622 005210 004537 007110 JSR R5,SARSUB  
623 005214 000062 SO.  
624 005216 063737 001414 001356 ADD DAC,TEMP  
625 005224 162737 000400 001356 SUB #400,TEMP  
626 005232 000207 RTS PC
```


L 3

MAINDEC-11-DVADA-B MACY11 30A(1052) 25-JUL-78 15:54 PAGE 21
 CVADAB.P11 17-JUL-78 00:00 T37 DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST SEQ 0037

670	005436	104401	012417		AJOFF:	TYPE	,IGND		:GROUND CHANNEL
671	005442	104407				RDLIN			:WAIT FOR CR
672	005444	005726				TST	(SP)+		:POP 1 WORD OFF STACK
673	005446	104401	012357		1\$:	TYPE	,XADJ		:ADJUST MESSAGE
674	005452	104401	012456			TYPE	,CRWR		:TYPE 'TYPE CR WHEN READY'
675	005456	012703	000005			MOV	#5,R3		:SET UP COUNT
676	005462	004737	005160		2\$:	JSR	PC,OFFSET		:TEST AND TYPE OFFSET ERROR
677	005466	004737	005234			JSR	PC,TOFF		:TYPE OFFSET
678	005472	004737	005256			JSR	PC,TCHK		:CHECK FOR A CHARACTER AND DELAY
679	005476	000771				BR	2\$::
680	005500	000762				BR	1\$::NOT WITHIN TOLLERANCE, TRY AGAIN
681	005502	000005				RESET			
682	005504	000137	002262			JMP	BEG2		
683	005510	104401	012505		AJGAIN:	TYPE	,IVOLT		:INPUT +5.115 VOLTS ON CHANNEL
684	005514	104401	012456			TYPE	,CRWR		
685	005520	104407				RDLIN			:WAIT FOR CR
686	005522	005726				TST	(SP)+		:POP 1 WORD OFF STACK
687	005524	104401	012551		1\$:	TYPE	,YADJ		:ADJUST MESSAGE
688	005530	104401	012373			TYPE	,MOLSB		:TYPE '' FOR 0.00 LSB ERROR''
689	005534	104401	012456			TYPE	,CRWR		
690	005540	012703	000005			MOV	#5,R3		:SET UP COUNT
691	005544	012737	007777	001420	2\$:	MOV	#7777,EDGE		:LOOK FOR 7776,7777 EDGE
692	005552	004537	007110			JSR	R5,SARSUB		
693	005556	000062				SO.			
694	005560	013737	001414	001356		MOV	DAC,TEMP		:SAVE DAC
695	005566	012737	007776	001420		MOV	#7776,EDGE		:LOOK FOR 7775,7776 EDGE
696	005574	004537	007110			JSR	R5,SARSUB		
697	005600	000062				SO.			
698	005602	063737	001414	001356		ADD	DAC,TEMP		:ADD RESULTS
699	005610	162737	000400	001356		SUB	#400,TEMP		:OFFSET RESULT
700	005616	004737	005234			JSR	PC,TOFF		:TYPE GAIN
701	005622	004737	005256			JSR	PC,TCHK		:CHECK FOR CHARACTER AND DELAY
702	005626	000746				BR	2\$::
703	005630	000735				BR	1\$::NOT WITHIN TOLLERANCE, TRY AGAIN
704	005632	000005				RESET			
705	005634	000137	002262			JMP	BEG2		

```

707 .SBTTL PRINT VALUES ROUTINE
708 005640 012737 005640 001374 BEGINP: MOV #BEGINP,TADDR ;TEST ADDRESS IN TADDR
709 005646 005077 173444 CLR @STREG ;CLEAR STATUS REGISTER
710 005652 104401 013665 TYPE ,HEADS ;TYPE OUT HEADING
711 005656 005046 CLR -(SP) ;CLEAR PSW
712 005660 012746 005666 MOV #1$,-(SP)
713 005664 000002 RTI
714 005666 017700 173246 1$: MOV @SWR,R0 ;READ CHANNEL FROM SWITCH REG.
715 005672 042700 177700 BIC #177700,R0 ;ISOLATE MUX BITS
716 005676 032777 020000 173234 BIT #BIT13,@SWR ;IS BIT 13 SET?
717 005704 001005 BNE 2$ ;;YES,SKIP TYPEOUT
718 005706 104401 012131 TYPE ,CH
719 005712 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
(1) ;;TYPE CHANNEL
(1) 005714 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 005716 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
(1) 005717 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
720 005720 012777 001620 173376 2$: MOV #RETURN,@VECTOR ;ADDRESS AFTER INTRPT.
721 005726 000300 SWAB R0 ;SWITCH BYTES
722 005730 052700 000100 BIS #BIT6,R0
723 005734 010077 173356 MOV R0,@STREG ;LOAD THE CHANNEL
724 005740 012702 000010 MOV #10,R2 ;TYPEOUT COUNTER
725 005744 005277 173346 3$: INC @STREG ;START CONVERSION
726 005750 000001 WAIT ;WAIT FOR INTRPT.
727 005752 017700 173344 MOV @ADBUFF,R0 ;READ CONVERTED VALUE
728 005756 032777 020000 173154 BIT #BIT13,@SWR ;IS BIT 13 SET?
729 005764 001403 BEQ 4$ ;NOT SET, TYPE OUT LIST
730 005766 010077 173150 MOV R0,@DISPLAY ;PUT VALUE IN DISPLAY FOR DISPLAY CONTR
731 005772 000735 BR 1$ ;REPEAT CONVERSION
732 005774 104401 012134 4$: TYPE ,SPACE
733 006000 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
(1) ;;PRINT OCTAL CONVERTED VALUE
(1) 006002 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 006004 004 .BYTE 4 ;;TYPE 4 DIGIT(S)
(1) 006005 001 .BYTE 1 ;;TYPE LEADING ZEROS
734 006006 012701 010000 MOV #10000,R1
735 006012 005301 5$: DEC R1
736 006014 001376 BNE 5$
737 006016 005302 DEC R2 ;DECREMENT THE COUNTER
738 006020 001351 BNE 3$ ;NO CARRIAGE RETURN
739 006022 104401 001171 TYPE ,$CRLF ;CARRIAGE RETURN
740 006026 000717 BR 1$ ;REPEAT CONVERSION

```

```

742          .SBTTL          LOGIC TEST SECTION
743 006030 012737 006030 001374 BEGL:  MOV      #BEGL, TADDR      ; TEST ADDRESS
744 006036 004737 002464          JSR      PC,TESTAD      ; NO OF ADDITIONAL AD'S
745 006042 004737 002670          1$:  JSR      PC,BEGINL      ; LOGIC TESTS
746 006046 004737 006206          JSR      PC,BUMPAD      ; MORE TO TEST?
747 006052 000773          BR       1$              ; TEST NEXT A/D
748 006054 012737 006042 011646 MOV      #1$,AGTST      ; ADDRESS FOR EOP
749 006062 000137 011650          JMP      $EOP          ; TYPE END OF PASS
750
751          .SBTTL          AUTO TEST
752 006066 012737 006066 001374 BEGINA: MOV      #BEGINA, TADDR      ; TEST ADDRESS
753 006074 004737 002464          JSR      PC,TESTAD      ; NO. OF AD'S TO BE TESTED
754 006100 004737 002670          1$:  JSR      PC,BEGINL      ; LOGIC TESTS
755 006104 104401 013057          TYPE      ,MEND        ; TYPE END OF LOGIC TEST
756 006110 013746 001316          MOV      STREG,-(SP)    ; SAVE STREG FOR TYPEOUT
757 006114 104403          TYPOS
758 006116          .BYTE      6          ; TYPE 6 DIGITS
759 006117          .BYTE      1          ; TYPE LEADING ZEROS
760 006120 104401 001171          TYPE      ,$CR LF      ; TYPE A CR,LF
761 006124 004737 004126          JSR      PC,WRAP
762 006130 004737 006206          JSR      PC,BUMPAD      ; TEST NEXT A/D
763 006134 000761          BR       1$              ; TEST NEXT AD
764 006136 012737 006100 011646 MOV      #1$,AGTST      ; ADDRESS FOR EOP
765 006144 000137 011650          JMP      $EOP          ; TYPE END OF PASS
766
767          .SBTTL          WRAPAROUND TEST
768 006150 012737 006150 001374 BEGINW: MOV      #BEGINW, TADDR      ; TEST ADDRESS
769 006156 004737 002464          JSR      PC,TESTAD      ; NO. OF AD'S TO BE TESTED
770 006162 004737 004126          1$:  JSR      PC,WRAP      ; WRAPAROUND TESTS
771 006166 004737 006206          JSR      PC,BUMPAD      ; MORE A/D'S TO BE TESTED?
772 006172 000773          BR       1$              ; YES-GO TEST NEXT ADV11
773 006174 012737 006162 011646 MOV      #1$,AGTST
774 006202 000137 011650          JMP      $EOP          ; INCREMENTS $PASS
  
```



```
776          .SBTTL      DETERMINE IF MORE ADV11'S TO BE TESTED
777 006206 005737 001364  BUMPAD: TST      NBEXT          ;ADDITIONAL AD'S?
778 006212 001434          BEQ      FIXADR        ;NO-INITIALIZE ADDRESSES
779 006214 006337 001440          ASL      TSTBIT          ;MOVE BIT TO NEXT MODULE
780 006220 063737 001336 001316          ADD     VADR,STREG      ;SET UP NEW ST. REG.
781 006226 063737 001336 001320          ADD     VADR,ADST1     ;SET UP NEW ADST1
782 006234 063737 001336 001322          ADD     VADR,ADBUFF    ;SET UP NEW BUFFER ADDRESS
783 006242 063737 001340 001324          ADD     VVCT,VECTOR    ;SET UP NEW VECTOR
784 006250 063737 001340 001330          ADD     VVCT,VECTR1
785 006256 063737 001340 001332          ADD     VVCT,VECTR2
786 006264 063737 001340 001334          ADD     VVCT,VECTR3
787 006272 005077 173032          CLR     @VECTR1
788 006276 005337 001364          DEC     NBEXT          ;ONE LESS ADV11
789 006302 000473          BR      BYPASS
790 006304 062716 000002          FIXADR: ADD    #2,(SP)
791 006310 012737 000006 000004          FIXONE: MOV    #6,@ERRVEC          ;SET UP ERRVEC
792 006316 012737 007302 000010          MOV    #DELAY4,@RESVEC        ;SETUP RESERVED INST. VECTOR
793 006324 012737 000001 001440          MOV    #1,TSTBIT          ;INITIALIZE MODULE ERROR TEST BIT
794 006332 013737 001250 001316          MOV    $BASE,STREG        ;RELOAD INITIAL ADDRESSES
795 006340 013737 001250 001320          MOV    $BASE,ADST1
796 006346 013737 001250 001322          MOV    $BASE,ADBUFF
797 006354 005237 001320          INC     ADST1
798 006360 062737 000002 001322          ADD     #2,ADBUFF
799 006366 013737 001244 001324          MOV    $VECT1,VECTOR
800 006374 042737 170000 001324          BIC    #170000,VECTOR
801 006402 113737 001245 001326          MOVB   $VECT1+1,BASEBR
802 006410 105037 001327          CLRB   BASEBR+1          ;CLEAR HIGH BYTE
803 006414 013737 001324 001330          MOV    VECTOR,VECTR1
804 006422 062737 000002 001330          ADD     #2,VECTR1
805 006430 013737 001324 001332          MOV    VECTOR,VECTR2
806 006436 062737 000004 001332          ADD     #4,VECTR2
807 006444 013737 001324 001334          MOV    VECTOR,VECTR3
808 006452 062737 000006 001334          ADD     #6,VECTR3
809 006460 005077 172644          CLR     @VECTR1
810 006464 013737 001366 001364          MOV    NMBEXT,NBEXT      ;RESET COUNTER
811          ;;LOAD .+2 AND HALT TRAP CATCH;;
812 006472 012700 000216          BYPASS: MOV    #216,R0          ;FILL .+2
813 006476 012701 000214          MOV    #214,R1          ;LOAD HALT
814 006502 020137 001344          1$:    CMP    R1,KBVECT
815 006506 001410          BEQ    2$
816 006510 010021          MOV    R0,(R1)+
817 006512 005021          CLR    (R1)+
818 006514 010100          MOV    R1,R0
819 006516 005720          TST    (R0)+
820 006520 020027 001002          CMP    R0,#1002
821 006524 001366          BNE    1$
822 006526 000207          RTS    PC          ;TEST NEXT A/D
823 006530 022021          2$:    CMP    (R0)+,(R1)+
824 006532 022021          CMP    (R0)+,(R1)+
825 006534 000762          BR     1$
```

