

MNCAD, MNCAG

MNCAD DIAG
CVMNAB0

AH-B086B-MC

COPYRIGHT '78-80

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

Table with multiple columns and rows of data, likely a technical specification or diagnostic chart. The content is extremely faint and illegible.

5

IDENTIFICATION

B 1

SEQ 0001

Product Code: AC-B085B-MC
Product Name: CVMNABO MNCAD Performance Test
Date: July 1979
Maintainer: Diagnostic Group

Copyright (C) 1978,1979

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

JNIBUS
DECTAPE

MASSBUS

1.0 ABSTRACT

This diagnostic has three starting addresses:

200 Normal
 204 Restart
 210 Option checkout with tester connected

This diagnostic tests the MNCAD/MNCAM/MNCAG with or without the optional test module(s).

When starting the diagnostic, the operator is asked about the presence of the test modules, clock and the type of console terminal. A list of tests available are displayed. The operator selects the test by the 'TEST CHARACTER' and then depresses the 'RETURN' key on the console. The following list indicates which 'TEST CHARACTER' corresponds to the test or function to execute:

W: Wraparound analog tests

Analog subtests
 Noise test
 Interchannel Settling test
 Differential Linearity and Relative Accuracy test
 (after the first pass)

C: Calibration loop for the MNCAD
 P: Print converted analog values loop
 L: Logic test (MNCAD and MNCAG)
 A: Auto tests

Logic subtests
 Analog subtests
 Noise test
 Interchannel Settling test
 Differential Linearity and Relative Accuracy test
 (after the first pass)

N: Noise tests on selected channels
 D: Differential Linearity and Relative Accuracy test on a selected channel
 S: Settling test between two selected channels
 F: Function test of the MNCAG front panel
 T: Test MNCAG channels analog input
 M: Common mode rejection test for MNCAG channels
 B: Base or vector address change
 G: Get new switch register value
 H: Help the operator and re-type the test list

2.0 REQUIREMENTS

2.1 Equipment

Computer with 16K of memory
 I/O Terminal (LA36, VT100, etc.)
 MNCAD/MNCAM/MNCAG Module(s)
 MNCAD-TA test module <optional>
 MNCAM-TA test module <optional>
 MNCAG-TA test module <optional>
 Bit map for graphic output (I.E. VT105, VT55) <optional>

2.2 Storage

This program uses 16K of memory.

3.0 LOADING PROCEDURE

Procedure for loading normal binary file should be followed.

4.0 STARTING PROCEDURE

4.1 Control Switch Settings

Standard PDP-11 Format

SW15=1	100000	Halt on error
SW14=1	040000	Loop on test
SW13=1	020000	Inhibit error typeouts
SW12=1	010000	Inhibit sizing the number of MNCAD (A/D)'S
SW11=1	004000	Inhibit iterations
SW10=1	002000	halt for video bit map display
SW9 =1	001000	Loop on error
SW8 =1	000400	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 when the tester is connected and tighter tolerances are used.

5.0 OPERATING PROCEDURE

Start the diagnostic at 200 or 210. The program requests an initial switch register value. The operator will normally depress the 'RETURN' key. The program now request if the MNCAD-TA test module is connected. The operator responds by typing a 'Y' or 'N' followed by depressing the 'RETURN' key. The request is repeated for the MNCAM-TA and MNCAG-TA test modules. The program will then request if a MNCCKW (CLOCK) is available on the system. The final request asks if the console terminal is a 'BIT-MAP' terminal (IE VT105 or VT55). A list of tests, loops, or functions available will be printed out. The operator selects the 'TEST CHARACTER', according to the table listed, and depresses the 'RETURN' key.

A control character (^C) is set aside for interrupting a test and transferring control to the beginning of the diagnostic. During the logic tests, while a 'RESET' is being performed, control C will not be executed. Therefore, continue typing control C 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, depress 'CTRL' and 'G' together or select the 'G' function. The program responds with the current contents of SWREG and a slash. Type the desired new contents of SWREG followed by a carriage return.

When a 'W' is selected, the program will report the number of MNCAD detected and will then give a channel table for the MNCAD (A/D) under test. If any test modules are connected, the program will then ask which channels to test. The program will run through the analog subtests, the noise test, the interchannel settling test, and after the first pass, the differential linearity and relative accuracy test.

If 'C' is typed, the program will ask for the channel to be used. It will then ask if the offset or gain adjustments are to be made. The program will run the calibration routine and loop on the MNCAD until it is calibrated and a 'RETURN' is typed. If an additional MNCAD (A/D) is to be calibrated, use the 'B' command to inform the program of its base and vector address.

If 'P' is typed, the program will ask for the channel to be used. It will then ask for the GAIN to be used for that channel. The program will then run the print values routine and will loop on that test until the operator type 'CTRL C' to change the selected channel or gain, the operator must type 'CTRL G'. The current switch register value will be reported. Bits 6 and 7 select the gain and bits 0 thru 5 select the channel to be used. If an additional MNCAD is to be tested, use the 'B' command to inform the program of its base and vector address. If 'A' is typed, the program will report the number of MNCAD detected and will then give a channel table for the MNCAD (A/D) under test. If any test module is connected, the program will then ask which channels to test. The program will run through the logic test for the MNCAD and MNCAG, analog subtests, the noise test, the interchannel settling test, and after the first pass, the differential linearity and relative accuracy tests.

If 'L' is typed, the program will then size the number of MNCAD (A/D)'S and report the number of units found. It will then give a channel table for the current MNCAD under test. The program will then execute the logic tests, printing 'END PASS' when it has completed an entire pass. If additional MNCAD (A/D)'S are detected, the test will be run successively on each MNCAD. If the MNCAD-TA test module is connected, the program will ask the operator to depress the test module 'EXTERNAL START' switch on the first pass.

If 'N' is typed, the program will report the number of MNCADS detected and will then give a channel table for the MNCAD under test. The program will then ask for the 'STARTING CHANNEL'. The operator now inputs the desired channel and depresses the 'RETURN'. The program will now ask for the 'ENDING CHANNEL'. The operator now inputs the last channels to be tested. If only one channel is desired, depress 'RETURN' for this answer. The program will now run the noise test on the selected channels. If the channel is a MNCAG channel, the noise test is repeated at each different gain.

If 'S' is typed, the program will report the number of MNCAD detected and will then give a channel table for the MNCAD under test. The program will then ask for the two channels that are to be tested. It is important that the two channels are at opposite input values (IE 0250 AND 7540).

IF 'D' IS TYPED, THE PROGRAM WILL REPORT THE NUMBER OF MNCAD detected and will then give a channel table for the MNCAD under test. The program will then ask for the 'STARTING CHANNEL'. The operator now inputs the desired channel and depresses the 'RETURN'. The program will now ask for the 'ENDING CHANNEL'. The operator now inputs the last channel to be tested. If only one channel is desired, depress 'RETURN' for this answer. The test requires that channels to be run must have a 'FULL RANGE RAMP' input.

If 'M' is typed, the program will request which channel will be used. The operator is now instructed to apply "+10 volts" to the channel input. The operator is then instructed to apply "-10 volts" to the channel input. The program will now report the results of the test.

If 'F' is typed, the program will request which channel will be used. The operator is now instructed what position to set the front panel switches. No analog input values will not be checked, only the front panel switches and digital read-back logic.

If 'T' is typed, the program will request which channel will be used. The operator is now instructed what position to set the front panel and MNCAG-TA switches. The analog input values will be tested for all gains and modes.

If 'H' is typed, the program will tell the operator what position to set the front panel and test module switches. It will then ask about the presence of the test modules, clock and the type of console terminal. The program will then type the list of tests available.

If 'B' is typed, the program will ask for the new bus address of the MNCAD. After the new address has been selected, the new vector address is requested. Upon completion of the input, the program will re-prompt the operator about the test to be run.

If 'G' is typed, the program will ask for the new switch register value. Upon completion of the new value, the program will re-prompt the operator about the tests to be run.

5.1 Inhibiting auto-size feature

Logic, auto and wraparound tests will automatically auto-size and report the number of MNCAD'S it detects on the system. To inhibit this feature, set switch register bit 12 to a one. Another way to inhibit this feature is to set bit 15 of location SENVM (1210). The operator can also use the program 'B' command to modify the default base and vector addresses for other than the first MNCAD.

5.2 End of pass typeouts

At the end of a pass in which no errors were detected, the following typeout will occur:

```
'END PASS 12'
```

At the end of a pass in which errors were detected, the following typeout will occur:

```
'END PASS 12 ;TOTAL ERROR COUNT = 5 ;BAD UNITS 000000000000100'
```

This indicates that:

Twelve passes thru the program have been made.
A total of 5 errors have been detected.
Unit # 3 was the unit with errors.