```

827 006536 104413          TYPSET: TYPDC
828 006540 104401 012141  TYPE      ,LSB
829 006544 013746 001362  MOV      CH2,-(SP)      ;;SAVE CH2 FOR TYPEOUT
(1)                                     ;;TYPE CH
(1) 006550 104403          TYPOS                                     ;;GO TYPE--OCTAL ASCII
(1) 006552      002        .BYTE      2                                     ;;TYPE 2 DIGIT(S)
(1) 006553      000        .BYTE      0                                     ;;SUPPRESS LEADING ZEROS
830 006554 104401 013777  TYPE      ,MAT                                     ;;TYPE ASCIIZ STRING
831 006560 004737 007044  JSR      PC,TYPEDG
832 006564 104401 012154  TYPE      ,SETCH
833 006570 013746 001360  MOV      CH1,-(SP)      ;;SAVE CH1 FOR TYPEOUT
(1)                                     ;;TYPE CH
(1) 006574 104403          TYPOS                                     ;;GO TYPE--OCTAL ASCII
(1) 006576      002        .BYTE      2                                     ;;TYPE 2 DIGIT(S)
(1) 006577      000        .BYTE      0                                     ;;SUPPRESS LEADING ZEROS
834 006600 104401 012176  TYPE      ,ATMSG
835 006604 013737 001360 006632  MOV      CH1,1$
836 006612 163737 001342 006632  SUB      BASECH,1$
837 006620 012777 000200 172474  MOV      #200,@ADBUFF
838 006626 004537 011036  JSR      R5,CONVRT
839 006632 000000          1$: 0
840 006634 013746 001356  MOV      TEMP,-(SP)      ;;SAVE TEMP FOR TYPEOUT
(1)                                     ;;TYPE VALUE
(1) 006640 104403          TYPOS                                     ;;GO TYPE--OCTAL ASCII
(1) 006642      004        .BYTE      4                                     ;;TYPE 4 DIGIT(S)
(1) 006643      001        .BYTE      1                                     ;;TYPE LEADING ZEROS
841 006644 020437 011630  CMP      R4,VSET
842 006650 003003          BGT      ERR
843 006652 104401 012245  TYPE      ,OKMSG
844 006656 000207          RTS      PC
845 006660 104401 012567  ERR:  TYPE      ,ERMSG
846 006664 000207          RTS      PC
847
848
849 006666 013737 001362 001370  ;;SUBROUTINE FOR SETTLING TESTS;;
850 006674 004537 007110  SET1A: MOV      CH2,DUMMY      ;LOAD DUMMY
851 006700 000062          JSR      R5,SARSUB      ;DO SAR ROUTINE AT 50%
852 006702 063702 001414          50.
853 006706 013737 001360 001370  ADD      DAC,R2      ;ADD RESULT TO R2
854 006714 004537 007110          MOV      CH1,DUMMY      ;CHANGE DUMMY VALUE
855 006720 000062          JSR      R5,SARSUB      ;DO SAR ROUTINE AT 50%
856 006722 163702 001414          50.
857 006726 000207          SUB      DAC,R2      ;SUBTRACT RESULT FROM R2
RTS      PC      ;RETURN

```

```
859 ;SUBROUTINE TO GET EDGE VALUE
860 ;CALL=JSR PC,GETEDG
861 ;CONVERSIONS ON A/D CHANNEL 'CHANL'
862 ;RESULT IN EDGE, USES R0
863 006730 012777 000200 172364 GETEDG: MOV #200,@ADBUFF ;LOAD VERNIER DAC
864 006736 113700 001372 MOVB CHANL,R0 ;GET CHANNEL
865 006742 000300 SWAB R0 ;SET UP A.D STATUS REG.
866 006744 052700 000100 BIS #100,R0 ;ENABLE INTPPT.
867 006750 010077 172342 MOV R0,@STREG
868 006754 012700 000100 MOV #100,R0 ;DAC SETTLING DELAY
869 006760 005300 1$: DEC R0
870 006762 001376 BNE 1$
871 006764 005037 001420 CLR EDGE
872 006770 012700 000010 MOV #10,R0
873 006774 012777 001620 172322 CONV: MOV #RETURN,@VECTOR ;RETURN ADDRESS
874 007002 005277 172310 INC @STREG ;START CONVERSION
875 007006 000001 WAIT ;WAIT FOR INTERRUPT
876 007010 067737 172306 001420 ADD @ADBUFF,EDGE
877 007016 005300 DEC R0
878 007020 001370 BNE CONV
879 007022 006237 001420 ASR EDGE
880 007026 006237 001420 ASR EDGE
881 007032 006237 001420 ASR EDGE
882 007036 005537 001420 ADC EDGE
883 007042 000207 RTS PC
884
885 ;:SUBROUTINE TO TYPE EDGE VALUES:;
886 007044 013703 001420 TYPEDG: MOV EDGE,R3
887 007050 010346 MOV R3,-(SP) ;:SAVE R3 FOR TYPEOUT
(1) ;:TYPE OCTAL VALUE OF EDGE
(1) 007052 104403 TYPOS ;:GO TYPE--OCTAL ASCII
(1) 007054 004 .BYTE 4 ;:TYPE 4 DIGIT(S)
(1) 007055 001 .BYTE 1 ;:TYPE LEADING ZEROS
888 007056 023727 007106 000001 CMP EDGFLG,#1
889 007064 001407 BEQ RET
890 007066 062703 000007 ADD #7,R3
891 007072 104401 012045 TYPE ,MINUS ;:TYPE ASCII STRING
892 007076 010346 MOV R3,-(SP) ;:SAVE R3 FOR TYPEOUT
(1) ;:TYPE EDGE VALUE
(1) 007100 104403 TYPOS ;:GO TYPE--OCTAL ASCII
(1) 007102 004 .BYTE 4 ;:TYPE 4 DIGIT(S)
(1) 007103 001 .BYTE 1 ;:TYPE LEADING ZEROS
893 007104 000207 RET: RTS PC
894 007106 000000 EDGFLG: 0
```

```
896 ;SUBROUTINE TO DO SUCCESSIVE APPROXIMATION ROUTINE
897 ;CALL=JSR R5,SARSUB
898 ; XXX;XXX=PERCENT
899 ;RESULT RETURNED IN 'DAC',USES R0,R1,R4
900 007110 012537 001432 SARSUB: MOV (R5)+,PERCNT ;GET PERCENT
901 007114 006337 001432 ASL PERCNT
902 007120 006337 001432 ASL PERCNT
903 007124 006337 001432 ASL PERCNT ;RESCALE PERCENT FOR 1600.
904 007130 006337 001432 ASL PERCNT ;POINTS PER BURST
905 007134 012737 000200 001422 SAR1: MOV #200,BITPNT ;INITIALIZE BIT POINTER AT MSB
906 007142 005037 001414 CLR DAC ;INITIALIZE DAC VALUE
907 007146 005000 TRY: CLR R0
908 007150 063737 001422 001414 ADD BITPNT,DAC ;TRY BIT
909 007156 013777 001414 172136 MOV DAC,@ADBUFF
910 007164 012701 003100 MOV #1600,R1 ;SET UP FOR 1600. CONVERSIONS
911 007170 113777 001370 172122 NXTCVT: MOVB DUMMY,@ADST1 ;PRESET MUX TO DUMMY CHANNEL
912 007176 012777 001620 172120 MOV #RETURN,@VECTOR ;RETURN ADDRESS
913 007204 052777 000101 172104 BIS #101,@STREG ;CONVERSION ON DUMMY CHANNEL
914 007212 000001 WAIT ;WAIT FOR INTERRUPT
915 007214 017704 172102 MOV @ADBUFF,R4 ;DUMMY READ
916 007220 013704 001372 MOV CHANL,R4
917 007224 000304 SWAB R4
918 007226 052704 000101 BIS #101,R4 ;INTERRUPT ENABLE START
919 007232 010477 172060 MOV R4,@STREG ;JUMP TO CHANNEL + START CONVERT
920 007236 000001 WAIT ;WAIT FOR INTERRUPT
921 007240 027737 172056 001420 CMP @ADBUFF,EDGE
922 007246 002001 BGE Z$
923 007250 005200 INC R0 ;COUNT RESULTS .LT. EDGE
924 007252 005301 Z$: DEC R1
925 007254 001345 BNE NXTCVT
926 007256 020037 001432 CMP R0,PERCNT
927 007262 003003 BGT SHIFT
928 007264 163737 001422 001414 SUB BITPNT,DAC ;TAKE THE BIT OUT
929 007272 006237 001422 SHIFT: ASR BITPNT
930 007276 001323 BNE TRY
931 007300 000205 RTS R5
932
933 ;*ROUTINE FOR PROCESSERS THAT CAN'T DO A SOB INSTRUCTION
934
935 007302 005300 DELAY4: DEC R0 ;DECREMENT R0, IS IT ZERO?
936 007304 001376 BNE DELAY4 ;NO
937 007306 000002 RTI ;RETURN
```

```

939      ;;DIFFERENTIAL LINEARITY SUBROUTINE;;
940 007310 104401 013202      DIFLIN: TYPE      ,MSG20
941 007314 013702 001376      MOV      RNA,R2      ;SET UP RANDOM NUMBER GENERATOR
942 007320 013704 001400      MOV      RNB,R4
943 007324 013705 001402      MOV      RNC,R5
944 007330 012700 017776      MOV      #BUFFER,R0
945 007334 012701 010000      MOV      #4096.,R1      ;4096 WORDS FOR HISTOGRAM
946 007340 005020      CLEAR1: CLR      (R0)+      ;CLEAR BUFFER AREA
947 007342 005301      DEC      R1
948 007344 001375      BNE     CLEAR1
949 007346 012700 017156      MOV      #DIST,R0      ;DISTRIBUTION BUFFER POINTER
950 007352 012701 000310      MOV      #200.,R1      ;200. WORDS FOR DISTRIBUTION
951 007356 005003      CLR      R3
952 007360 005037 001434      CLR     OUT
953 007364 005037 001346      CLR     WIDE
954 007370 005037 001350      CLR     NARROW
955 007374 005037 001352      CLR     FIRST
956 007400 005037 001354      CLR     SKIPST
957 007404 005020      CLEAR2: CLR     (R0)+      ;CLEAR DISTRIBUTION BUFFER AREA
958 007406 005301      DEC     R1
959 007410 001375      BNE     CLEAR2
960 007412 012700 000003      CHANNL: MOV     #3,R0      ;CHANNEL 3
961 007416 063700 001342      ADD     BASECH,R0
962 007422 000300      SWAB    R0      ;LOAD MUX BITS
963 007424 052700 000100      BIS     #100,R0
964 007430 010077 171662      MOV     R0,@STREG
965 007434 012737 001440 001416      MOV     #800.,DELAY      ;NOMINAL STATE WIDTH - 1 LSB
966 007442 012777 001630 171654      MOV     #RET1,@VECTOR
967 007450 012701 007776      AGAIN: MOV     #4094.,R1
968 007454 060402      NEXT:  ADD     R4,R2
969 007456 060502      ADD     R5,R2
970 007460 005502      ADC     R2
971 007462 060204      ADD     R2,R4
972 007464 060504      ADD     R5,R4
973 007466 005504      ADC     R4
974 007470 060205      ADD     R2,R5
975 007472 060405      ADD     R4,R5
976 007474 005505      ADC     R5
977 007476 042700 177770      BIC     #177770,R0      ;MASK IT TO 4 BITS ONLY
978 007502 001401      BEQ     CONVR
979 007504 077001      DELAY3: SOB    R0,DELAY3      ;STALL TIME
980 007506 005277 171604      CONVR:  INC     @STREG      ;START CONVERSION
981 007512 000001      WAIT
982 007514 000240      NOP
983 007516 017700 171600      MOV     @ADBUFF,R0      ;GET CONVERTED VALUE
984 007522 001416      BEQ     DELAY1      ;IGNORE IF =0
985 007524 020027 007777      CMP     R0,#7777      ;IGNORE IF =7777
986 007530 001416      BEQ     DELAY2
987 007532 006300      ASL     R0
988 007534 005260 017776      INC     BUFFER(R0)      ;MAKE HISTOGRAM
989 007540 100016      BPL     OKAY
990 007542 012760 077777 017776      MOV     #077777,BUFFER(R0)      ;PREVENT OVERFLOW
991 007550 000412      BR     OKAY

```

```
993 007552 005037 001356      NOTOK: CLR      TEMP
994 007556 000407              BR      OKAY
995 007560 020027 007777      DELAY1: CMP     RO,#7777      ;EQUALIZE LOOP TIME
996 007564 001400              BEQ     DELAY2              ;WITH DUMMY INSTR.
997 007566 005201              DELAY2: INC     R1
998 007570 005263 001356      INC     TEMP(R3)
999 007574 100766              BMI     NOTOK
1000 007576 005301              OKAY:  DEC     R1
1001 007600 001325              BNE     NEXT
1002 007602 005337 001416      AROUND: DEC    DELAY
1003 007606 001320              BNE     AGAIN
1004 007610 012700 007776      MOV     #4094.,RO
1005 007614 012701 020000      MOV     #BUFFER+2,R1
1006 007620 012102              READ:  MOV     (R1)+,R2      ;GET STATE WIDTH
1007 007622 006202              ASR     R2                  ;1 LSB - 800.
1008 007624 006202              ASR     R2
1009 007626 006202              ASR     R2
1010 007630 005502              ADC     R2                  ;1 LSB = 100.
1011 007632 020227 000310      CMP     R2,#200.           ;OUT OF RANGE?
1012 007636 002403              BLT     INRNGE
1013 007640 005237 001434      INC     OUT                  ;YES - INCREMENT COUNTER
1014 007644 000423              BR      TYPBAD
1015 007646 006302              INRNGE: ASL    R2
1016 007650 005262 017156      INC     DIST(R2)           ;MAKE STATE WIDTH DISTRIBUTION
1017 007654 006202              ASR     R2
1018 007656 020227 000062      CMP     R2,#50.           ;IS IT 1/2 LSB?
1019 007662 002007              BGE     NOTNAR
1020 007664 005237 001350      INC     NARROW
1021 007670 005702              TST     R2                  ;IS IT A SKIPPED STATE?
1022 007672 001002              BNE     31$
1023 007674 005237 001354      INC     SKIPST
1024 007700 000405              BR      TYPBAD
1025 007702 020227 000226      NOTNAR: CMP    R2,#150.     ;IS IT 1.5 LSB?
1026 007706 003425              BLE     LAST
1027 007710 005237 001346      INC     WIDE
1028 007714 005737 001352      TYPBAD: TST    FIRST
1029 007720 001004              BNE     60$
1030 007722 005237 001352      INC     FIRST
1031 007726 104401 012111      TYPE   ,STATE
1032 007732 010103              60$:  MOV     R1,R3
1033 007734 162703 020000      SUB     #BUFFER+2,R3
1034 007740 006203              ASR     R3
1035 007742 010346              MOV     R3,-(SP)           ;;SAVE R3 FOR TYPEOUT
(1)                                ;;TYPE STATE
(1) 007744 104403              TYPOS                                ;;GO TYPE--OCTAL ASCII
(1) 007746 004                .BYTE 4                            ;;TYPE 4 DIGIT(S)
(1) 007747 001                .BYTE 1                            ;;TYPE LEADING ZEROS
1036 007750 104401 012105      TYPE   ,DASH
1037 007754 104413              TYPDC
1038 007756 104401 012076      TYPE   ,LSBMSG
```

1040	007762	005300		LAST:	DEC	R0	
1041	007764	001315			BNE	READ	
1042	007766	112737	000177	014562	MOVB	#177,DECPNT	
1043	007774	013702	001354		MOV	SKIPST,R2	;GET NO. OF SKIPPED STATES
1044	010000	104413			TYPDC		;TYPE IT
1045	010002	104401	012604		TYPE	,SKPMSG	;TYPE MESSAGE
1046	010006	005737	001354		TST	SKIPST	
1047	010012	001403			BEQ	1\$	
1048	010014	104401	012567		TYPE	,ERMSG	;TYPE 'ERROR'
1049	010020	000402			BR	NAR	
1050	010022	104401	012245	1\$:	TYPE	,OKMSG	;TYPE #OK#
1051	010026	013702	001350	NAR:	MOV	NARROW,R2	;GET NO. OF NARROW STATES
1052	010032	104413			TYPDC		;TYPE IT
1053	010034	104401	012626		TYPE	,NARMSG	;TYPE MESSAGE
1054	010040	013702	001346		MOV	WIDE,R2	
1055	010044	063702	001434		ADD	OUT,R2	
1056	010050	104413			TYPDC		;TYPE NO. OF WIDE STATES
1057	010052	104401	012665		TYPE	,WIDMSG	;TYPE MESSAGE
1058	010056	013702	001434		MOV	OUT,R2	
1059	010062	104413			TYPDC		;TYPE NO. OF STATES OUTSIDE 2 LSB
1060	010064	104401	012724		TYPE	,OUTMSG	;TYPE MESSAGE
1061	010070	005737	001434		TST	OUT	
1062	010074	001403			BEQ	11\$	
1063	010076	104401	012567		TYPE	,ERMSG	;TYPE 'ERROR'
1064	010102	000402			BR	HALF	
1065	010104	104401	012245	11\$:	TYPE	,OKMSG	;TYPE 'OK'
1066	010110	013702	001350	HALF:	MOV	NARROW,R2	
1067	010114	063702	001346		ADD	WIDE,R2	
1068	010120	063702	001434		ADD	OUT,R2	
1069	010124	010200			MOV	R2,R0	
1070	010126	104413			TYPDC		;TYPE NO. OF STATES OUTSIDE LIMITS
1071	010130	112737	000056	014562	MOVB	#56,DECPNT	
1072	010136	104401	012757		TYPE	,HAFMSG	
1073	010142	020027	000051		CMP	R0,#41.	;COMPARE IT TO NOMINAL
1074	010146	003403			BLE	21\$	
1075	010150	104401	012567		TYPE	,ERMSG	;TYPE 'ERROR'
1076	010154	000402			BR	SWDIST	
1077	010156	104401	012245	21\$:	TYPE	,OKMSG	;TYPE 'OK'
1078	010162	005737	001410	SWDIST:	TST	FLAG	;VT55?
1079	010166	001426			BEQ	RELACC	
1080	010170	004737	010646		JSR	PC,DELCLR	;WAIT AWHILE, THEN CLEAR VT55
1081	010174	104401	013234		TYPE	,MSG16	
1082	010200	104401	014026		TYPE	,BUFF1	;TYPE BUFF1-PRINT GRID
1083	010204	012700	017156		MOV	#DIST,R0	;POINTER TO STATE WIDTH DISTRIBUTION
1084	010210	012701	000310		MOV	#200.,R1	;GO 200. TIMES UP TO 2 LSB
1085	010214	012002		NXTY1:	MOV	(R0)+,R2	
1086	010216	004737	011342		JSR	PC,LOADY	
1087	010222	005002			CLR	R2	
1088	010224	004737	011342		JSR	PC,LOADY	
1089	010230	005301			DEC	R1	
1090	010232	001370			BNE	NXTY1	
1091	010234	104401	013747		TYPE	,C2	;TYPE ASCII STRING
1092	010240	004737	010646		JSR	PC,DELCLR	