6.0 ERRORS

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

```
UNIT:      Unit number
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:	64 conversions/8 seconds @ 110 baud	
Wraparound test:	7 minutes first pass; 22 minutes	
	for successive passes	
Logic test:	30 seconds	
Auto test:	8 minutes first pass, 23 minutes	
	for successive passes	
Noise test:	20 seconds per selected channel	
Differential Linearity	14 minutes	
Settling test:	15 seconds	
Front panel on MNCAG:	Operator intervention	
Test MNCAG inputs:	Operator intervention	
Common mode test:	Operator intervention	

7.2 Status register and vector addresses

The program enables testing more than one MNCAD. The first MNCAD'S status register address must be in \$BASE (1244), its vector address must be in the low byte of \$VECT1 (1240). The operator may use the 'B' program command to change the default values.

7.3 Switch register

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

7.4 Bit map graphic output terminal available

The operator may inform the program that the console is a bit map terminal (I.E. VT105 or VT55) by answering 'YES' to the initial program starting question. The program will then display the results of the differential linearity and relative accuracy tests on the bit map terminal screen.

8.0 RESTRICTIONS

8.1 Testing

No external connections to the MNCAD during program execution.

8.2 Starting restriction

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

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

8.3 Possible program 'BOMBS'

The first test of this program check to see if the MNCAD responds to the expected address. If the MNCAD does not respond, a bus error occurs and a error is reported to the operator. Also bus errors can occur during the time the program sizes to see how many MNCAD'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 an 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

MNCAD TESTING

These 28 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. If the MNCAD-TA (test module) is connected, the operator is requested to change the position of the switch on the MNCAD-TA.

MNCAG TESTING

When a MNCAG has been detected, these 5 logic subtests are run sequentially after the MNCAD tests. Their purpose is to check that each of the GAIN register bits can be loaded and properly read-back. It also ensures that loading the GAIN bits of the selected channel does not effect the condition of the GAIN bits of another channel.

9.2 Calibration loop for MNCAD

If 'C' is typed, the program will ask for a channel. Type channel number followed by depressing 'RETURN'. The program will ask you if you want offset or gain. Apply voltage requested to selected channel. Adjust pot requested for 0.00 LSB 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 re-adjust the same pot again.

9.3 Print converted analog value loop

The program collects 8 samples and then reports the average value to the operator. this loop allows the operator to check the converted values of each channel. the operator may also change the gain of the MNCAG channels.

9.4 Differential linearity and relative accuracy

This test determines the width of each state to within 0.01 LSB. The basic process consists of applying a FULL SCALE ramp input and creating a histogram buffer of converted values. The values in the histogram buffer are then compared to a set of nominal limit values.

9.5 Settling test

The purpose of this test is to verify that the time allowed for settling to a new input value after switching channels does not result in an error that exceeds the expected amount for such a change.

9.6 Noise test

This test measures the short-term MINC-11 system noise. RMS noise equals 1 standard deviation of the Gaussian curve, PEAK noise equals 3 standard deviation of the Gaussian curve.

9.7 Analog tests

These 8 subtests check the converted values of the selected channels and their output.

9.8 Function test of the MNCAG front panel

This test enables the operator to verify proper operation of the MNCAG front panel controls and digital read-back logic. The program asks the operator to set the MNCAG front panel switches. The program will then read the status and gain bits and compare it to the expected value. Analog testing of the different gains is not performed in this test.

9.9 Test MNCAG channels analog input value

This test is used to verify proper operation of the analog control logic. The test requires that the operator set the switches on the MNCAG-TA test module and the front panel switches. The program will verify the converted value to an expected value for that gain and mode settings. This test checks all the gains and modes of the MNCAG front panel switches.

21	BASIC DEFINITIONS
22	OPERATIONAL SWITCH SETTINGS
29	TRAP CATCHER
56	ACT11 HOOKS
58	APT PARAMETER BLOCK
59	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
106	MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
171	INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE
179	INITIALIZE THE COMMON TAGS
191	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
221	OPERATOR INPUT DECODER
335	DETERMINE THE NUMBER OF MNCAD'S ON THE SYSTEM
389	T1 +15 VOLT TEST (TESTER ONLY)
417	T2 -15 VOLT TEST (TESTER ONLY)
435	T3 FLOAT A ONE THRU MULTIPLEXER BITS
447	T4 LOAD AND READ BACK ERROR I.E. BIT14
451	T5 LOAD AND READ BACK INTERRUPT ENABLE BIT6
457	T6 LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BITS
461	T7 LOAD AND READ BACK EXTERNAL START ENABLE BIT4
465	T10 LOAD AND READ BACK MAINT. TST BIT2
470	T11 LOAD AND READ BACK ENABLE I.D. BIT3
475	T12 LOAD AND READ BACK ERROR FLAG BIT15
479	T13 TEST INIT CLEARS BITS 2-6,8-14
487	T14 TEST INIT CLEARS ERROR FLAG
493	T15 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
501	T16 TEST INIT CLEARS DONE FLAG
511	T17 TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
520	T20 TEST ALL '0'S RESULTS USING MAINT. ADTST. BIT
530	T21 TEST ALL '1'S RESULT USING MAINT. ADTST. BIT
541	T22 GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
568	T23 TEST INTERRUPT OCCURS WHEN ERROR AND I.E.E. IS SET
593	T24 TEST ERROR FLAG SETS IF 2ND CONVERSION ENDS BEFORE READING BUFFER
606	T25 TEST ERROR FLAG SETS IF START 2ND CONV. BEFORE DONE FLAG SETS
621	T26 TEST CHANNELS 0-7 FOR SINGLE ENDED
634	T27 TEST CLOCK OVERFLOW STARTS A/D (TESTER ONLY)
647	T30 TEST CLOCK OVERFLOW STARTS A/D (IF MNCKW IS AVAILABLE)
660	T31 TEST MNCAD S.E.- DIFF MODE STATUS BIT (TESTER ONLY)
672	T32 TEST MNCAM S.E.- DIFF MODE STATUS BIT (TESTER ONLY)
684	T33 TEST MNCAD S.E.- DIFF MODE STATUS BIT (MNCAD-TA ONLY)
713	T34 TEST EXTERNAL START STARTS A/D (MNCAD-TA OR TESTER)
789	T35 VERIFY 'HOLD' FROM MNCAG CHANNEL 10
792	T36 VERIFY 'HOLD' FROM MNCAG CHANNEL 11
795	T37 VERIFY 'HOLD' FROM MNCAG CHANNEL 12
798	T40 VERIFY 'HOLD' FROM MNCAG CHANNEL 13
802	T41 MNCAG GAIN BITS LOGIC TESTS
819	T42 END OF MNCAD, MNCAG LOGIC TESTS
822	WRAPAROUND ANALOG TEST SECTION
824	T43 TEST CH0 GROUND
832	T44 TEST CH1 +4.5 VOLT
839	T45 TEST CH2 -4.5 VOLT
846	T46 TEST CH5 GROUND (MNCAD-TA OR TESTER EXCEPT IF MNCAG)
862	T47 TEST CH4 +2.6 VOLTS (MNCAD-TA OR TESTER)
870	T50 TEST CH6 -2.2 VOLTS (MNCAD-TA OR TESTER)

879	T51	TEST VOLTAGE ON SINGLE-ENDED CHANNELS (MNCAD-TA OR MNCAM-TA OR TESTER)
911	T52	TEST VOLTAGE ON DIFFERENTIAL CHANNELS (MNCAD-TA OR MNCAM-TA OR TESTER)
941	T53	TEST VERNIER OFFSET DAC ON CHO
954	T54	OFFSET ON CHO
976	T55	TEST RAMP RANGE, CH3
1004	T56	NOISE TEST, 1 EDGE (SINGLE ENDED AND MNCAG CHANNELS ONLY)
1113	T57	INTERCHANNEL SETTling TEST, 1 EDGE
1161	T60	DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST* (CHANNEL 3 ONLY AFTER FIRST PASS)
1169	T61	END OF WRAPAROUND ANALOG TESTS
1382		MNCAD CALIBRATION SECTION
1436		SWITCH GAIN MANUAL INTERVENTION TEST
1492		MNCAG TEST MODULE INTERACTIVE TESTS
1755		PRINT VALUES ROUTINE
1819		LOGIC TEST SECTION START-UP
1830		AUTO TEST START-UP
1841		WRAPAROUND TEST START-UP
1851		NOISE TEST START-UP
1874		MNCAG COMMON MODE REJECTION TEST
1915		DIFFERENTIAL LINEARITY AND REL. ACC. START-UP
1939		SETTLING TEST START-UP
2712		DETERMINE IF MORE MNCAD'S TO BE TESTED
3460		END OF PASS ROUTINE
3570		ASCII MESSAGES
3729		ASCII TEXT MESSAGES
3781		TTY INPUT ROUTINE
3783		READ AN OCTAL NUMBER FROM THE TTY
3785		SCOPE HANDLER ROUTINE
3798		ERROR HANDLER ROUTINE
3799		ERROR MESSAGE TIMEOUT ROUTINE
3800		POWER DOWN AND UP ROUTINES
3803		TYPE ROUTINE
3804		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3805		APT COMMUNICATIONS ROUTINE
3807		BINARY TO OCTAL (ASCII) AND TYPE
3808		BINARY TO ASCII AND TYPE ROUTINE
3809		TRAP DECODER
(3)		TRAP TABLE

```
20      .TITLE  CVMNA-B MNCAD/MNCAM/MNCAG DIAGNOSTIC
(1)    :*(COPYRIGHT (C) 1979
(1)    :*DIGITAL EQUIPMENT CORP.
(1)    :*MAYNARD, MASS. 01754
(1)    :*
(1)    :*PROGRAM BY RAY SHOOP
(1)    :*
(1)    :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)    :*PACKAGE (MAINDEC-11-DZQAC-(3)), 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= 1          ;;CODE FOR HORIZONTAL TAB
(1)    000012  LF= 12         ;;CODE FOR LINE FEED
(1)    000015  CR= 15         ;;CODE FOR CARRIAGE RETURN
(1)    000200  CRLF= 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)    177776  PS= 177776     ;;PROCESSOR STATUS WORD
(1)    .EQUIV  PS,PSW
(1)    177774  STKLMT= 177774 ;;STACK LIMIT REGISTER
(1)    177772  PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)    177570  DSWR= 177570   ;;HARDWARE SWITCH REGISTER
(1)    177570  DDISP= 177570  ;;HARDWARE DISPLAY REGISTER
(1)
(1)    :*GENERAL PURPOSE REGISTER DEFINITIONS
(1)    000000  R0= %0          ;;GENERAL REGISTER
(1)    000001  R1= %1          ;;GENERAL REGISTER
(1)    000002  R2= %2          ;;GENERAL REGISTER
(1)    000003  R3= %3          ;;GENERAL REGISTER
(1)    000004  R4= %4          ;;GENERAL REGISTER
(1)    000005  R5= %5          ;;GENERAL REGISTER
(1)    000006  R6= %6          ;;GENERAL REGISTER
(1)    000007  R7= %7          ;;GENERAL REGISTER
(1)    000006  SP= %6          ;;STACK POINTER
(1)    000007  PC= %7          ;;PROGRAM COUNTER
(1)
(1)    :*PRIORITY LEVEL DEFINITIONS
(1)    000000  PR0= 0          ;;PRIORITY LEVEL 0
(1)    000040  PR1= 40         ;;PRIORITY LEVEL 1
(1)    000100  PR2= 100        ;;PRIORITY LEVEL 2
(1)    000140  PR3= 140        ;;PRIORITY LEVEL 3
(1)    000200  PR4= 200        ;;PRIORITY LEVEL 4
(1)    000240  PR5= 240        ;;PRIORITY LEVEL 5
(1)    000300  PR6= 300        ;;PRIORITY LEVEL 6
(1)    000340  PR7= 340        ;;PRIORITY LEVEL 7
(1)
(1)    :*'SWITCH REGISTER' SWITCH DEFINITIONS
(1)    100000  SW15= 100000
(1)    040000  SW14= 40000
(1)    020000  SW13= 20000
(1)    010000  SW12= 10000
```

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

(1) ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

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

(1)	000004	ERRVEC= 4	::TIME OUT AND OTHER ERRORS
(1)	000010	RESVEC= 10	::RESERVED AND ILLEGAL INSTRUCTIONS
(1)	000014	TBITVEC=14	::'T' BIT
(1)	000014	TRTVEC= 14	::TRACE TRAP


```

(1)      000014      BPTVEC= 14      ;;BREAKPOINT TRAP (BPT)
(1)      000020      IOTVEC= 20      ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)      000024      PWRVEC= 24      ;;POWER FAIL
(1)      000030      EMTVEC= 30      ;;EMULATOR TRAP (EMT) **ERROR**
(1)      000034      TRAPVEC=34      ;;'TRAP' TRAP
(1)      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
(1)      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
(1)      000240      PIRQVEC=240    ;;PROGRAM INTERRUPT REQUEST VECTOR
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      INHIBIT SIZING # OF MNCAD'S
(1)      *      11      INHIBIT ITERATIONS
(1)      *      10      HALT FOR VIEWING BIT MAP TERMINAL DISPLAY
(1)      *      9      LOOP ON ERROR
(1)      *      8      LOOP ON TEST IN SWR<7:0>
23      171000      ABASE- 171000
24      000400      AVECT1- 400
25
26      000100      000100      000200      000002      .-100
27      000100      000104      000200      000002      .WORD 104,200,2
28
29      .SBTTL TRAP CATCHER
30
31      000000      . 0
32      *ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A '+2'
33      *AND 'JSR PC,RO' SEQUENCE TO CATCH ILLEGAL INTERRUPTS.
34      *AND INTERRUPTS TO THE WRONG VECTOR.
35      *LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
36      *VECTORS.
46      000004      000004      000200      .-4
47      000004      027276      000200      .WORD IOTRD,200      ;HANDLE BUSS ERROR.
48      000174      000174      .-174
49      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER.
50      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER.
51
52      000200      000137      001626      JMP BEGIN      ;START ADDRESS
53      000204      000137      001634      JMP @MBEG?     ;RESTART ADDRESS
54      000210      000137      001642      JMP @MBEGIN2   ;START ADDRESS FOR OPTION TESTER
  
```