```

1094 ;CHANGE HISTOGRAM ERROR TO RELATIVE ACCURACY ERROR
1095
1096 010244 005001 RELACC: CLR R1 ;RUNNING ERROR = 0
1097 010246 005003 CLR R3 ;MAXIMUM ERROR = 0
1098 010250 104401 013617 TYPE ,MSG21
1099 010254 012700 020000 MOV #BUFFER+2,R0
1100 010260 011002 NXTSTA: MOV (R0),R2 ;STATE WIDTH = R2
1101 010262 162702 001440 SUB #800.,R2 ;STATE WIDTH ERROR IN R2
1102 010266 060201 ADD R2,R1 ;UPDATE RUNNING ERROR
1103 010270 010120 MOV R1,(R0)+ ;SAVE IN BUFFER
1104 010272 010104 MOV R1,R4 ;SAVE IN R4 ALSO
1105 010274 100001 BPL PLUS ;IS IT POSITIVE?
1106 010276 005404 NEG R4 ;NO - MAKE IT POSITIVE
1107 010300 020403 PLUS: CMP R4,R3 ;CHECK AGAINST PREVIOUS MAX. ERROR
1108 010302 003405 BLE NOTNEW ;NOT A NEW MAXIMUM
1109 010304 010403 MOV R4,R3 ;UPDATE MAXIMUM IN R3
1110 010306 010005 MOV R0,R5
1111 010310 162705 020000 SUB #BUFFER+2,R5
1112 010314 006205 ASR R5 ;R5=EDGE VALUE AT MAX. RELACC
1113 010316 020027 037774 NOTNEW: CMP R0,#BUFFER+8190. ;DONE?
1114 010322 001356 BNE NXTSTA ;NO - REPEAT
1115 010324 006203 ASR R3 ;RESCALE FROM 1 LSB = 800. SCALING
1116 010326 006203 ASR R3 ;TO 1 LSB - 100. SCALING
1117 010330 006203 ASR R3
1118 010332 005503 ADC R3
1119 010334 010302 MOV R3,R2
1120 010336 104413 TYPDC
1121 010340 104401 013644 TYPE ,LINEA
1122 010344 010546 MOV R5,-(SP) ;;SAVE R5 FOR TYPEOUT
(1) ;TYPE VALUE
(1) 010346 104403 TYPOS ;GO TYPE--OCTAL ASCII
(1) 010350 004 .BYTE 4 ;TYPE 4 DIGIT(S)
(1) 010351 001 .BYTE 1 ;TYPE LEADING ZEROS
1123 010352 104401 012243 TYPE ,SLASH ;PRINT '/'
1124 010356 005205 INC R5
1125 010360 010546 MOV R5,-(SP) ;;SAVE R5 FOR TYPEOUT
(1) ;TYPE VALUE
(1) 010362 104403 TYPOS ;GO TYPE--OCTAL ASCII
(1) 010364 004 .BYTE 4 ;TYPE 4 DIGIT(S)
(1) 010365 001 .BYTE 1 ;TYPE LEADING ZEROS
1126 010366 020337 011632 CMP R3,VLIN
1127 010372 003403 BLE 418
1128 010374 104401 012567 TYPE ,ERMSG
1129 010400 000402 BR 428
1130 010402 104401 012245 418: TYPE ,OKMSG
1131 010406 005737 001410 428: TST FLAG ;VT55?
1132 010412 001503 BEQ L02
1133 010414 012700 017776 MOV #BUFFER,R0
1134 010420 012701 010000 MOV #4096.,R1
    
```



```
1136 010424 011002          GETDAT: MOV      (R0),R2          ;GET RELATIVE ACCURACY ERROR SCALED 1LSB - 800.
1137 010426 006202          ASR      R2          ;RESCALE IT TO 1 LSB = 100.
1138 010430 006202          ASR      R2
1139 010432 006202          ASR      R2
1140 010434 005502          ADC      R2
1141 010436 062702 000166    ADD      #118.,R2          ;AND MOVE IT TO MID-SCREEN
1142 010442 010220          MOV      R2,(R0)+        ;PUT IT BACK INTO BUFFER
1143 010444 005301          DEC      R1
1144 010446 001366          BNE      GETDAT
1145 010450 012700 017776    MOV      #BUFFER,R0
1146 010454 012704 017776    MOV      #BUFFER,R4
1147 010460 012705 020000    MOV      #BUFFER+2,R5
1148 010464 012701 001000    MOV      #512.,R1
1149 010470 012702 000007    NXTB:   MOV      #7.,R2
1150 010474 012003          MOV      (R0)+,R3
1151 010476 010337 001424    MOV      R3,MIN          ;MINIMUM
1152 010502 010337 001430    MOV      R3,MAX          ;MAXIMUM
1153 010506 012003          NXTCMP: MOV      (R0)+,R3
1154 010510 020337 001424    CMP      R3,MIN
1155 010514 002002          BGE      MAXTST
1156 010516 010337 001424    MOV      R3,MIN          ;NEW MINIMUM
1157 010522 020337 001430    MAXTST: CMP      R3,MAX
1158 010526 003402          BLE      TSTB
1159 010530 010337 001430    MOV      R3,MAX          ;NEW MAXIMUM
1160 010534 005302          TSTB:   DEC      R2
1161 010536 001363          BNE      NXTCMP
1162 010540 013724 001424    MOV      MIN,(R4)+
1163 010544 013725 001430    MOV      MAX,(R5)+
1164 010550 022425          CMP      (R4)+,(R5)+    ;BUMP EACH ONCE MORE
1165 010552 005301          DEC      R1
1166 010554 001345          BNE      NXTB
1167 010556 104401 013142    TYPE     ,MSG18
1168 010562 104401 014054    TYPE     ,BUFF2        ;TYPE BUFF2
1169 010566 012700 017776    MOV      #BUFFER,R0
1170 010572 004737 010624    JSR      PC,LOAD
1171 010576 104401 013754    TYPE     ,C3          ;TYPE ASCII STRING
1172 010602 012700 020000    MOV      #BUFFER+2,R0
1173 010606 004737 010624    JSR      PC,LOAD
1174 010612 104401 013747    TYPE     ,C2          ;TYPE ASCII STRING
1175 010616 004737 010646    JSR      PC,DELCLR
1176 010622 000207          L02:    RTS      PC
1177 010624 012701 001000    LOAD:   MOV      #512.,R1
1178 010630 012002          LOAD0: MOV      (R0)+,R2
1179 010632 005720          TST      (R0)+
1180 010634 004737 011342    JSR      PC,LOADY
1181 010640 005301          DEC      R1
1182 010642 001372          BNE      LOAD0
1183 010644 000207          RTS      PC
```

K 4

```

1185 010646 032777 010000 170264 DELCLR: BIT #BIT12,@SWR ;TEST FOR HALT FOR DISPLAY
1186 010654 001402 BEQ 1$ ;:DON'T HALT FOR DISPLAY
1187 010656 000000 HALT
1188 010660 000407 BR 3$ ;;
1189 010662 005000 1$: CLR R0
1190 010664 012701 000020 MOV #20,R1 ;DELAY BEFORE CLEANING SCREEN
1191 010670 005300 2$: DEC R0
1192 010672 001376 BNF 2$
1193 010674 005301 DEC R1
1194 010676 001374 BNE 2$
1195 010700 104401 014074 3$: TYPE ,VTINIT
1196 010704 000207 RTS PC
1197 ;;TYPE RMS AND PEAK VALUES;;
1198 010706 104401 012203 TYPRP: TYPE ,NOI
1199 010712 005737 001404 TST RMS
1200 010716 100002 BPL POSRMS
1201 010720 005037 001404 CLR RMS ;RMS<0,SET RMS=0
1202 010724 005737 001406 POSRMS: TST PEAK
1203 010730 100002 BPL POSPEA
1204 010732 005037 001406 CLR PEAK ;PEAK<0,SET PEAK=0
1205 010736 013702 001404 POSPEA: MOV RMS,R2
1206 010742 104413 TYPDC
1207 010744 104401 013026 TYPE ,MESR ;TYPE " LSB RMS. "
1208 010750 013702 001406 MOV PEAK,R2
1209 010754 104413 TYPDC
1210 010756 104401 013041 TYPE ,MESR ;TYPE " LSB PEAK AT "
1211 010762 004737 007044 JSR PC,TYPEDG
1212 010766 104401 012213 TYPE ,CHAN ;TYPE " ON CHANNEL "
1213 010772 013746 001372 MOV ,CHAN,-(SP) ;;SAVE CHANL FOR TYPEOUT
(1) ;;TYPE CHANL
(1) 010776 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 011000 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
(1) 011001 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1214 011002 023737 001404 011624 CMP RMS,VNR ;WITHIN LIMITS?
1215 011010 003007 BGT ER
1216 011012 023737 001406 011626 CMP PEAK,VNP ;WITHIN LIMITS?
1217 011020 003003 BGT ER
1218 011022 104401 012245 TYPE ,OKMSG
1219 011026 000207 RTS PC
1220 011030 104401 012567 ER: TYPE ,ERMSG
1221 011034 000207 RTS PC
  
```

```

1223      ::ROUTINE TO AVERAGE 8 CONVERSIONS::
1224 011036 012500      CONVRT: MOV      (R5)+,R0      ;GET CHANNEL VALUE
1225 011040 063700 001342      ADD      BASECH,R0
1226 011044 010037 001372      MOV      R0,CHANL
1227 011050 000300      SWAB     R0
1228 011052 005037 001356      CLR      TEMP
1229 011056 010077 170234      MOV      R0,@STREG      ;LOAD CHANNEL INTO MIX BITS
1230 011062 012700 010000      MOV      #10000,R0
1231 011066 005300      2$: DEC      R0
1232 011070 001376      BNE     2$
1233 011072 012777 001620 170224      MOV      #RETURN,@VECTOR      ;LOAD VECTOR
1234 011100 012700 000010      MOV      #10,R0      ;SET UP COUNTER
1235 011104 152777 000101 170204 1$: BISH   #101,@STREG      ;SET INTRPT. EN., START CONV.
1236 011112 000001      WAIT     ;WAIT FOR CONVERSION
1237 011114 067737 170202 001356      ADD      @ADBUFF,TEMP      ;READ BUFFER
1238 011122 005300      DEC      R0
1239 011124 001367      BNE     1$      ;DO 8 TIMES
1240 011126 006237 001356      ASR     TEMP      ;AVERAGE VALUE
1241 011132 006237 001356      ASR     TEMP
1242 011136 006237 001356      ASR     TEMP
1243 011142 005537 001356      ADC     TEMP
1244 011146 000205      RTS     R5      ;RETURN
1245
1246      ::COMPARE $GDDAT AND $BDDAT::
1247 011150 012537 001124      COMPAR: MOV      (R5)+,$GDDAT      ;GET GOOD DATA
1248 011154 013537 001412      MOV      @(R5)+,$SPREAD      ;GET SPREAD
1249 011160 013737 001356 001126      MOV      TEMP,$BDDAT      ;GET BAD(ACTUAL) DATA
1250 011166 013701 001126      MOV      $BDDAT,R1
1251 011172 013700 001124      MOV      $GDDAT,R0
1252 011176 160100      SUB     R1,R0      ;GET DIFFERENCE
1253 011200 100001      BPL     7$
1254 011202 005400      NEG     R0
1255 011204 020037 001412      7$: CMP     R0,$SPREAD      ;COMPARE IT TO SPREAD
1256 011210 003001      BGT     10$      ;GO TO ERROR PRINTOUT
1257 011212 005725      TST     (R5)+
1258 011214 000205      10$: RTS     R5      ;BUMP RETURN POINTER AROUND ERROR CALL

```

M 4

```

1260      :: SUBROUTINE TO TYPE INTRPT. TST MSG.::
1261 011216 005737 001202      DUMP:  TST  SPASS
1262 011222 001021              BNE  20$
1263 011224 012737 011266 001110      MOV  #20$,SLPERR
1264 011232 012737 011266 001106      MOV  #20$,SLPADR
1265 011240 104401 014004              TYPE ,METST
1266 011244 010046              MOV  R0,-(SP)
      ;TYPE ASCII STRING
      ;SAVE R0 FOR TYPEOUT
      ;TYPE TEST NO.
      ;GO TYPE--OCTAL ASCII
      ;TYPE 2 DIGIT(S)
      ;SUPPRESS LEADING ZEROS
(1)
(1) 011246 104403              TYPOS
(1) 011250 002                .BYTE 2
(1) 011251 000                .BYTE 0
1267 011252 104401 013103              TYPE ,ONAD
1268 011256 013746 001316              MOV  STREG,-(SP)
      ;SAVE STREG FOR TYPEOUT
      ;TYPE BUS ADDRESS
      ;GO TYPE--OCTAL ASCII
      ;TYPE 6 DIGITS
      ;TYPE LEADING ZEROS
(1)
(1) 011262 104403              TYPOS
(1) 011264 006                .BYTE 6
(1) 011265 001                .BYTE 1
1269 011266 000207      20$:  RTS  PC
1270
1271 011270 005737 001202      DUMP:  TST  SPASS
1272 011274 001010              BNE  30$
1273 011276 012737 011316 001110      MOV  #30$,SLPERR
1274 011304 012737 011316 001106      MOV  #30$,SLPADR
1275 011312 104401 012230              TYPE ,DONE
1276 011316 000207      30$:  RTS  PC
  
```

```
1278 ;SUBROUTINE TO RESET & SET INTRPT. EN. ;
1279 011320 000005 RST: RESET
1280 011322 052777 000100 167614 BIS #100,@STKS
1281 011330 005046 CLR -(SP) ;CLEAR PSW
1282 011332 012746 011340 MOV #18,-(SP)
1283 011336 000002 RTI
1284 011340 000207 18: RTS PC
1285
1286 ;SUBROUTINE LOADY:
1287 011342 005702 LOADY: TST R2 ;ROUTINE TO LOAD VALUE INTO R2
1288 011344 100001 BPL PLUSR2 ;AS A V155 V-VALUE
1289 011346 005002 CLR R2
1290 011350 020227 000353 PLUSR2: CMP R2,#235.
1291 011354 002402 BLT LESS
1292 011356 012702 000353 MOV #235.,R2
1293 011362 010203 LESS: MOV R2,R3
1294 011364 042702 177740 BIC #177740,R2
1295 011370 052702 000040 BIS #40,R2
1296 011374 105777 167550 B10: TSTB @STPS ;PRINT CHARACTER
1297 011400 100375 BPL B10
1298 011402 110277 167544 MOVB R2,@STPB
1299 011406 006203 ASR R3
1300 011410 006203 ASR R3
1301 011412 006203 ASR R3
1302 011414 006203 ASR R3
1303 011416 006203 ASR R3
1304 011420 042703 177770 BIC #177770,R3
1305 011424 052703 000040 BIS #40,R3
1306 011430 105777 167514 B11: TSTB @STPS ;PRINT CHARACTER
1307 011434 100375 BPL B11
1308 011436 110377 167510 MOVB R3,@STPB
1309 011442 000207 RTS PC
```

```
1311      ;;SUBROUTINE TO TYPE DECIMAL VALUE;;
1312      ;;IN R2 AS X.XX;;
1313 011444 005702      DECTYP: TST      R2      ;TEST VALUE TO BE TYPED
1314 011446 100003      BPL      POS
1315 011450 104401 012045      TYPE      ,MINUS      ;TYPE MINUS SIGN
1316 011454 005402      NEG      R2
1317 011456 020227 001747      POS:    CMP      R2,#999.      ;>999. REPLACE IT WITH 999.
1318 011462 003402      BLE      OKAYD
1319 011464 012702 001747      MOV      #999.,R2
1320 011470 105037 014564      OKAYD: CLRB     ONES      ;CLEAR ONES
1321 011474 105037 014563      CLRB     TENS      ;CLEAR TENS
1322 011500 105037 014561      CLRB     HUNS      ;CLEAR HUNS
1323 011504 005702      TESTR2: TST     R2      ;CONVERT VALUE TO A DECIMAL VALUE
1324 011506 001424      BEQ     TYPOUT
1325 011510 005302      DEC     R2
1326 011512 105237 014564      INCB     ONES
1327 011516 123727 014564 000012      CMPB     ONES,#10.
1328 011524 001367      BNE     TESTR2
1329 011526 105037 014564      CLRB     ONES
1330 011532 105237 014563      INCB     TENS
1331 011536 123727 014563 000012      CMPB     TENS,#10.
1332 011544 001357      BNE     TESTR2
1333 011546 105037 014563      CLRB     TENS
1334 011552 105237 014561      INCB     HUNS
1335 011556 000752      BR      TESTR2
1336 011560 152737 000060 014561      TYPOUT: BISB     #60,HUNS      ;PREPARE FOR TYPOUT
1337 011566 152737 000060 014563      BISB     #60,TENS
1338 011574 152737 000060 014564      BISB     #60,ONES
1339 011602 104401 014561      TYPE     ,HUNS      ;TYPE VALUE
1340 011606 000002      RTI
1341 011610 000000      V0:     0      ;TOLERANCE VALUES FOR FUNCTIONAL TESTS
1342 011612 000002      V2:     2
1343 011614 000004      V4:     4
1344 011616 000012      V12:    12
1345 011620 000062      V50D:   50.
1346 011622 000326      V326:   326
1347
1348 011624 000050      VNR:    40.      ;.4 LSB,NORMAL LIMITS FOR SYSTEM
1349 011626 000310      VNP:    200.     ;2 LSB, INTEGRATION AND FIELD USE ON SPEC TESTS
1350 011630 000144      VSET:   100.     ;1 LSB
1351 011632 000175      VLIN:   125.     ;1.25 LSB
1352 011634 100000      BIT15
1353
1354 011636 052777 000100 167300      AGATST: BIS     #100,@$TKS
1355 011644 000137      JMP     @(PC)+
1356 011646 001644      AGTST:  BECIN
```

```
1358 .SBTTL END OF PASS ROUTINE
(1)
(2) ::*****
(1) :*INCREMENT THE PASS NUMBER ($PASS)
(1) :*IF THERES A MONITOR GO TO IT
(1) :*IF THERF ISN'T JUMP TO AGATST
(1)
(1) $EOP:
(2) 011650 NOP
(1) 011650 000240 CLR $TSTNM ;;ZERO THE TEST NUMBER
(1) 011652 005037 001102 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
(1) 011656 005037 001160 INC $PASS ;;INCREMENT THE PASS NUMBER
(1) 011662 005237 001202 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
(1) 011666 042737 100000 001202 DEC (PC)+ ;;LOOP?
(1) 011674 005327 $EOPCT: .WORD 1
(1) 011676 000001 BGT $DOAGN ;;YES
(1) 011700 003035 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
(1) 011702 012737 $ENDCT: .WORD 1
(1) 011704 000001 $ENDCT: $EOPCT
(1) 011706 011676 TYPE ,65$ ;;TYPE ASCIZ STRING
(3) 011710 104401 011716 BR 64$ ;;GET OVER THE ASCIZ
(3) 011714 000414 ;;65$: .ASCIZ <15><12>/ENDPASS GOOD UNITS /
(3) 64$:
(3) 011746 MOV GUNITS,-(SP) ;;SAVE GUNITS FOR TYPEOUT
(3) 011746 013746 001436 TYPBN ;;GO TYPE--BINARY ASCII
(3) 011752 104405 $GET42: MOV @42,R0 ;;GET MONITOR ADDRESS
(1) 011754 013700 000042 BEQ $DOAGN ;;BRANCH IF NO MONITOR
(1) 011760 001405 RESET ;;CLEAR THE WORLD
(1) 011762 000005 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
(1) 011764 004710 NOP ;;SAVE ROOM
(1) 011766 000240 NOP ;;FOR
(1) 011770 000240 NOP ;;ACT11
(1) 011772 000240 $DOAGN:
(1) 011774 JMP @(PC)+ ;;RETURN
(1) 011774 000137 $RTNAD: .WORD AGATST
(1) 011776 011636 $ENULL: .BYTE -1,-1,0
(1) 012000 377 377 000 .EVEN ;;NULL CHARACTER STRING
(1) 012004
```

MAINDEC-11-DVADA-B
CVADAB.P11 17-JUL-78

MACY11 30A(1052) 25-JUL-78 15:54 PAGE 39
00:00 ASCII MESSAGES

SEQ 0055