```
56 .SBTTL ACT11 HOOKS
(1)
(2)
(1)
(1) 000214 $SVPC- ;SAVE PC
(1) 000046 .-46
(1) 000046 027164 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1) 000052 .=52
(1) 000052 000000 .WORD 0 ;;2)SET LOC.52 TO ZERO
(1) 000214 .=$SVPC ;; RESTORE PC
57 -1000
58 .SBTTL APT PARAMETER BLOCK
(1)
(2)
(1)
(2)
(1) 001000 .SX= ;;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)
(1)
(1)
(1) 001000 $APTHD:
(1) 001000 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001170 $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 002260 $STMT: .WORD 1200. ;;RUN TIM OF LONGEST TEST
(1) 001006 000764 $PASTM: .WORD 500. ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 003244 $UNITM: .WORD 1700. ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000031 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```



```
(2) 001210          $ETABLE:          ;; APT ENVIRONMENT TABLE
(2) 001210          $ENV: .BYTE   AENV          ;; ENVIRONMENT BYTE
(2) 001211          $ENVM: .BYTE   AENVM        ;; ENVIRONMENT MODE BITS
(2) 001212 000000   $$WREG: .WORD   ASWREG       ;; APT SWITCH REGISTER
(2) 001214 000000   $USWR: .WORD   AUSWR       ;; USER SWITCHES
(2) 001216 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) 001220          $MAMS1: .BYTE   AMAMS1     ;; HIGH ADDRESS, M.S. BYTE
(2) 001221          $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) 001222 000000   $MADR1: .WORD   AMADR1     ;; HIGH ADDRESS, BLK#1
(2)                  : *                MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
(2) 001224          $MAMS2: .BYTE   AMAMS2     ;; HIGH ADDRESS, M.S. BYTE
(2) 001225          $MTYP2: .BYTE   AMTYP2     ;; MEM. TYPE, BLK#2
(2) 001226 000000   $MADR2: .WORD   AMADR2     ;; MEM. LAST ADDRESS, BLK#2
(2) 001230          $MAMS3: .BYTE   AMAMS3     ;; HIGH ADDRESS, M.S. BYTE
(2) 001231          $MTYP3: .BYTE   AMTYP3     ;; MEM. TYPE, BLK#3
(2) 001232 000000   $MADR3: .WORD   AMADR3     ;; MEM. LAST ADDRESS, BLK#3
(2) 001234          $MAMS4: .BYTE   AMAMS4     ;; HIGH ADDRESS, M.S. BYTE
(2) 001235          $MTYP4: .BYTE   AMTYP4     ;; MEM. TYPE, BLK#4
(2) 001236 000000   $MADR4: .WORD   AMADR4     ;; MEM. LAST ADDRESS, BLK#4
(2) 001240 000400   $VECT1: .WORD   AVECT1     ;; INTERRUPT VECTOR#1, BUS PRIORITY#1
(2) 001242 000000   $VECT2: .WORD   AVECT2     ;; INTERRUPT VECTOR#2, BUS PRIORITY#2
(2) 001244 171000   $BASE: .WORD   ABASE      ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001246 000000   $DEVN: .WORD   ADEVN      ;; DEVICE MAP
(2) 001250 000000   $CDW1: .WORD   ACDW1      ;; CONTROLLER DESCRIPTION WORD#1
(2) 001252          $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 $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPL).
(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) 001252 $ERRTB:
61
70 ;ITEM 1
71 001252 037135 037727 040270 EM1,DH1,DT1,DF1 ;MNCAD STATUS REG. ERROR
001260 040412
72 ;ITEM 2
73 001262 037173 040057 040324 EM2,DH3,DT3,DF1 ;MNCAD FAILED TO INTERRUPT
001270 040412
74 ;ITEM 3
75 001272 037233 040057 040324 EM3,DH3,DT3,DF1 ;MNCAD UNEXPECTED INTERRUPT
001300 040412
76 ;ITEM 4
77 001302 037274 037773 040304 EM4,DH2,DT2,DF1 ;MNCAD ERROR ON A/D CHANNEL
001310 040412
78 ;ITEM 5
79 001312 037335 040113 040336 EM5,DH5,DT5,DF1 ;EXISTING MNCAD NOW FAILS TO RESPOND
001320 040412
80 ;ITEM 6
81 001322 037416 040137 040350 EM6,DH6,DT6,DF1 ;BUS ERROR ON SPECIFIED DEFAULT ADDRESS
001330 040412
82 ;ITEM 7
83 001332 037514 040160 040360 EM7,DH7,DT7,DF1 ;INCORRECT I.D. VALUE
001340 040412
84 ;ITEM 10
85 001342 037541 037727 040270 EM10,DH1,DT1,DF1 ;'MNCAG HOLD' SIGNAL IN ERROR
001350 040412
86 ;ITEM 11
87 001352 037605 040216 040374 EM11,DH12,DT12,DF1 ;'INCORRECT' MNCAG (PREAMP) FRONT PANEL SWITCH POSITION
001360 040412
88 ;ITEM 12
89 001362 037661 040216 040374 EM12,DH12,DT12,DF1 ;MNCAG GAIN REGISTER IN ERROR
001370 040412
90 001372 000000 ADTA: 0 ;MNCAD-TA INDICATOR
91 001374 000000 AMTA: 0 ;MNCAM-TA INDICATOR
92 001376 000000 AGTA: 0 ;MNCAG-TA INDICATOR
93 001400 001000 BARFO: BIT9 ;DELAY FACTOR FOR CPU, SO THE HELP MESSAGE WONT GET MESSED UP
94 ;AND OTHER TESTS
```

96				
97	001402	171000	MNCADO: ABASE	:ADDRESS OF MNCAD #0
98	001404	000400	AVECT1	:VECTOR OF MNCAD #0
99	001406	171004	ABASE+4	: #1
100	001410	000410	AVECT1+10	: #1
101	001412	171010	ABASE+10	: #2
102	001414	000460	AVECT1+60	: #2
103	001416	171014	ABASE+14	: #3
104	001420	000470	AVECT1+70	: #3
105				
106			.SBTTL MISCELLANEGUS, TEMPORARY, AND STORAGE LOCATIONS	
107	001422	171000	STREG: ABASE	:ADDRESS OF STATUS REGISTER
108	001424	171001	ADST: ABASE+1	:UPPER BYTE OF STATUS REG.
109	001426	171002	ADBUFF: ABASE+2	:ADDRESS OF A/D BUFFER
110	001430	000400	VECTOR: AVECT1	:VECTOR ADDRESS
111	001432	000402	VECTR1: AVECT1+2	
112	001434	000404	VECTR2: AVECT1+4	:ERROR VECTOR ADDRESS
113	001436	000406	VECTR3: AVECT1+6	
114	001440	000000	BASECH: 0	:BASE CHANNEL
115	001442	000000	BASEND: 0	:END CHANNEL
116	001444	000060	KBVECT: 60	
117	001446	171020	KWCSR: 171020	:NORMAL MNCKW ADDRESS
118	001450	171022	KWBPR: 171022	:MNCKW BUF REG.
119			: TESTER DEVICES	
120	001452	170400	GSTREG: 170400	:KNOWN GOOD A/D CSR
121	001454	170402	GADBUF: 170402	:KNOWN GOOD A/D DBR
122	001456	000410	GVECT: 410	:KNOWN GOOD A/D VECTOR
123	001460	170430	CLKCSR: 170430	:CLOCK CSR
124	001462	170432	CLKBPR: 170432	:CLOCK BPR
125	001464	167770	DRVCSR: 167770	:DRV11 CSR
126	001466	167772	DRVDOR: 167772	:DRV11 DOR
127	001470	167774	DRVDIR: 167774	:DRV11 DIR
128			: COMMON TAGS	
129	001472	000000	WIDE: 0	:NO. OF WIDE STATES
130	001474	000000	NARROW: 0	:NO. OF NARROW STATES
131	001476	000000	FIRST: 0	
132	001500	000000	SKIPST: 0	:NO. OF SKIPPED STATES
133	001502	000000	TEMP: 0	:WORK AREA
134	001504	000000	TEMP1: 0	:RESTART INDICATOR
135	001506	000000	CH1: 0	:FIRST CHANNEL
136	001510	000000	CH2: 0	:SECOND CHANNEL
137	001512	000000	NBEXT: 0	:NO. OF MNCAD'S TO BE TESTED
138	001514	000000	NMBEXT: 0	:NO. OF MNCAD'S TO BE TESTED
139	001516	000000	DUMMY: 0	:DUMMY CHANNEL
140	001520	000000	CHANL: 0	:CHANNEL VALUE
141	001522	000000	RMS: 0	:RMS NOISE VALUE
142	001524	000000	PEAK: 0	:PEAK NOISE VALUE
143	001526	000000	VTFLAG: 0	:BIT MAP TERMINAL FLAG
144	001530	000000	SPREAD: 0	:DEVIATION FROM THE NOMINAL
145	001532	000000	DAC: 0	:SAR VALUE
146	001534	000000	DELAY: 0	:TIME DELAY COUNTER
147	001536	000000	EDGE: 0	:EDGE VALUE
148	001540	000000	BITPNT: 0	
149	001542	000000	MIN: 0	:MIN VALUE
150	001544	000000	WFTST: 0	
151	001546	000000	KWAD: 0	:MNCKW AVAILABLE TO TEST CLOCK STARTS

```

152 001550 000000 MAX: 0 ;MAX VALUE
153 001552 000000 PERCNT: 0 ;PERCENT FOR SAR ROUTINE
154 001554 000000 OUT: 0
155 001556 000000 EVER: 0
156 001560 000000 BADUNT: 0 ;BAD UNIT MAP
157 001562 000001 MASKNM: 1 ;CURRENT UNIT MAP
158 001564 000000 UNITBD: 0
159
160 001566 UNEXP:
(1) 001566 012737 001602 001162 MOV #1$, $ESCAPE ;;ESCAPE TO 1$ ON ERROR
161 001574 005237 001103 INC $ERFLG
162 001600 104003 ERROR 3
163 001602 005037 001162 1$: CLR $ESCAPE ;RETURN ESCAPE TO NORMAL
164 001606 000002 RTI ;UNEXPECTED INTERRUPT
165 001610 022776 000001 000000 RETURN: CMP #1, @0(SP) ;DOES IT RETURN TO A WAIT?
166 001616 001002 BNE RET2 ;NO
167 001620 062716 000002 RET1: ADD #2, (SP) ;BUMP RETURN ADDRESS
168 001624 000002 RET2: RTI
169
170
171 .SBTTL INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE
172 001626 005037 001544 BEGIN: CLR WFTEST
173 001632 000406 BR RBEG
174 001634 005237 001504 BEG2: INC TEMP1 ;SET RESTART FLAG
175 001640 000405 BR RBEG1
176 001642 012737 100000 001544 BEGIN2: MOV #BIT15, WFTEST ;INDICATE TESTER IS CONNECTED
177 001650 005037 001504 RBEG: CLR TEMP1 ;CLEAR RESTART FLAG
178 001654 004737 026402 RBEG1: JSR PC, ARESET ;GENERATE A CONTROLLED BUS RESET
179
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) MOV # $CMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
(1) CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) CMF #SWR, R6 ;;DONE?
(1) BNE -6 ;;LOOP BACK IF NO
(1) MOV #STACK, SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) MOV # $SCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) MOV #340, @IOTVEC+2 ;;LEVEL 7
(1) MOV # $ERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) MOV #340, @EMTVEC+2 ;;LEVEL 7
(1) MOV # $TRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) MOV #340, @TRAPVEC+2 ;;LEVEL 7
(1) MOV # $PWRDN, @PWRVEC ;;POWER FAILURE VECTOR
(1) MOV #340, @PWRVEC+2 ;;LEVEL 7
(1) MOV $ENDCT, $EOPCT ;;SETUP END-OF-PROGRAM COUNTER
(1) CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) MOVB #1, $ERMAX ;;ALLOW ONE ERROR PER TEST
(1) MOV #., $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) MOV #., $LPERR ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
(2) MOV #64$, @ERRVEC ;;SET UP ERROR VECTOR
(2) MOV #DSWR, SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER

```

```
(2) 002046 022777 177777 177064      CMP    #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
(2) 002054 001012                      BNE    66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                                ;;AND THE HARDWARE SWR IS NOT
(2) 002056 000403                      BR     65$          ;;BRANCH IF NO TIMEOUT
(2) 002060 012716 002066 64$:        MOV    #65$,(SP)    ;;SET UP FOR TRAP RETURN
(2) 002064 000002                      RTI
(2) 002066 012737 000176 001140 65$:  MOV    #SWREG,SWR    ;;POINT TO SOFTWARE SWR
(2) 002074 012737 000174 001142      MOV    #DISPREG,DISPLAY
(2) 002102 012637 000004 66$:        MOV    (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002106 005037 001176                      CLR    $PASS        ;;CLEAR PASS COUNT
(2) 002112 132737 000200 001211      BITB   #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 002120 001403                      BEQ    67$          ;;YES,USE NON-APT SWITCH
(2) 002122 012737 001212 001140      MOV    #SSWREG,SWR  ;;NO,USE APT SWITCH REGISTER
(2) 002130
180
181 002130 012737 005046 043160      MOV    #5046,$TYPE  ;;CLR -(SP)
182 002136 012737 012746 043162      MOV    #12746,$TYPE+2 ;;MOV #TYPE+12,-(SP)
183 002144 012737 043172 043164      MOV    #TYPE+12,$TYPE+4
184 002152 012737 000002 043166      MOV    #RTI,$TYPE+6 ;;RTI
185 002160 004737 040470      JSR    PC,$TKINT    ;;ENABLE TKB INTR.
186 002164 005737 001504      TST    TEMP1        ;;TEST IF RESTART
187 002170 001005                      BNE    20$          ;;BR IF YES
188 002172 005737 000042      TST    @#42         ;;TEST IF CHAIN MODE
189 002176 001002                      BNE    20$          ;;BR IF CHAIN MODE
190 002200 104401 036027      TYPE   ,INITVT     ;;INITILIZE THE TERMINAL
191 002204
(1)
(1) .SBTTL TYPE PROGRAM NAME
(1) ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002204 005227 177777      INC    #-1          ;;FIRST TIME?
(1) 002210 001050                      BNE    68$          ;;BRANCH IF NO
(1) 002212 022737 027164 000042      CMP    #SENDAD,@#42 ;;ACT-11?
(1) 002220 001444                      BEQ    68$          ;;BRANCH IF YES
(1) 002222 104401 002270      TYPE   ,69$        ;;TYPE ASCIZ STRING
(2)
(2) .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002226 005737 000042      TST    @#42         ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002232 001012                      BNE    70$          ;;BRANCH IF YES
(2) 002234 123727 001210 000001      CMPB   $ENV,#1     ;;ARE WE RUNNING UNDER APT?
(2) 002242 001406                      BEQ    70$          ;;BRANCH IF YES
(2) 002244 023727 001140 000176      CMP    SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
(2) 002252 001005                      BNE    71$          ;;BRANCH IF NO
(2) 002254 104407      GTSWR
(2) 002256 000403      BR     71$          ;;GET SOFT-SWR SETTINGS
(2) 002260 112737 000001 001134 70$:  MOVB   #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
(2) 002266
(1) 002266 000421      BR     68$          ;;GET OVER THE ASCIZ
(1) ;;69$: .ASCIZ <CRLF>#CVMA-B MNCAD (A/D) DIAGNOSTIC#<CRLF>
(1) 002332
192 002332 013746 000010 68$:  MOV    @RESVEC,-(SP) ;;SAVE RESERVED VECTOR
193 002336 012737 002376 000010      MOV    #1$,RESVEC  ;;SET UP ILLEGAL INST. TRAP
194 002344 012700 000001      MOV    #1,R0        ;;SET R0 TO ONE
195 002350 077001      SOB    R0           ;;TRY SOB INSTRUCTION
196 002352 012737 077001 024152      MOV    #77001,DELAY1 ;;SET UP FOR SOB
197 002360 012737 077001 024266      MOV    #77001,DELAY2
198 002366 012737 077001 024402      MOV    #77001,DELAY3
199 002374 000412      BR     2$
```


200	002376	022626			1\$:	CMP	(SP)+,(SP)+	;POP TWO WORDS OFF STACK
201	002400	012737	104420	024152		MOV	#DELAY,DELAY1	;INSTRUCTION FAILED
202	002406	012737	104420	024266		MOV	#DELAY,DELAY2	:
203	002414	012737	104420	024402		MOV	#DELAY,DELAY3	:
204	002422	012637	000010		2\$:	MCV	(SP)+, @WRESVEC	;RESTORE ERROR VECTOR
205	002426	004737	023024		3\$:	JSR	PC, FIXONE	;INITIALIZE ADDRESSES
206	002432	004737	026632			JSR	PC, WFADJ	;SET UP TOLLERANCES
207	002436	105737	001134			TSTB	\$AUTOB	;TEST IF CHAIN/APT
208	002442	001402				BEQ	4\$	
209	002444	000137	015226			JMP	BEG1	;GO TO LOGIC TESTS
210	002450	005737	001504		4\$:	TST	TEMP1	;TEST IF RESTART
211	002454	001125				BNE	MTEST1	
212	002456	005737	001544			TSI	WFTST	;CHECK IF TESTER CONNECTED ?
213	002462	001414				BEQ	MTEST	;BR IF NO TESTER
214	002464	104401	032160			TYPE	,SDDIF	;SET MNCAD-TA TO DIFF
215	002470	104401	032276			TYPE	,SDMDIF	;SET MNCAM-TA TO DIFF
216	002474	005237	001372			INC	ADTA	;SET AD-TA AVAIL FLAG
217	002500	005237	001374			INC	AMTA	;SET AM-TA AVAIL FLAG
218	002504	005237	001376			INC	AGTA	;SET AG-TA AVAIL FLAG
219	002510	000137	002724			JMP	MTESTO	;BYPASS NORMAL START-UP Q + A
220								
221						.SBTTL	OPERATOR INPUT DECODER	
222	002514	104401	001165		MTEST:	TYPE	, \$CRLF	
223	002520	104401	031734			TYPE	, SADTST	;TELL OPER. ABOUT MNCAD FRONT PANEL SW.
224	002524	104401	032015			TYPE	, SAGTST	;TELL OPER. ABOUT MNCAG FRONT PANEL SW.
225	002530	104401	031615			TYPE	, YESNO	;ASK FOR INPUT
226	002534	004537	002644			JSR	R5, ASKTA	;ASK ABOUT MNCAD-TA
227	002540	027753				DWRFAD		
228	002542	001372				ADTA		
229	002544	000402				BR	1\$;BR IF NONE
230	002546	104401	032110			TYPE	, SDSE	;TELL OPER. TO SET MNCAD-TA SWITCH TO SINGLE END
231	002552	004537	002644		1\$:	JSR	R5, ASKTA	;ASK ABOUT MNCAM-TA
232	002556	030027				DWRFAM		
233	002560	001374				AMTA		
234	002562	000402				BR	2\$;BR IF NONE
235	002564	104401	032230			TYPE	, SDMSE	;TELL OPER. TO SET MNCAM-TA SWITCH TO SINGLE END
236	002570	004537	002644		2\$:	JSR	R5, ASKTA	;ASK ABOUT MNCAG-TA
237	002574	030101				DWRFAG		
238	002576	001376				AGTA		
239	002600	000406				BR	4\$;BR IF NONE
240	002602	104401	032761			TYPE	, TXTP2	;TELL OPER. TO SET MNCAG-TA SWITCHES
241	002606	104401	032575			TYPE	, SVM	;AND MODE SWITCHES TO VOLTAGE
242	002612	104401	001165			TYPE	, \$CRLF	
243	002616	004537	002644		4\$:	JSR	R5, ASKTA	;ASK IF MNCKW IS IN SYSTEM
244	002622	031051				SCLOCK		
245	002624	001546				KWAD		
246	002626	000240				NOP		;MUST LEAVE NOP HERE
247	002630	004537	002644			JSR	R5, ASKTA	;ASK IF VT55/VT105 TERMINAL IS CONNECTED
248	002634	030156				DWRMAP		
249	002636	001526				VTFLAG		
250	002640	000240				NOP		;MUST LEAVE NOP HERE
251	002642	000430				BR	MTESTO	
252	002644	012537	002656		ASKTA:	MOV	(R5)+, 10\$;GET MESSAGE POINTER
253	002650	104401	001165			TYPE	, \$CRLF	;FRESH LINE
254	002654	104401				TYPE		;ABOUT DWARF MODULE
255	002656	027753			10\$:	DWRFAD		

256	002660	104412			RDLIN		
257	002662	012600			MOV	(SP)+,R0	;GET INPUT
258	002664	005075	000000		CLR	@(R5)	;SET NO MNCXX-TA FLAG
259	002670	042710	000040		BIC	#40,(R0)	;ENSURE UPPER CASE
260	002674	122710	000131		CMPB	#'Y',(R0)	;TEST IF 1ST CHAR IS Y
261	002700	001004			BNE	1\$;BR IF NOT 'Y'
262	002702	005235			INC	@(R5)+	;SET MNCXX-TA CONNECTED FLAG
263	002704	000240			NOP		
264	002706	000240			NOP		
265	002710	000240			NOP		
266	002712	005725		1\$:	TST	(R5)+	;BUMP EXIT
267	002714	000240			NOP		
268	002716	000240			NOP		
269	002720	000240			NOP		
270	002722	000205			RTS	R5	;EXIT
271							
272	002724	104401	036061		MTEST0:	TYPE	,PRIME1
273	002730	004737	026402		MTEST1:	JSR	PC,ARESET
274	002734	052777	000100	176202		BIS	#BIT6,@\$TKS
275	002742	005046			CLR	-(SP)	
276	002744	012746	002752		MOV	#1\$,-(SP)	
277	002750	000002			RTI		;LOWER PS
278	002752	005037	001176	1\$:	CLR	\$PASS	;INIT
279	002756	005037	001112		CLR	\$ERTTL	; THINGS
280	002762	005037	001556		CLR	EVER	
281	002766	104401	037045		TYPE	,DOT	;TYPE THE 'DOT'
282	002772	104412			RDLIN		
283	002774	012600			MOV	(SP)+,R0	;READ ANSWER
284	002776	142710	000040		BICB	#40,(R0)	
285	003002	121027	000101		CMPB	(R0),#'A	;IS IT A?
286	003006	001002			BNE	2\$;:NO, TRY C
287	003010	000137	015274		JMP	BEGINA	;GO TO AUTO TEST
288	003014	121027	000103	2\$:	CMPB	(R0),#'C	;IS IT C?
289	003020	001002			BNE	3\$;:NO, TRY P
290	003022	000137	012600		JMP	BEGINC	;GO TO CALIBRATION LOOP
291	003026	121027	000120	3\$:	CMPB	(R0),#'P	;IS IT P?
292	003032	001002			BNE	4\$;:NO, TRY L
293	003034	000137	014706		JMP	BEGINP	;GO TO TYPE/DISPLAY CONVERSIONS TEST
294	003040	121027	000114	4\$:	CMPB	(R0),#'L	;IS IT L?
295	003044	001002			BNE	5\$;:NO, TRY M
296	003046	000137	015226		JMP	BEGINM	;GO TO LOGIC TESTS
297	003052	121027	000127	5\$:	CMPB	(R0),#'W	;IS IT W?
298	003056	001002			BNE	6\$;:NO, TRY AGAIN
299	003060	000137	015340		JMP	BEGINW	;GO TO WRAPAROUND TEST
300	003064	121027	000102	6\$:	CMPB	(R0),#'B	;IS IT B?
301	003070	001002			BNE	7\$;:NO TRY AGAIN
302	003072	000137	022622		JMP	BASEXC	;GO CHANGE BASE AND VECTOR ADDRESS
303	003076	121027	000110	7\$:	CMPB	(R0),#'H	;IF IT H?
304	003102	001002			BNE	10\$;:NO, TRY AGAIN
305	003104	000137	002514		JMP	MTEST	;YES, HELP THE OPERATOR
306	003110	121027	000107	10\$:	CMPB	(R0),#'G	;IS IT G?
307	003114	001002			BNE	11\$;:NO, TRY AGAIN
308	003116	104407			GTSWR		
309	003120	000703			BR	MTEST1	
310	003122	121027	000126	11\$:	CMPB	(R0),#'V	;IS IT V?
311	003126	001004			BNE	12\$;:NO, TRY AGAIN

312	003130	005237	001526		INC	VTFLAG	:SET BIT MAP AVAILABLE FLAG + RUN WRAPAROUND
313	003134	000137	015340		JMP	BEG'NW	:AND RUN WRAP TEST'S
314	003140	121027	000116	12\$:	CMPB	(R0),#N	:IS IT N?
315	003144	001002			BNE	13\$:NO, TRY AGAIN
316	003146	000137	015400		JMP	BEGINN	:RUN NOISE TESTS
317	003152	121027	000106	13\$:	CMPB	(R0),#F	:IS IT F?
318	003156	001002			BNE	14\$:NO, TRY AGAIN
319	003160	000137	013054		JMP	BEGINF	:RUN SWITCH GAIN/PREAMP FRONT PANEL TEST
320	003164	121027	000124	14\$:	CMPB	(R0),#T	:IT IT T?
321	003170	001002			BNE	15\$:NO, TRY AGAIN
322	003172	000137	013346		JMP	BEGINI	:RUN TEST MODULE VERIFY TESTS
323	003176	121027	000104	15\$:	CMPB	(R0),#D	:IS IT D?
324	003202	001002			BNE	16\$:NO, TRY AGAIN
325	003204	000137	016064		JMP	BEGINI	:RUN DIFFERENTIAL AND RELAC. TEST ONLY
326	003210	121027	000115	16\$:	CMPB	(R0),#M	:IS IT M?
327	003214	001002			BNE	17\$:NO, TRY AGAIN
328	003216	000137	015524		JMP	BEGINM	:RUN COMMON MODE TESTS
329	003222	121027	000123	17\$:	CMPB	(R0),#S	:IS IT S?
330	003226	001002			BNE	77\$:NO, TRY AGAIN
331	003230	000137	016224		JMP	BEGINS	:RUN SETTLING TEST ONLY
332	003234	104401	030233	77\$:	TYPE	,QUEST	
333	003240	000633			BR	MTEST1	:WAIT FOR CHARACTER

```

335 .SBTTL DETERMINE THE NUMBER OF MNCAD'S ON THE SYSTEM
336 003242 013737 001244 001126 TESTAD: MOV $BASE,$BDDAT ;GET BASE ADDRESS
337 003250 005037 001202 CLR $UNIT ;CLR UNIT NUMBER
338 003254 012737 003330 000004 MOV #2,$ERRVEC ;LOAD RETURN ADDRESS
339 003262 005777 175640 1$: TST @BDDAT ;TEST IF ADDRESS EXISTS
340 003266 062737 000004 001126 ADD #4,$BDDAT ;UPDATE BUS ADDRESS
341 003274 005237 001202 INC $UNIT ;UPDATE UNIT COUNT
342 003300 005737 001210 TST $ENV ;TEST IF 'DO NOT SIZE'
343 003304 100424 BMI $S ;BR IF NO SIZING
344 003306 032777 010000 175624 BIT #SW12,@SWR ;TEST IF INHIBIT SIZING IS SET
345 003314 001020 BNF $S ;BR IF SET
346 003316 022737 000004 001202 CMP #4,$UNIT ;TEST IF MAX NUMBER
347 003324 001356 BNE 1$ ;BR IF NOT
348 003326 000413 BR $S ;BR IF MAX
349 003330 022626 2$: CMP (SP)+,(SP)+ ;RESTORE STACK
350 003332 005737 001202 TST $UNIT ;TEST IF ANY EXIST
351 003336 001007 BNE $S ;BR IF ANY ARE THERE
352 003340 005737 000042 TST @#42 ;TEST IF XXDP CHAIN MODE
353 003344 001004 BNE $S ;BR IF YES
354 003346 104006 ERROR 6 ;BASE ADDRESS CAUSED A BUS TRAP
355 003350 005726 TST (SP)+ ;POP 1 ARG.
356 003352 000137 027076 JMP $EOP
357 003356 012737 027276 000004 3$: MOV #IOTRD,ERRVEC
358 003364 012737 000200 000006 MOV #200,ERRVEC+2
359 003372 005737 001556 TST EVER ;TEST IF # HAS BEEN REPORTED
360 003376 100427 BMI 4$ ;IF YES BRANCH
361 003400 005737 001544 TST WFTST ;TEST IF IN TESTER MODE
362 003404 100415 BMI 7$ ;BR IF TESTER
363 003406 104401 035105 TYPE ,FOUND1 ;TELL OPERATOR # OF MNCAD'S FOUND
364 003412 013746 001202 MOV $UNIT,-(SP) ;PUT # TO BE TYPED ON STACK
365 003416 104405 TYPDS
366 003420 104401 035130 TYPE ,FOUND2 ;FINISH MESSAGE
367 003424 005737 001202 TST $UNIT ;TEST IF ANY UNITS
368 003430 001003 BNE 7$ ;ANY UNIT
369 003432 005726 TST (SP)+ ;POP 1 ARG. OFF STACK
370 003434 000137 027076 JMP $EOP ;REPORT EOP
371 003440 013737 001202 001556 7$: MOV $UNIT,EVER ;SAVE THE # OF MNCAD'S FOR LATER
372 003446 052737 100000 001556 BIS #BIT15,EVER ;SET 'REPORTED # FLAG'
373 003454 000410 BR $S
374 003456 123737 001556 001202 4$: CMPB EVER,$UNIT ;TEST IF ANY HAVE GONE AWAY
375 003464 001404 BEQ $S ;BR IF ALL ARE STILL THERE
376 003466 113737 001556 001502 MOVB EVER,TEMP ;SAVE FOR ERROR REPORT
377 003474 104005 ERROR 5 ;EXISTING DEVICE FAILED TO RESPOND
378 003476 005037 001202 5$: CLR $UNIT ;RESET UNIT POINTER
379 003502 113737 001556 001514 MOVB EVER,NMBEXT ;GET # OF UNITS
380 003510 005337 001514 DEC NMBEXT ;ADJUST IT
381 003514 004737 023024 JSR PC,FXONE ;FIX BUS AND VECTOR ADDRESSES
382 003520 005037 001560 CLR BADUNT ;RESET BAD UNIT INDICATOR
383 003524 005046 CLR -(SP) ;LOWER PRIORITY LEVEL 0
384 003526 012746 003534 MOV #6$,-(SP)
385 003532 000002 RTI
386 003534 000207 6$: RTS PC ;EXIT
  
```


470
(3)
(3)
(2) 004172 000004
471 004174 012737 000010 001124
472 004202 104415
473 004204 104001
474
475
(3)
(3)
(2) 004206 000004
476 004210 012737 100000 001124
477 004216 104415
478 004220 104001
479
(3)
(3)
(2) 004222 000004
(1) 004224 012737 000300 001160
480 004232 005037 001124
481 004236 012777 077574 175156
482 004244 000005
483 004246 052777 000100 174670
484 004254 104414
485 004256 104001

```
*****
:*TEST 11      LOAD AND READ BACK ENABLE I.D. BIT3
*****
TST11: SCOPE
        MOV     #BIT3,$GDDAT      ;LOAD EXPECTED DATA
        CHKIT
        ERROR  1                  ;FAILED TO LOAD + READ ENABLE I.D. BIT

*****
:*TEST 12      LOAD AND READ BACK ERROR FLAG BIT15
*****
TST12: SCOPE
        MOV     #BIT15,$GDDAT    ;LOAD EXPECTED DATA
        CHKIT
        ERROR  1                  ;FAILED TO LOAD + READ ERROR FLAG

*****
:*TEST 13      TEST INIT CLEARS BITS 2-6,8-14
*****
TST13: SCOPE
        MOV     #300,$TIMES      ;;DO 300 ITERATIONS
        CLR     $GDDAT           ;LOAD EXPECTED DATA
2$:     MOV     #77574,@STREG    ;SET STATUS REGISTER
        RESET
        BIS     #100,@$TKS      ;SET INTRPT. ENABLE
        CHECK
        ERROR  1                  ;RESET FAILED TO CLEAR AD ST. REG. BITS
```