```

1360
1361 012004 005015 047516 051511 .SBTTL ASCII MESSAGES
      012012 020105 042524 052123 NOIMSG: .ASCIZ <15><12>/NOISE TEST/<15><12>
      012020 005015 000
1362 012023 015 051412 052105 SETMSG: .ASCIZ <15><12>/SETTLING TEST/<15><12>
      012030 046124 047111 020107
      012036 042524 052123 005015
      012044 000
1363 012045 055 000 MINUS: .BYTE 55,0
1364 012047 077 000 QUEST: .BYTE 77,0
1365 012051 136 101 040 AMSG: .BYTE 136,101,40,40,0
      012054 040 000
1366 012056 136 103 040 CMSG: .BYTE 136,103,40,40,0
      012061 040 000
1367 012063 136 107 015 GMSG: .BYTE 136,107,15,12,123,127,122,105,107,72,0
      012066 012 123 127
      012071 122 105 107
      012074 072 000
1368 012076 046040 041123 005015 LSBMSG: .ASCIZ / LSB/<15><12>
      012104 000
1369 012105 055 020055 000 DASH: .ASCIZ /-- /
1370 012111 02 040524 042524 STATE: .ASCIZ /STATE-- WIDTH/<15><12>
      012116 024455 053440 042111
      012124 044124 005015 000
1371 012131 103 000110 CH: .ASCIZ /CH/
1372 012134 020040 020040 000 SPACE: .ASCIZ / /
1373 012141 040 051514 020102 LSB: .ASCIZ / LSB ON CH/
      012146 047117 041440 000110
1374 012154 051440 052105 046124 SETCH: .ASCIZ / SETTLING FROM CH/
      012162 047111 020107 051106
      012170 046517 041440 000110
1375 012176 040440 020124 000 ATMSG: .ASCIZ / AT /
1376 012203 116 044517 042523 NOI: .ASCIZ /NOISE: /
      012210 020072 000
1377 012213 040 047117 041440 CHAN: .ASCIZ / ON CHANNEL /
      012220 040510 047116 046105
      012226 000040
1378 012230 020040 020040 047504 DONE: .ASCIZ / DONE/<15><12>
      012236 042516 005015 000
1379 012243 057 000 SLASH: .ASCIZ #/#
1380 012245 040 020040 047440 OKMSG: .ASCIZ / OK/<15><12>
      012252 006513 000012
1381 012256 005015 054524 042520 LCHAN: .ASCIZ <15><12>/TYPE CHANNEL & CR: /
      012264 041440 040510 047116
      012272 046105 023040 041440
      012300 035122 000040
1382 012304 005015 054524 042520 SEL: .ASCIZ <15><12>/TYPE 'D' FOR OFFSET, 'G' FOR GAIN & CR: /
      012312 021040 021117 043040
      012320 051117 047440 043106
      012326 042523 026124 021040
      012334 021107 043040 051117
      012342 043440 044501 020116
      012350 020046 051103 020072
      012356 000

```


1384	012357	015	040412	045104	XADJ: .ASCII	<15><12>/ADJUST R15/
	012364	051525	020124	030522		
	012372	065				
1385	012373	040	047506	020122	MOLSB: .ASCIZ	/ FOR 0.00 LSB ERROR/
	012400	027060	030060	046040		
	012406	041123	042440	051122		
	012414	051117	000			
1386	012417	015	044412	050116	IGND: .ASCII	<15><12>/INPUT A GROUND ON THE CHANNEL/
	012424	052125	040440	043440		
	012432	047522	047125	020104		
	012440	047117	052040	042510		
	012446	041440	040510	047116		
	012454	046105				
1387	012456	005015	054524	042520	CRWR: .ASCIZ	<15><12>/TYPE CR WHEN READY/<15><12>
	012464	041440	020122	044127		
	012472	047105	051040	040505		
	012500	054504	005015	000		
1388	012505	015	044412	050116	IVOLT: .ASCIZ	<15><12>/INPUT +5.115 VOLTS ON THE CHANNEL/
	012512	052125	025440	027065		
	012520	030461	020065	047526		
	012526	052114	020123	047117		
	012534	052040	042510	041440		
	012542	040510	047116	046105		
	012550	000				
1389	012551	015	040412	045104	YADJ: .ASCIZ	<15><12>/ADJUST R3/
	012556	051525	020124	031522		
	012564	000				
1390	012565	053	000		POSITV: .ASCIZ	/+/
1391	012567	040	025052	051105	ERMSG: .ASCIZ	/ **ERROR**/<15><12>
	012574	047522	025122	006452		
	012602	000012				
1392	012604	051440	044513	050120	SKPMSG: .ASCIZ	/ SKIPPED STATE(S)/
	012612	042105	051440	040524		
	012620	042524	051450	000051		
1393	012626	047040	051101	047522	NARMSG: .ASCIZ	# NARROW (< 1/2 LSB) STATE(S)#<15><12>
	012634	020127	036050	030440		
	012642	031057	046040	041123		
	012650	020051	052123	052101		
	012656	024105	024523	005015		
	012664	000				
1394	012665	040	044527	042504	WIDMSG: .ASCIZ	# WIDE (> 1 1/2 LSB) STATE(S)#<15><12>
	012672	024040	020076	020061		
	012700	027461	020062	051514		
	012706	024502	051440	040524		
	012714	042524	051450	006451		
	012722	000012				
1395	012724	051440	040524	042524	OUTMSG: .ASCIZ	/ STATE(S) WIDER THAN 2 LSB/
	012732	051450	020051	044527		
	012740	042504	020122	044124		
	012746	047101	031040	046040		
	012754	041123	000			

1397	012757	040	052123	052101	HAFMSG: .ASCIZ # STATE-WIDTH(S) OUTSIDE + OR - 1/2 LSB#
	012764	026505	044527	052104	
	012772	024110	024523	047440	
	013000	052125	044523	042504	
	013006	025440	047440	020122	
	013014	020055	027461	020062	
	013022	051514	000102		
1398	013026	046040	041123	051040	MESR: .ASCIZ / LSB RMS, /
	013034	051515	020054	000	
1399	013041	040	051514	020102	MESP: .ASCIZ / LSB PEAK AT /
	013046	042520	045501	040440	
	013054	020124	000		
1400	013057	015	042412	042116	MEND: .ASCII <15><12>/END OF LOGIC TESTS/
	013064	047440	020106	047514	
	013072	044507	020103	042524	
	013100	052123	123		
1401	013103	040	047117	040440	ONAD: .ASCIZ / ON ADV11 AT /
	013110	053104	030461	040440	
	013116	020124	000		
1402	013121	040	042101	030526	MSG50: .ASCIZ / ADV11'S FOUND/<15><12>
	013126	023461	020123	047506	
	013134	047125	006504	000012	
1403	013142	005012	025412	027461	MSG18: .ASCII <12><12><12>#+1/2 LSB#<15><12><12><12><12><12><12><12><12><12><12><12><1
	013150	020062	051514	006502	
	013156	005012	005012	005012	
	013164	005012	005012	005012	
1404	013172	030455	031057	051514	.ASCIZ \-1/2LSB\
	013200	000102			
1405					
1406					
1407	013202	044504	043106	051105	MSG20: .EVEN .ASCIZ /DIFFERENTIAL LINEARITY:/<15><12>
	013210	047105	044524	046101	
	013216	046040	047111	040505	
	013224	044522	054524	006472	
	013232	000012			

1409	013234	020040	020040	020040	MSG16: .ASCII /	STATE-WIDTH DISTRIBUTION/<15><12><12><12>
	013242	020040	020040	020040		
	013250	020040	020040	020040		
	013256	020040	052123	052101		
	013264	026505	044527	052104		
	013272	020110	044504	052123		
	013300	044522	052502	044524		
	013306	047117	005015	005012		
1410	013314	020040	020043	043117	.ASCII / # OF STATES/<12><12><12><12><12><12><12><12><12><12><12><12><12><12><12><	
	013322	051440	040524	042524		
	013330	005123	005012	005012		
	013336	005012	005012	005012		
	013344	005012	005012	005012		
	013352	005012				
1411	013354	020040	020040	020040	.ASCII /	STATE WIDTH (LSB)/<15>
	013362	020040	020040	020040		
	013370	020040	020040	020040		
	013376	020040	020040	020040		
	013404	020040	020040	020040		
	013412	020040	020040	020040		
	013420	020040	020040	020040		
	013426	020040	020040	020040		
	013434	051440	040524	042524		
	013442	053440	042111	044124		
	013450	024040	051514	024502		
	013456	005015				
1412	013460	030040	020040	020040	.ASCIZ # 0	1/2 1 1 1/2 2#
	013466	020040	020040	020040		
	013474	020040	020040	027461		
	013502	020062	020040	020040		
	013510	020040	020040	020040		
	013516	020040	020061	020040		
	013524	020040	020040	020040		
	013532	020040	030440	030440		
	013540	031057	020040	020040		
	013546	020040	020040	020040		
	013554	020040	031040	000		
1413	013561	015	052012	050131	MSG71: .ASCIZ <15><12>/TYPE LETTER & CR FOR TEST: /	
	013566	020105	042514	052124		
	013574	051105	023040	041440		
	013602	020122	047506	020122		
	013610	042524	052123	020072		
	013616	000				
1414	013617	122	046105	052101	MSG21: .ASCIZ /RELATIVE ACCURACY:/<15><12>	
	013624	053111	020105	041501		
	013632	052503	040522	054503		
	013640	006472	000012			
1415	013644	046040	041123	046440	LINEA: .ASCIZ / LSB MAXIMUM AT /	
	013652	054101	046511	046525		
	013660	040440	020124	000		
1416	013665	015	050012	044522	HEAD5: .ASCII <15><12>/PRINT VALUES--/	
	013672	052116	053040	046101		
	013700	042525	026523	055		

```

1418 013705 040 042523 020124 ASKCH: .ASCIZ / SET CHANNEL IN SWR LOW BYTE/<15><12>
      013712 044103 047101 042516
      013720 020114 047111 051440
      013726 051127 046040 053517
      013734 041040 052131 006505
      013742 000012
1419 013744 055033 000 CO: .ASCIZ <33><132>
1420 013747 033 015462 000110 C2: .ASCIZ <33><62><33><110> ;CLEAR GRAPH MODE AND HOME
1421 013754 000112 C3: .ASCIZ <112>
1422 013756 005015 043117 051506 MOFSET: .ASCIZ <15><12>/OFFSET =/
      013764 052105 036440 000
1423 013771 040 051514 020102 MLSB: .ASCIZ / LSB /
      013776 000
1424 013777 040 052101 000040 MAT: .ASCIZ / AT /
1425 014004 005015 042440 052116 METST: .ASCIZ <15><12>/ ENTERING TEST /
      014012 051105 047111 020107
      014020 042524 052123 000040
1426 014026 033 061 101 BUFF1: .BYTE 33,61,101,61,111,62,114,41,60,45,63,51,66,55,71,61,74,110,41,40,112,0
      014031 061 111 062
      014034 114 041 060
      014037 045 063 051
      014042 066 055 071
      014045 061 074 110
      014050 041 040 112
      014053 000
1427 014054 033 061 101 BUFF2: .BYTE 33,61,101,47,111,61,104,50,65,44,62,110,40,40,102,0
      014057 047 111 061
      014062 104 050 065
      014065 044 062 110
      014070 040 040 102
      014073 000
1428 014074 033 110 033 VTINIT: .BYTE 33,110,33,112,33,61,101,40,33,62,0 ;HOME & ERASE SCREEN & CLEAR GRA
      014077 112 033 061
      014102 101 040 033
      014105 062 000
  
```

1430	014107	015	005012	042115	HEAD1: .ASCII	<15><12><12>/MD-11-DVADA-B	ADV11 DIAGNOSTIC/<15><12>
	014114	030455	026461	053104			
	014122	042101	026501	020102			
	014130	020040	040440	053104			
	014136	030461	042040	040511			
	014144	047107	051517	044524			
	014152	006503	012				
1431	014155	012	035101	040440	.ASCII	<12>/A: AUTO TEST/	
	014162	052125	020117	042524			
	014170	052123					
1432	014172	005015	035103	041440	.ASCII	<15><12>/C: CALIBRATION/	
	014200	046101	041111	040522			
	014206	044524	047117				
1433	014212	005015	035120	050040	.ASCII	<15><12>/P: PRINT VALUES/	
	014220	044522	052116	053040			
	014226	046101	042525	123			
1434	014233	015	046012	020072	.ASCII	<15><12>/L: LOGIC/	
	014240	047514	044507	103			
1435	014245	015	053412	020072	.ASCII	<15><12>/W: WRAPAROUND/<15><12>	
	014252	051127	050101	051101			
	014260	052517	042116	005015			
	014266	000					
1436	014267	123	040524	052524	EM1: .ASCII	/STATUS REG. ERROR/	
	014274	020123	042522	027107			
	014302	042440	051122	051117			
	014310	000					
1437	014311	106	044501	042514	EM2: .ASCII	/FAILED TO INTERRUPT/	
	014316	020104	047524	044440			
	014324	052116	051105	052522			
	014332	052120	000				
1438	014335	125	042516	050130	EM3: .ASCII	/UNEXPECTED INTERRUPT/	
	014342	041505	042524	020104			
	014350	047111	042524	051122			
	014356	050125	000124				
1439	014362	051105	047522	020122	EM4: .ASCII	#ERROR ON A/D CHANNEL#	
	014370	047117	040440	042057			
	014376	041440	040510	047116			
	014404	046105	000				

1441	014407	105	051122	041520	DH1:	.ASCIZ	/ERRPC	STREG	EXPECTED	ACTUAL/		
	014414	051440	051124	043505								
	014422	042440	050130	041505								
	014430	042524	020104	041501								
	014436	052524	046101	000								
1442	014443	105	051122	041520	DH2:	.ASCIZ	/ERRPC	STREG	CHANNEL	NOMINAL	TOLERANCE	ACTUAL/
	014450	020040	052123	042522								
	014456	020107	020040	044103								
	014464	047101	042516	020114								
	014472	047040	046517	047111								
	014500	046101	020040	047524								
	014506	042514	040522	041516								
	014514	020105	040440	052103								
	014522	040525	000114									
1443	014526	051105	050122	020103	DH3:	.ASCIZ	/ERRPC	STREG	ACTUAL/			
	014534	020040	020040	051440								
	014542	051124	043505	020040								
	014550	020040	041501	052524								
	014556	046101	000									
144	014561	000										
1445	014562	056										
1446	014563	000										
1447	014564	000	000									
1448												
1449												
1450	014566	001116	001316	001124	DT1:	\$ERRPC	, STREG	, \$GDDAT	, \$BDDAT	, 0		
	014574	001126	000000									
1451	014600	001116	001316	001372	DT2:	\$ERRPC	, STREG	, CHANL	, \$GDDAT	, SPREAD	, \$BDDAT	, 0
	014606	001124	001412	001126								
	014614	000000										
1452	014616	001116	001316	001126	DT3:	\$ERRPC	, STREG	, \$BDDAT	, 0			
	014624	000000										
1453	014626	000000			DF1:	0						