487
(3)
(3)
(2) 004260 000004
(1) 004262 012737 000300 001160
488 004270 012777 100000 175124
489 004276 000005
490 004300 052777 000100 174636
491 004306 104414
492 004310 104001
493
(3)
(3)
(2) 004312 000004
(1) 004314 012737 000100 001160
494 004322 005277 175074
495 004326 012737 000200 001124
496 004334 004737 016552
497 004340 042777 100000 175054
498 004346 104414
499 004350 104001
500 004352 017700 175050
501
(3)
(3)
(2) 004356 000004
(1) 004360 012737 000300 001160
502 004366 005037 001124
503 004372 005277 175024
504 004376 105777 175020
505 004402 100375
506 004404 000005
507 004406 052777 000100 174530
508 004414 104414
509 004416 104001
510
511
(3)
(3)
(2) 004420 000004
512 004422 005037 001124
513 004426 005277 174770
514 004432 105777 174764
515 004436 100375
516 004440 017700 174762
517 004444 104414
518 004446 104001

```
*****  
*TEST 14 TEST INIT CLEARS ERROR FLAG  
*****  
TST14: SCOPE  
MOV #300,$TIMES ;;DO 300 ITERATIONS  
MOV #BIT15,@STREG ;SET BIT 15  
RESET ;ISSUE INIT  
BIS #100,@$TKS ;SET INTRPT. EN. FOR KEYBOARD  
CHECK  
ERROR 1  
*****  
*TEST 15 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.  
*****  
TST15: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
INC @STREG ;START CONVERSION  
MOV #BIT7,$GDDAT ;LOAD EXPECTED  
JSR PC,STALL ;DELAY  
BIC #BIT15,@STREG ;MASK OUT ERROR BIT  
CHECK  
ERROR 1 ;A/D DONE FLAG FAILED TO SET;BIT0 FAILED TO CLEAR  
MOV @ADBUFF,R0 ;CLEAR DONE FLAG FOR ITERATIONS  
*****  
*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  
BIS #BIT6,@$TKS ;ENABLE INTR.  
CHECK  
ERROR 1 ;DONE FLAG FAILED TO CLEAR  
*****  
*TEST 17 TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE  
*****  
TST17: SCOPE  
CLR $GDDAT ;CLEAR EXPECTED  
INC @STREG ;SET A/D START CONVERSION BIT  
1$: TSTB @STREG ;WAIT FOR FLAG  
BPL 1$  
MOV @ADBUFF,R0 ;READ CONVERTED VALUE  
CHECK  
ERROR 1 ;DONE FLAG FAILED TO CLEAR
```



```
520 (3) *****  
(3) *TEST 20 TEST ALL '0'S RESULTS USING MAINT. ADTST. BIT  
(2) 004450 000004  
521 004452 005037 001124  
522 004456 005037 001520  
523 004462 005037 001530  
524 004466 012777 000005 174726  
525 004474 105777 174722 1$: TSTB @STREG ;WAIT FOR DONE  
526 004500 100375 BPL 1$  
527 004502 017737 174720 001126 MOV @ADBUFF,$BDDAT ;RESULTS TO BDDAT FOR CHECKING  
528 004510 001401 BEQ TST21 ;GOTO NEXT TEST  
529 004512 104004 ERROR 4 ;DID NOT GET ALL '0'S RESULT WITH MAINT. ADTST  
530 (3) *****  
(3) *TEST 21 TEST ALL '1'S RESULT USING MAINT. ADTST. BIT  
(2) 004514 000004  
531 004516 012737 007777 001124  
532 004524 012737 000001 001520  
533 004532 005037 001530  
534 004536 012777 000405 174656  
535 004544 105777 174652 1$: TSTB @STREG ;WAIT FOR DONE  
536 004550 100375 BPL 1$  
537 004552 017737 174650 001126 MOV @ADBUFF,$BDDAT ;RESULTS TO BDDAT FOR CHECKING  
538 004560 023737 001124 001126 CMP $GDDAT,$BDDAT ;EQUAL?  
539 004566 001401 BEQ TST22 ;GOTO NEXT TEST  
540 004570 104004 ERROR 4 ;DID NOT GET ALL '1'S RESULT WITH MAINT. ADTST  
541 (3) *****  
(3) *TEST 22 GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION  
(2) 004572 000004  
(1) 004574 012737 000100 001160  
542 004602 012737 004610 001106  
543 004610 042777 000100 174326 10$: BIC #BIT6,@STKS ;REMOVE TKB INTERRUPT  
544 004616 005046 CLR -(SP) ;RESET PRIORITY  
545 004620 012746 004626 MOV #1$,-(SP)  
546 004624 000002 RTI  
547 004626 004737 023502 1$: JSR PC,SETINT ;LOAD VECTOR AREA WITH TRAP CATCHER  
548 004637 012777 004714 174570 MOV #3$,@VECTOR ;INTERRUPT VECTOR ADDRESS  
549 004640 012777 000200 174564 MOV #200,@VECTR1 ;SET UP NEW PSW  
550 004646 012777 000101 174546 MOV #BIT6!BIT0,@STREG ;SET INTERRUPT ENABLE BIT + START (ONVERSION  
551 004654 105777 174542 2$: TSTB @STREG ;WAIT FOR DONE  
552 004660 100375 BPL 2$ ;FLAG TO SET  
553 004662 017737 174534 001126 MOV @STREG,$BDDAT ;READ STATUS REGISTER  
554 004670 005077 174526 CLR @STREG ;ENSURE INTR. ENABLE IS CLEARED  
555 004674 017737 174526 001124 MOV @ADBUFF,$GDDAT ;READ TO CLEAR DONE FLAG  
556 004702 012737 000300 001124 MOV #BIT7!BIT6,$GDDAT ;LOAD EXPECTED GOOD DATA  
557 004710 104002 ERROR 2 ;FAILED TO INTERRUPT ON DONE  
558 004712 004001 BR 4$ ;BRANCH TO NEXT TEST  
559 004714 022626 3$: CMP (SP)+,(SP)+ ;RESET STACK POINTER  
560 004716 013777 001432 174504 4$: MOV VECTR1,@VECTOR ;SET UP FOR UNEXPECTED INTERRUPT  
561 004724 012777 004700 174500 MOV #4700,@VECTR1  
562 004732 005046 CLR -(SP) ;CLEAR PSW  
563 004734 012746 004742 MOV #5$,-(SP)  
564 004740 000002 RTI  
565 004742 005077 174454 5$: CLR @STREG
```

```

566 004746 005777 174454      TST      @ADBUFF      ;CLEAR DONE BIT
567
568
(3)
(3)
(2) 004752 000004
(1) 004754 012737 000100 001160
569 004762 012737 004770 001106
570 004770 042777 000100 174146 10$: BIC      #BIT6,@STKS      ;REMOVE TKB INTERRUPT
571 004776 005046          CLR      -(SP)          ;LOWER PRIORITY
572 005000 012746 005006      MOV      #1$,-(SP)
573 005004 000002          RTI
574 005006 004737 023502      1$: JSR      PC,SETINT      ;LOAD VECTOR AREA WITH TRAP CATCHER
575 005012 012777 005064 174414      MOV      #2$,@VECTR2      ;SETUP VECTOR ADDRESS
576 005020 012777 000200 174410      MOV      #200,@VECTR3      ;SET UP NEW PSW
577 005026 012777 140000 174366      MOV      #BIT15!BIT14,@STREG ;CAUSE AN INTERRUPT
578 005034 017737 174362 001126      MOV      @STREG,$BDDAT      ;BAD DATA
579 005042 012737 140000 001124      MOV      #BIT15!BIT14,$GDDAT ;GOOD DATA
580 005050 005077 174346          CLR      @STREG          ;CLEAR STATUS
581 005054 005777 174346          TST      @ADBUFF          ;AND CLEAR DONE
582 005060 104002          ERROR    2              ;'ERRCR' BIT FAILED TO GENERATE AN INTERRUPT
583 005062 000401          BR       3$
584 005064 022626      2$: CMP      (SP)+,(SP)+      ;POP STACK
585 005066 005077 174330      3$: CLR      @STREG          ;CLEAR STATUS REG.
586 005072 005777 174330          TST      @ADBUFF          ;FALSE READ TO CLEAR DONE
587 005076 013777 001436 174330      MOV      VECTR3,@VECTR2      ;RESET VECTOR
588 005104 012777 004700 174324      MOV      #4700,@VECTR3      ;
589 005112 005046          CLR      -(SP)          ;RESET PRIORITY
590 005114 012746 005122      MOV      #4$,-(SP)
591 005120 000002          RTI
592 005122 005077 174274      4$: CLR      @STREG
593
(3)
(3)
(2) 005126 000004
594 005130 012777 000001 174264
595 005136 052777 000100 174000
596 005144 105777 174252      1$: TSTB     @STREG          ;WAIT FOR
597 005150 100375          BPL     1$
598 005152 012737 100200 001124      2$: MOV      #BIT15!BIT7,$GDDAT ;LOAD EXPECTED VALUE
599 005160 012777 000001 174234      MOV      #BIT0,@STREG      ;START 2ND CONVERSION
600 005166 004737 016552      JSR     PC,STALL          ;DELAY
601 005172 104414      4$: CHECK
602 005174 104001      ERROR    1              ;ERROR FLAG NOT SET WHEN 2ND
603
604 005176 017700 174224      MOV      @ADBUFF,R0      ;CONVERT ENDS BEFORE READ BUFFER FROM FIRST
;CLEAR DONE FLAG