```

(1) 015014 122723 000015      CMPB    #15,(R3)+      ;;CHECK FOR RETURN
(1) 015020 001356              BNE     28             ;;LOOP IF NOT RETURN
(1) 015022 105063 177777      CLRB   -1(R3)         ;;CLEAR RETURN (THE 15)
(1) 015026 104401 001172      TYPE   ,SLF           ;;TYPE A LINE FEED
(1) 015032 012603              MOV     (SP)+,R3      ;;RESTORE R3
(1) 015034 011646              MOV     (SP),-(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 015036 016666 000004 000002 MOV     4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
(1) 015044 012766 015056 000004 MOV     #STTYIN,4(SP)
(1) 015052 000002              RTI                    ;;RETURN
(1) 015054 000              98:   .BYTE 0         ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 015055 000              .BYTE 0               ;;TERMINATOR
(1) 015056 000010              STTYIN: .BLKB 8        ;;RESERVE 8 BYTES FOR TTY INPUT
(1) 015066 052536 005015 000      %CNTLU: .ASCIIZ /%U/<15><12> ;;CONTROL 'U'
(1) 015073 136 006507 000012      %CNTLG: .ASCIIZ /%G/<15><12> ;;CONTROL 'G'
(1) 015100 005015 053523 020122      %MSWR:  .ASCIIZ <15><12>/SWR = /
(1) 015106 020075 000
(1) 015111 040 047040 053505      %NEW:   .ASCIIZ / NEW = /
(1) 015116 036440 000040
  
```



```

(1) 015676 001001          BNE      6$          ;;BRANCH IF NO
(1) 015700 000000          HALT           ;;YES
(1) 015702          6$:   RTI              ;;RETURN
(1) 015702 000002          .SBTTL  ERROR MESSAGE TIMEOUT ROUTINE
1461
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 015704          $ERRTYP:
(1) 015704 104401 001171    TYPE      ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 015710 010046          MOV      R0,-(SP)        ;;SAVE R0
(1) 015712 005000          CLR      R0              ;;PICKUP THE ITEM INDEX
(1) 015714 153700 001114    BISB    @R0,$ITEMB,R0
(1) 015720 001004          BNE      1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)
(2) 015722 013746 001116    MOV      $ERRPC,-(SP)    ;;TYPE THE PC OF THE ERROR
(2)
(2) 015726 104402          TYP0C          ;;SAVE $ERRPC FOR TYPEOUT
(1) 015730 000426          BR       6$              ;;ERROR ADDRESS
(1) 015732 005300          1$:   DEC      R0          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 015734 006300          ASL     R0              ;;GET OUT
(1) 015736 006300          ASL     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1) 015740 006300          ASL     R0              ;;WORK FOR THE ERROR TABLE
(1) 015742 062700 001256    ADD     #SERRTB,R0      ;;FORM TABLE POINTER
(1) 015746 012037 015756    MOV     (R0)+,2$        ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 015752 001404          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1) 015754 104401          TYPE    'ERROR MESSAGE'
(1) 015756 000000          2$:   .WORD    0          ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 015760 104401 001171    TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 015764 012037 015774    3$:   MOV     (R0)+,4$    ;;PICKUP 'DATA HEADER' POINTER
(1) 015770 001404          BEQ     5$              ;;SKIP TYPEOUT IF 0
(1) 015772 104401          TYPE    'DATA HEADER'
(1) 015774 000000          4$:   .WORD    0          ;;'DATA HEADER' POINTER GOES HERE
(1) 015776 104401 001171    TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 016002 011000          5$:   MOV     (R0),R0      ;;PICKUP 'DATA TABLE' POINTER
(1) 016004 001004          BNE     7$              ;;GO TYPE THE DATA
(1) 016006 012600          6$:   MOV     (SP)+,R0     ;;RESTORE R0
(1) 016010 104401 001171    TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 016014 000207          RTS     PC              ;;RETURN
(1) 016016          7$:
(2) 016016 013046          MOV     @R0+,-(SP)      ;;SAVE @R0+ FOR TYPEOUT
(2) 016020 104402          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 016022 005710          TST     (R0)           ;;IS THERE ANOTHER NUMBER?
(1) 016024 001770          BEQ     4$              ;;BR IF NO
(1) 016026 104401 016034    TYPE    ,8$            ;;TYPE TWO(2) SPACES
(1) 016032 000771          BR      7$              ;;LOOP
(1) 016034 020040 000          8$:   .ASCII  / /          ;;TWO(2) SPACES
(1) 016040 016040          .EVEN
    
```



```
(1) 016224 000770 BR 7$ ;;LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 016226 112716 000040 8$: MOVB #' (SP) ;;REPLACE TAB WITH SPACE
(1) 016232 004737 016252 9$: JSR PC,$TYPEC ;;TYPE A SPACE
(1) 016236 132737 000007 016316 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
(1) 016244 001372 BNE 9$ ;;TAB STOP
(1) 016246 005726 TST (SP)+ ;;POP SPACE OFF STACK
(1) 016250 000724 BR 2$ ;;GET NEXT CHARACTER
(1) 016252 105777 162672 $TYPEC: STB @$STPS ;;WAIT UNTIL PRINTER IS READY
(1) 016256 100375 BPL $TYPEC
(1) 016260 116677 000002 162664 MOVB 2(SP),@$STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 016266 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
(1) 016274 001003 BNE 1$ ;;BRANC- IF NO
(1) 016276 105037 016316 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
(1) 016302 000406 BR $TYPEX ;;EXIT
(1) 016304 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
(1) 016312 001402 BEQ $TYPEX ;;BRANCH IF YES
(1) 016314 105227 INCB (PC)+ ;;COUNT THE CHARACTER
(1) 016316 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
(1) 016320 000207 $TYPEX: RTS PC
```

1464 .SBTTL APT COMMUNICATIONS ROUTINE

```
(1)
(2) ;*****
(1) 016322 112737 000001 016566 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
(1) 016330 112737 000001 016564 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
(1) 016336 000403 BR $ATYC
(1) 016340 112737 000001 016566 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) 016346 $ATYC:
(3) 016346 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 016350 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(1) 016352 105737 016564 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
(1) 016356 001450 BEQ 5$ ;;IF NOT: BR
(1) 016360 122737 000001 001214 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 016366 001031 BNE 3$ ;;IF NOT: BR
(1) 016370 132737 000100 001215 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 016376 001425 BEQ 3$ ;;IF NOT: BR
(1) 016400 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
(1) 016404 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 016412 005737 001174 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) 016416 001375 BNE 1$ ;;IF NOT: WAIT
(1) 016420 010037 001210 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
(1) 016424 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
(1) 016426 001376 BNE 2$
(1) 016430 163700 001210 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 016434 006200 ASR R0 ;;GET MESSAGE LNTH IN WORDS
(1) 016436 010037 001212 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
(1) 016442 012737 000004 001174 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 016450 000413 BR 5$
(1) 016452 017637 000004 016476 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) 016460 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 016466 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 016472 004737 016040 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 016476 000000 4$: .WORD 0
```

```

(1) 016500
(1) 016500 105737 016566
(1) 016504 001416
(1) 016506 005737 001214
(1) 016512 001413
(1) 016514 005737 001174
(1) 016520 001375
(1) 016522 017637 000004 001176
(1) 016530 062766 000002 000004
(1) 016536 005237 001174
(1) 016542 105037 016566
(1) 016546 105037 016565
(1) 016552 105037 016564
(3) 016556 012601
(3) 016560 012600
(1) 016562 000207
(1) 016564 000
(1) 016565 000
(1) 016566 000
(1) 016570
(1) 000200
(1) 000001
(1) 000100
(1) 000040

5$:
10$: TSTB $FFLG :: SHOULD REPORT FATAL ERROR?
      BEQ 12$ :: IF NOT: BR
      TST $ENV :: RUNNING UNDER APT?
      BEQ 12$ :: IF NOT: BR
11$: TST $MSGTYPE :: FINISHED LAST MESSAGE?
      BNE 11$ :: IF NOT: WAIT
      MOV @4(SP), $FATAL :: GET ERROR #
      ADD #2, 4(SP) :: BUMP RETURN ADDR.
      INC $MSGTYPE :: TELL APT TO TAKE ERROR
12$: CLRB $FFLG :: CLEAR FATAL FLAG
      CLRB $LFLG :: CLEAR LOG FLAG
      CLRB $MFLG :: CLEAR MESSAGE FLAG
      MOV (SP)+, R1 :: POP STACK INTO R1
      MOV (SP)+, R0 :: POP STACK INTO R0
      RTS PC :: RETURN
      $MFLG: .BYTE 0 :: MESSG. FLAG
      $LFLG: .BYTE 0 :: LOG FLAG
      $FFLG: .BYTE 0 :: FATAL FLAG
      .EVEN