```

606
(3)
(3)
(2) 005202 000004
607 005204 012737 100000 001124
608 005212 012777 000001 174202
609 005220 112777 000001 174174
610 005226 112777 000001 174166
611 005234 017737 174162 001126
612 005242 042737 077777 001126
613 005250 023737 001124 001126
614 005256 001401
615 005260 104001
616
617 005262 105777 174134
618 005266 100375
619 005270 017700 174132
620 005274 005077 174122
621
(3)
(3)
(2) 005300 000004
622 005302 005037 001124
623 005306 012777 000010 174106
624 005314 005277 174102
625 005320 105777 174076
626 005324 100375
627 005326 017737 174074 001126
628 005334 042737 007777 001126
629 005342 001401
630 005344 104001
631 005346 062777 000400 174046
632 005354 032777 004000 174040
633 005362 001754
634
(3)
(3)
(2) 005364 000004
635 005366 005737 001544
636 005372 100020
637 005374 012737 000240 001124
638 005402 013777 001124 174012
639 005410 012777 177776 17404
640 005416 012777 000011 174034
641 005424 004737 016552
642 005430 104414
643 005432 104001
644 005434 005777 173766
645 005440 005077 173756

```
*****  
*TEST 25 TEST ERROR FLAG SETS IF START 2ND CONV. BEFORE DONE FLAG SETS  
*****  
TST25: SCOPE  
MOV #BIT15,$GDDAT ;LOAD EXPECTED DATA  
MOV #BIT0,@STREG ;START CONVERSION  
MOVB #BIT0,@STREG ;START NEXT CONVERSION  
MOVB #BIT0,@STREG ;ONCE AGAIN IN CASE REFRESH INTERVENED  
MOV @STREG,$BDDAT ;READ STATUS REGISTER  
BIC #7777,$BDDAT ;MASK OUT BIT 15  
CMP $GDDAT,$BDDAT ;COMPARE RESULTS  
BEQ 1$ ;BRANCH OVER ERROR  
ERROR 1 ;ERROR FLAG NOT SET WHEN 2ND  
 ;CONVERT BEGINS BEFORE FIRST DONE  
1$: TSTB @STREG ;WAIT FOR DONE  
BPL 1$ ;WAIT  
MOV @ADBUFF,R0  
CLR @STREG ;CLEAR STATUS REGISTER  
*****  
*TEST 26 TEST CHANNELS 0-7 FOR SINGLE ENDED  
*****  
TST26: SCOPE  
CLR $GDDAT  
MOV #BIT3,@STREG ;ENABLE PREAMP STATUS  
1$: INC @STREG ;START A CONVERSION  
2$: TSTB @STREG ;IS CONVERSION DONE?  
BPL 2$ ;NO, WAIT TILL IT IS DONE  
MOV @ADBUFF,$BDDAT ;GET PREAMP STATUS  
BIC #7777,$BDDAT ;MASK OUT CONVERTED VALUE  
BEQ 3$ ;SKIP OVER ERROR IF ZERO  
ERROR 1 ;CHANNEL 0-7 CANNOT EVER BE DIFFERENTIAL  
3$: ADD #BIT8,@STREG ;INCREMENT CHANNEL TO BE TESTED  
BIT #BIT11,@STREG ;IS IT DONE?  
BEQ 1$ ;NO  
*****  
*TEST 27 TEST CLOCK OVERFLOW STARTS A/D (TESTER ONLY)  
*****  
TST27: SCOPE  
TST WFTST ;RUNNING ON TESTER ?  
BPL 2$ ;NO, GO TO NEXT TEST  
MOV #BIT7!BIT5,$GDDAT ;SET UP EXPECTED RESULT  
MOV $GDDAT,@STREG ;ENABLE CLOCK OVERFLOW START  
MOV #177776,@CLKBPR ;SET CLOCK NEAR OVERFLOW  
MOV #11,@CLKCSR ;START CLOCK AT LINE RATE  
JSR PC,STALL ;DELAY  
CHECK ;CHECK RESULT  
ERROR 1 ;DONE FLAG FAILED TO SET  
2$: TST @ADBUFF ;CLEAR DONE FLAG  
CLR @STREG ;INHIBIT CLOCK OVERFLOW START
```

647
(3)
(3)
(2) 005444 000004
648 005446 005737 001546
649 005452 001424
650 005454 012737 000240 001124
651 005462 013777 001124 173732
652 005470 012777 177777 173752
653 005476 012777 000011 173742
654 005504 004737 016552
655 005510 104414
656 005512 104001
657 005514 005777 173706
658 005520 005077 173676
659

```
*****  
*TEST 30 TEST CLOCK OVERFLOW STARTS A/D (IF MNCKW IS AVAILABLE)  
*****  
TST30: SCOPE  
TST KWAD ;TEST IF OPERATOR SAID YES  
BEQ TST31 ;BR IF ANSWER WAS NO  
MOV #BIT7!BITS,$GDDAT ;LOAD EXPECTED  
MOV $GDDAT,@STREG ;LOAD STATUS REG.  
MOV #177777,@KWBPBPR ;LOAD PRESET REGISTER  
MOV #11,@KWCSR ;ENABLE CLOCK  
JSR PC,STALL ;DELAY  
CHECK ;CHECK RESULTS  
ERROR 1 ;DONE FLAG FAILED TO SET WITH CLOCK STARTS  
TST @ADBUFF ;CLEAR DONE FLAG  
CLR @STREG ;INHIBIT CLOCK START
```

660
(3)
(3)
(2) 005524 000004
(1) 005526 012737 000100 001160
661 005534 005737 016536
662 005540 001412
663 005542 004537 012332
664 005546 000200
665 005550 020000
666 005552 004010
667 005554 004537 012332
668 005560 000000
669 005562 000000
670 005564 004010
671

```
*****  
*TEST 31 TEST MNCAD S.E.- DIFF MODE STATUS BIT (TESTER ONLY)  
*****  
TST31: SCOPE  
MOV #100,$TIMES ;DO 100 ITERATIONS  
TST WFAD ;TEST IF TESTING MNCAD  
BEQ TST32 ;BR IF NOT  
JSR R5,TSTSDF ;GO TO SUBROUTINE AND DO THE TESTING  
BIT7 ;1ST IN DIFFERENTIAL MODE  
20000 ;EXPECTED DATA  
4010 ;ON CHANNEL 10  
JSR R5,TSTSDF ;REPEAT  
0 ;THEN IN SINGLE ENDED MODE  
0 ;EXPECTED DATA  
4010 ;ON CHANNEL 10
```

672
(3)
(3)
(2) 005566 000004
(1) 005570 012737 000100 001160
673 005576 005737 016540
674 005602 001412
675 005604 004537 012332
676 005610 000200
677 005612 020000
678 005614 006010
679 005616 004537 012332
680 005622 000000
681 005624 000000
682 005626 010010

```
*****  
*TEST 32 TEST MNCAM S.E.- DIFF MODE STATUS BIT (TESTER ONLY)  
*****  
TST32: SCOPE  
MOV #100,$TIMES ;DO 100 ITERATIONS  
TST WFAM ;TEST IF TESTING MNCAM  
BEQ TST33 ;BR IF NOT  
JSR R5,TSTSDF ;GO TO SUBROUTINE AND DO THE TESTING  
BIT7 ;1ST IN DIFFERENTIAL MODE  
20000 ;EXPECTED DATA  
6010 ;ON CHANNEL 14 <1ST MNCAM ON TESTER IF DIFF.>  
JSR R5,TSTSDF ;REPEAT  
0 ;THEN IN SINGLE ENDED MODE  
0 ;EXPECTED DATA  
10010 ;ON CHANNEL 20 <1ST MNCAM ON TESTER IF S.E.>
```

```
684 (3) (3) (2) 005630 000004 (1) 005632 012737 000001 001160 685 005640 005737 001544 686 005644 001043 687 005646 005737 001372 688 005652 001440 689 005654 013700 044544 690 005660 042700 177700 691 005664 022700 000003 692 005670 001431 693 005672 005737 001176 694 005676 001026 695 005700 104401 032160 696 005704 104401 034466 697 005710 104412 698 005712 005726 699 005714 004537 012332 700 005720 000000 701 005722 020000 702 005724 004010 703 005726 104401 032110 704 005732 104401 034466 705 005736 104412 706 005740 005726 707 005742 004537 012332 708 005746 000000 709 005750 000000 710 005752 004010
```

```
*****  
*TEST 33 TEST MNCAD S.E.- DIFF MODE STATUS BIT (MNCAD-TA ONLY)  
*****  
T33: SCOPE  
MOV #1,STIMES ;DO 1 ITERATION  
TST WFTST ;RUNNING ON TESTER ?  
BNE TST34 ;BR IF YES  
TST ADTA ;IS MNCAD-TA AVAILABLE ?  
BEQ TST34 ;BR IF NO  
MOV CHTABL+10,R0 ;GET CHANNEL #10 TYPE  
BIC #177700,R0 ;MASK OFF OTHER BITS  
CMP #3,R0 ;TEST IF MNCAG  
BEQ TST34 ;BR IF AG CHANNEL-CANT CHANGE SE/DIF IF MNCAG IS CH10  
TST $PASS ;TEST IF FIRST PASS  
BNE TST34 ;BR IF NOT  
TYPE ,SDDIF ;TELL OPERATOR TO SET MNCAD-TA TO DIFFERENTIAL  
TYPE ,CRWR ;TELL OPERATOR TO DEPRESS 'RETURN'  
RDLIN ;WAIT FOR 'CR'  
TST (SP)+ ;CLEAN STACK  
JSR R5,TSTSDF ;GO TO SUBROUTINE TO DO THE TESTING  
0 ;NA  
20000 ;EXPECTED DATA  
4010 ;ON CHANNEL 10  
TYPE ,SDSE ;TELL OPERATOR TO SET MNCAD-TA TO S.E.  
TYPE ,CRWR ;TELL OPERATOR TO DEPRESS 'RETURN'  
RDLIN  
TST (SP)+ ;CLEAN STACK  
JSR R5,TSTSDF ;TEST THE MODE BIT  
0 ;NA  
0 ;EXPECTED DATA  
4010 ;ON CHANNEL 10
```

712
 713
 (3)
 (3)
 (2) 005754 000004
 (1) 005756 012737 000001 001160
 714 005764 005737 016542
 715 005770 001073
 716 005772 013700 044544
 717 005776 042700 177700
 718 006002 022700 000003
 719 006006 001464
 720 006010 000240
 721 006012 000240
 722 006014 000240
 723 006016 000240
 724 006020 005737 001176
 725 006024 001055
 726 006026 012737 000220 001124
 727 006034 013777 001124 173360
 728 006042 005737 001544
 729 006046 100011
 730 006050 052777 000400 173410
 731 006056 042777 000400 173402
 732 006064 004737 016552
 733 006070 000425
 734 006072 004737 016564 2\$:
 735 006076 000424
 736 006100 005737 001372
 737 006104 001421
 738 006106 104401 032344
 739 006112 004737 042342
 740 006116 013746 001564
 (1) 006122 104403
 (1) 006124 001
 (1) 006125 000
 741 006126 104401 034466
 742 006132 104412
 743 006134 005726
 744 006136 042777 100000 173256
 745 006144 104414 3\$:
 746 006146 104001
 747 006150 005777 173252 4\$:
 748 006154 005077 173242
 749
 787

```

*****
*TEST 34 TEST EXTERNAL START STARTS A/D (MNCAD-TA OR TESTER)
*****
TST34: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
TST WFLAG ;:TEST IF TESTING MNCAG ON TESTER
BNE TST35 ;:BR IF YES
MOV CHTABL+10,R0 ;:GET CHANNEL 10 TYPE
BIC #177'00,R0 ;:MASK OFF OTHER BITS
CMP #3,R0 ;:TEST IF CH10 IS A MNCAG CHANNEL
BEQ TST35 ;:BR IF IT IS A MNCAG
NOP
NOP
NOP
TST $PASS ;:TEST IF FIRST PASS
BNE TST35 ;:BR IF NOT FIRST PASS
MOV #BIT7!BIT4,$GDDAT ;:SET UP EXPECTED RESULT
MOV $GDDAT,@STREG ;:ENABLE EXTERNAL START
TST WFTST ;:RUNNING IN TESTER MODE?
BPL 2$ ;:NO
BIS #BIT8,@DRVDR ;:GENERATE EXTERNAL START
BIC #BIT8,@DRVDR ;:RESET BIT
JSR PC,STALL ;:DELAY
BR 3$ ;:TEST RESULTS
JSR PC,AFIRST ;:TEST IF FIRST PASS
BR 4$ ;:BR IF NOT FIRST PASS
TST ADTA ;:IF MNCAD-TA AVAILABLE ?
BEQ 4$ ;:BR IF NO
TYPE ,EXTST ;:TYPE MESSAGE ABOUT EXT. START
JSR PC,WHICHU ;:DETERMINE UNIT #
MOV UNITBD,-(SP) ;:SAVE UNITBD FOR TYPEOUT
TYPOS ;:GO TYPE--OCTAL ASCII
.BYTE 1 ;:TYPE 1 DIGIT(S)
.BYTE 0 ;:SUPPRESS LEADING ZEROS
TYPE ,CRWR ;:TYPE 'TYPE CR WHEN READY'
RDLIN ;:WAIT FOR CR
TST (SP)+ ;:POP WORD OFF STACK
BIC #BIT15,@STREG ;:CLEAR A/D ERROR
3$: CHECK ;:CHECK RESULT
ERROR 1 ;:DONE FLAG FAILED TO SET
4$: TST @ADBUFF ;:CLEAR DONE FLAG
CLR @STREG ;:INHIBIT EXTERNAL START
  