APTSIZE=200
APTENV=001
APTSPOOL=100
APTCSUP=040
  
```


(1)	016732	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
(1)	016734	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
(1)	016740	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
(1)	016744	110337	017010		MOVB	R3,8\$::SAVE FOR TYPING
(1)	016750	104401	017010		TYPE	,8\$::GO TYPE THIS DIGIT
(1)	016754	105337	017012	7\$:	DECB	\$OCNT	::COUNT BY 1
(1)	016760	003347			BGT	2\$::BR IF MORE TO DO
(1)	016762	002402			BLT	6\$::BR IF DONE
(1)	016764	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
(1)	016766	000744			BR	2\$::GO DO THE LAST DIGIT
(1)	016770	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
(1)	016772	012604			MOV	(SP)+,R4	::RESTORE R4
(1)	016774	012603			MOV	(SP)+,R3	::RESTORE R3
(1)	016776	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
(1)	017004	012616			MOV	(SP)+,(SP)	
(1)	017006	000002			RTI		::RETURN
(1)	017010	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
(1)	017011	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
(1)	017012	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
(1)	017013	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
(1)	017014	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE


```

1470      .SBTTL  TRAP DECODER
(1)
(2)
(1)      ::*****
(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) 017072 010046      $TRAP:  MOV    RO,-(SP)          ;;SAVE RO
(1) 017074 016600 000002      MOV    2(SP),RO        ;;GET TRAP ADDRESS
(1) 017100 005740      TST    -(RO)          ;;BACKUP BY 2
(1) 017102 111000      MOVB   (RO),RO        ;;GET RIGHT BYTE OF TRAP
(1) 017104 006300      ASL    RO             ;;POSITION FOR INDEXING
(1) 017106 016000 017126      MOV    $TRPAD(RO),RO  ;;INDEX TO TABLE
(1) 017112 000200      RTS    RO             ;;GO TO ROUTINE
(1)
(1)
(1)      ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
(1)
(1) 017114 011646      $TRAP2: MOV   (SP),-(SP)  ;;MOVE THE PC DOWN
(1) 017116 016666 000004 000002      MOV   4(SP),2(SP)      ;;MOVE THE PSW DOWN
(1) 017124 000002      RTI                    ;;RESTORE THE PSW
(1)
(1)      .SBTTL  TRAP TABLE
(3)
(3)      ::THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)      ::BY THE 'TRAP' INSTRU TION.
(3)
(3)      :
(3)      : ROUTINE
(3)      : -----
(3) 017126 017114      $TRPAD: .WORD  $TRAP2
(3) 017130 016040      $TYPE   ::CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3) 017132 016614      $TYPOC  ::CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 017134 016570      $TYPOS  ::CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 017136 016630      $TYPON  ::CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 017140 017016      $TYPBN  ::CALL=TYPBN     TRAP+5(104405)  TYPE BINARY (ASCII) NUMBER
(1)
(1)
(3) 017142 014630      $RDCHR  ::CALL=RDCHR     TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
(3) 017144 014750      $RDLIN  ::CALL=RDLIN     TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
(3) 017146 015122      $RDOCT  ::CALL=RDOCT     TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
1471 017150 004102      TEST    ::CALL=CHECK    TRAP+11(104411)
1472 017152 004074      TESTIT  ::CALL=CHKIT     TRAP+12(104412)
1473 017154 011444      DECTYP  ::CALL=TYPDC     TRAP+13(104413)

```

1475			.EVEN		
1476	017156	000310	DIST: .BLKW	200.	;STATE-WIDTH DISTRIBUTION
1477	017776	010000	BUFFER: .BLKW	4096.	;BUFFER AREA
1478					
1479		000001	.END		

EM3	014335	64	1438#						
EM4	014362	70	1439#						
ER	011030	1215	1217	1220#					
ERMSG	012567	530	845	1048	1063	1075	1128	1220	1391#
ERR	006660	842	845#						
ERRVEC=	000004	21#	163*	249	250*	265*	791*	1459*	
FIRST	001352	90#	955*	1028	1030*				
FIXADR	006304	778	790#						
FIXONE	006310	187	791#						
FLAG	001410	105#	165*	182*	1078	1131			
GETDAT	010424	1136#	1144						
GETEDG	006730	566	595	863#					
GMSG	012063	145	1367#						
GNS =	***** U	30	1358	1470	1471	1472	1473		
GUNITS	001436	116#	239*	254*	1358	1460*			
HAFMSG	012757	1072	1397#						
HALF	010110	1064	1066#						
HEAD1	014107	185	1430#						
HEAD5	013665	710	1416#						
HT -	000011	21#	1463						
HUNS	014561	1322*	1334*	1336*	1339	1444#			
IGND	012417	670	1386#						
INRNGE	007646	1012	1015#						
IOTVEC=	000020	21#	163*						
ISERV	001464	126#	189						
IVOLT	012505	683	1388#						
KBVECT	001344	87#	188	814					
LAST	007762	1026	1040#						
LEND	005156	613	615#						
LESS	011362	1291	1293#						
LF =	000012	21#	1463						
LINEA	013644	1121	1415#						
LOAD	010624	1170	1173	1177#					
LOADY	011342	1086	1088	1180	1287#				
LOADO	010630	1178#	1182						
LO2	010622	1132	1176#						
LSB	012141	828	1373#						
LSBMSG	012076	1038	1368#						
MAT	013777	830	1424#						
MAX	001430	113#	1152*	1157	1159*	1163			
MAXTST	010522	1155	1157#						
MEND	013057	755	1400#						
MESP	013041	1210	1399#						
MESR	013026	1207	1398#						
METST	014004	1265	1425#						
MIN	001424	111#	1151*	1154	1156*	1162			
MINUS	012045	891	1315	1363#					
MLSB	013771	632	1423#						
MOFSET	013756	523	1422#						
MSG16	013234	1081	1409#						
MSG18	013142	1167	1403#						
MSG20	013202	940	1407#						
MSG21	013617	1098	1414#						
MSG50	013121	262	1402#						
MSG71	013561	201	1413#						
MOLSB	012373	688	1385#						

NAR	010026	1049	1051#							
NARMSG	012626	1053	1393#							
NARROW	001350	89#	954*	1020*	1051	1066				
NBEXT	001364	95#	251*	253*	261	263*	264	777	788*	810*
NEXT	007454	968#	1001							
NMBEXT	001366	96#	264*	810						
NOI	012203	1198	1376#							
NOIMSG	012004	563	1361#							
NONE	001636	142	144	155#						
NOTNAR	007702	1019	1025#							
NOTNEW	010316	1108	1113#							
NOTOK	007552	993#	999							
NOVT55	002212	173	176	181	185#	233				
NXTCMP	010506	1153#	1161							
NXTCVT	007170	911#	925							
NXTSTA	010260	1100#	1114							
NXTY1	010214	1085#	1090							
NXT8	010470	1149#	1166							
OFFERR	004464	528	530#							
OFFOK	004472	529	532#							
OFFSET	005160	522	617#	676						
OKAY	007576	989	991	994	1000#					
OKAYD	011470	1318	1320#							
OKMSG	012245	532	843	1050	1065	1077	1130	1218	1380#	
ONAD	013103	1267	1401#							
ONES	014564	1320*	1326*	1327	1329*	1338*	1447#			
OUT	001434	115#	952*	1013*	1055	1058	1061	1068		
OUTMSG	012724	1060	1395#							
PEAK	001406	104#	568*	577*	580*	1202	1204*	1208	1216	
PERCNT	001432	114#	900*	901*	902*	903*	904*	926		
PIRQ =	177772	21#								
PIRQVE=	000240	21#								
PLUS	010300	1105	1107#							
PLUSR2	011350	1288	1290#							
POPPO	001616	150#	156							
POS	011456	1314	1317#							
POSITV	012565	630	1390#							
POSPEA	010736	1203	1205#							
POSRMS	010724	1200	1202#							
PRO =	000000	21#								
PR1 =	000040	21#								
PR2 =	000100	21#								
PR3 =	000140	21#								
PR4 =	000200	21#								
PR5 =	000240	21#								
PR6 =	000300	21#								
PR7 =	000340	21#								
PS =	177776	21#								
PSW =	177776	21#								
PURVEC=	000024	21#								
QUEST	012047	155	224	1364#						
RBEG	001660	160	162#							
RDCHR =	104406	1455	1470#							
RDLIN =	104407	202	661	671	685	1457	1470#			
RDOCT =	104410	148	657	1470#						
READ	007620	1006#	1041							

.SWRLO	22#	
.SACT1	10#	34
.SAPT8	10#	37#
.SAPTH	10#	36
.SAPTY	10#	1464
.SCATC	7#	30
.SCMTA	7#	37
.SEOP	7#	1358
.SERRO	7#	1460
.SERRT	9#	1461
.SPARM	8#	
.SPOWE	8#	
.SRAND	10#	
.SRDOC	10#	1457
.SREAD	8#	1455
.SSAVE	8#	
.SSCOP	8#	1459
.SSPAC	9#	
.SSWDO	9#	
.STRAP	9#	1470
.STYP8	8#	1468
.STYPD	10#	
.STYPE	9#	1463
.STYPO	8#	1466

. ABS. 037776 000

ERRORS DETECTED: 0

CVADAB, CVADAB/CRF=CVADAB
RUN-TIME: 28 11 1 SECONDS
RUN-TIME RATIO: 326/41-7.8
CORE USED: 26K (51 PAGES)

C	1		K	5	MAINDE
D	1		L	5	MAINDE
E	1		M	5	MAINDE
F	1		N	5	MAINDE
G	1		B	6	MAINDE
H	1		C	6	MAINDE
I	1		D	6	MAINDE
J	1		E	6	MAINDE
K	1	MAINDE	F	6	MAINDE
L	1	MAINDE	G	6	MAINDE
M	1	MAINDE	H	6	MAINDE
N	1	MAINDE	I	6	MAINDE
B	2	MAINDE	J	6	MAINDE
C	2	MAINDE	K	6	MAINDE
D	2	MAINDE	L	6	MAINDE
E	2	MAINDE	M	6	MAINDE
F	2	MAINDE	N	6	MAINDE
G	2	MAINDE	B	7	MAINDE
H	2	MAINDE	C	7	MAINDE
I	2	MAINDE	D	7	MAINDE
J	2	MAINDE	E	7	MAINDE
K	2	MAINDE	F	7	MAINDE
L	2	MAINDE	G	7	MAINDE
M	2	MAINDE	H	7	MAINDE
N	2	MAINDE	I	7	MAINDE
B	3	MAINDE	J	7	MAINDE
C	3	MAINDE	K	7	MAINDE
D	3	MAINDE			
E	3	MAINDE			
F	3	MAINDE			
G	3	MAINDE			
H	3	MAINDE			
I	3	MAINDE			
J	3	MAINDE			
K	3	MAINDE			
L	3	MAINDE			
M	3	MAINDE			
N	3	MAINDE			
B	4	MAINDE			
C	4	MAINDE			
D	4	MAINDE			
E	4	MAINDE			
F	4	MAINDE			
G	4	MAINDE			
H	4	MAINDE			
I	4	MAINDE			
J	4	MAINDE			
K	4	MAINDE			
L	4	MAINDE			
M	4	MAINDE			
N	4	MAINDE			
B	5	MAINDE			
C	5	MAINDE			
D	5	MAINDE			
E	5	MAINDE			
F	5	MAINDE			
G	5	MAINDE			
H	5	MAINDE			
I	5	MAINDE			