```

```

789
(3)
(3)
(2) 006160 000004
(1) 006162 012737 000100 001160
790 006170 005737 016542
(3) 006174 001470
(1) 006176 012737 006210 001110
(1) 006204 004737 014006
(1)
(1)
(1) 006210 112777 000010 173206
(1) 006216 012737 000001 001124
(1) 006224 017737 173240 001126
(1) 006232 042737 177776 001126
(1) 006240 001001
(1) 006242 104010
(1)
(1)
(1) 006244 012777 000170 173214
(1) 006252 042777 000010 173206
(1)
(1) 006260 112777 000010 173'36
(1)
(1) 006266 005037 001124
(1) 006272 017737 173172 001126
(1) 006300 042737 177776 001126
(1) 006306 001401
(1) 006310 104010
(1)
(1) 006312 105277 173104
(1) 006316 105777 173100
(1) 006322 100375
(1) 006324 017737 173140 001126
(1) 006332 017700 173070
(1) 006336 012737 000001 001124
(1) 006344 042737 177776 001124
(2) 006352 001001
(1) 006354 104010
(1) 006356

:*****
:*TEST 35 VERIFY 'HOLD' FROM MNCAG CHANNEL 10
:*****
TST35: SCOPE
MOV #100,$TIMES ;:DO 100 ITERATIONS
TST WFLAG ;:CHECK IF 'WFLAG' FOUND AN MNCAG
BEQ TST36 ;:BR IF NO MNCAG FOUND
MOV #1,$SLPERR ;:LOAD ERROR RETURN
JSR PC,CLRCHT ;:DO CONVERSION ON AG CHANNELS TO INIT. THE LOGIC
;NOW SELECT CHANNEL 10 BUT DONT TELL THE TESTER TO 'HOLD'
;CHECK FOR FALSE 'MNCAG HOLD'
1$: MOV #10,@ADST1 ;:LOAD MUX WITH MNCAG CHANNEL
MOV #1,$GDDAT ;:LOAD EXPECTED DATA
MOV @DRVDIR,$BDDAT ;:READ TESTER INPUT REGISTER
BIC #177776,$BDDAT ;:MASK OFF OTHER BITS
BNE 2$ ;:BR IF BIT IS ON
ERROR 10 ;:UNEXPECTED 'HOLD' SENSED FROM M.U.T. CHANNEL 10
;NOW TELL THE TESTER TO 'HOLD' THE CHANNEL
;AND VERIFY THAT MNCAG CHANNEL DOES HOLD
2$: MOV #170,@DRVDIR ;:TELL TESTER TO HOLD
BIC #10,@DRVDIR ;:BY SETTING ALL THESE BITS AND CLEARING
;THE BIT FOR THE CHANNEL
MOV #10,@ADST1 ;:RE-CLOCK 'QUAD HOLD BUFFER LATCH'
;IN THE MNCAG 'HOLD' LOGIC
CLR $GDDAT ;:CLEAR EXPECTED VALUE
MOV @DRVDIR,$BDDAT ;:READ TESTER
BIC #177776,$BDDAT ;:CLEAR OFF BITS
BEQ 3$ ;:BR IF BIT IS OFF
ERROR 10 ;:'HOLD' FROM MNCAG FAILED TO SET CHANNEL 10
;NOW CONVERT ON THE SELECTED CHANNEL AND CHECK 'HOLD' CLEARS
3$: INCB @STREG ;:CONVERT
4$: TSTB @STREG ;:WAIT FOR READY
BPL 4$
MOV @DRVDIR,$BDDAT ;:READ TESTER
MOV @ADBUFF,R0 ;:READ 10/D BUFFER
MOV #1,$GDDAT ;:LOAD EXPECTED
BIC #177776,$GDDAT ;:CLEAR OTHER BITS
BNE 5$ ;:BR IF BIT IS OFF
ERROR 10 ;:'MNCAG HOLD' FAILED TO CLEAR FOR CHANNEL 10
5$:

```

```
792
(3)
(3)
(2) 006356 000004
(1) 006360 012737 000100 001160
793 006366 005737 016542
(3) 006372 001470
(1) 006374 012737 006406 001110
(1) 006402 004737 014006
(1)
(1) 006406 112777 000011 173010
(1) 006414 012737 000001 001124
(1) 006422 017737 173042 001126
(1) 006430 042737 177776 001126
(1) 006436 001001
(1) 006440 104010
(1)
(1)
(1) 006442 012777 000170 173016
(1) 006450 042777 000020 173010
(1)
(1) 006456 112777 000011 172740
(1)
(1) 006464 005037 001124
(1) 006470 017737 172774 001126
(1) 006476 042737 177776 001126
(1) 006504 001401
(1) 006506 104010
(1)
(1) 006510 105277 172706
(1) 006514 105777 172702
(1) 006520 100375
(1) 006522 017737 172742 001126
(1) 006530 017700 172672
(1) 006534 012737 000001 001124
(1) 006542 042737 177776 001124
(2) 006550 001001
(1) 006552 104010
(1) 006554

:*****
:*TEST 36 VERIFY 'HOLD' FROM MNCAG CHANNEL 11
:*****
TST36: SCOPE
MOV #100,$TIMES ;DO 100 ITERATIONS
TST WFAG ;CHECK IF 'WFCHK' FOUND AN MNCAG
BEQ TST37 ;BR IF NO MNCAG FOUND
MOV #1,$SLPERR ;LOAD ERROR RETURN
JSR PC,CLRCHT ;DO CONVERSION ON AG CHANNELS TO INIT. THE LOGIC
;NOW SELECT CHANNEL 11 BUT DONT TELL THE TESTER TO 'HOLD'
;CHECK FOR FALSE 'MNCAG HOLD'
1$: MOVB #11,@ADST1 ;LOAD MUX WITH MNCAG CHANNEL
MOV #1,$GDDAT ;LOAD EXPECTED DATA
MOV @DRVDIR,$BDDAT ;READ TESTER INPUT REGISTER
BIC #177776,$BDDAT ;MASK OFF OTHER BITS
BNE 2$ ;BR IF BIT IS ON
ERROR 10 ;UNEXPECTED 'HOLD' SENSED FROM M.U.T. CHANNEL 11
;NOW TELL THE TESTER TO 'HOLD' THE CHANNEL
;AND VERIFY THAT MNCAG CHANNEL DOES HOLD
2$: MOV #170,@DRVDIR ;TELL TESTER TO HOLD
BIC #20,@DRVDIR ;BY SETTING ALL THESE BITS AND CLEARING
;THE BIT FOR THE CHANNEL
;RE-CLOCK 'QUAD HOLD BUFFER LATCH'
;IN THE MNCAG 'HOLD' LOGIC
CLR $GDDAT ;CLEAR EXPECTED VALUE
MOV @DRVDIR,$BDDAT ;READ TESTER
BIC #177776,$BDDAT ;CLEAR OFF BITS
BEQ 3$ ;BR IF BIT IS OFF
ERROR 10 ;'HOLD' FROM MNCAG FAILED TO SET CHANNEL 11
;NOW CONVERT ON THE SELECTED CHANNEL AND CHECK 'HOLD' CLEARS
3$: INCB @STREG ;CONVERT
4$: TSTB @STREG ;WAIT FOR READY
BP_ 4$
MOV @DRVDIR,$BDDAT ;READ TESTER
MOV @ADBUFF,R0 ;READ 11/D BUFFER
MOV #1,$GDDAT ;LOAD EXPECTED
BIC #177776,$GDDAT ;CLEAR OTHER BITS
BNE 5$ ;BR IF BIT IS OFF
ERROR 10 ;'MNCAG HOLD' FAILED TO CLEAR FOR CHANNEL 11
5$:
```



```

795
(3)
(3)
(2) 006554 000004
(1) 006556 012737 000100 001160
796 006564 005737 016542
(3) 006570 001470
(1) 006572 012737 006604 001110
(1) 006600 004737 014006
(1)
(1)
(1) 006604 112777 000012 172612
(1) 006612 012737 000001 001124
(1) 006620 017737 172644 001126
(1) 006626 042737 177776 001126
(1) 006634 001001
(1) 006636 104010
(1)
(1)
(1) 006640 012777 000170 172620
(1) 006646 042777 000040 172612
(1)
(1) 006654 112777 000012 172542
(1)
(1) 006662 005037 001124
(1) 006666 017737 172576 001126
(1) 006674 042737 177776 001126
(1) 006702 001401
(1) 006704 104010
(1)
(1) 006706 105277 172510
(1) 006712 105777 172504
(1) 006716 100375
(1) 006720 017737 172544 001126
(1) 006726 017700 172474
(1) 006732 012737 000001 001124
(1) 006740 042737 177776 001124
(2) 006742 001001
(1) 006750 104010
(1) 006752
  
```

```

:*****
:*TEST 37 VERIFY 'HOLD' FROM MNCAG CHANNEL 12
:*****
TST37: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
TST WFLAG ;;CHECK IF 'WFLAG' FOUND AN MNCAG
BEQ TST40 ;;BR IF NO MNCAG FOUND
MOV #1$, $LPERR ;;LOAD ERROR RETURN
JSR PC, CLRCHT ;;DO CONVERSION ON AG CHANNELS TO INIT. THE LOGIC
;NOW SELECT CHANNEL 12 BUT DONT TELL THE TESTER TO 'HOLD'
CHECK FOR FALSE 'MNCAG HOLD'
1$: MOVB #12, @ADST1 ;;LOAD MUX WITH MNCAG CHANNEL
MOV #1, $GDDAT ;;LOAD EXPECTED DATA
MOV @DRVDIR, $BDDAT ;;READ TESTER INPUT REGISTER
BIC #177776, $BDDAT ;;MASK OFF OTHER BITS
BNE 2$ ;;BR IF BIT IS ON
ERROR 10 ;;UNEXPECTED 'HOLD' SENSED FROM M.U.T. CHANNEL 12
;NOW TELL THE TESTER TO 'HOLD' THE CHANNEL
AND VERIFY THAT MNCAG CHANNEL DOES HOLD
2$: MOV #170, @DRVDIR ;;TELL TESTER TO HOLD
BIC #40, @DRVDIR ;;BY SETTING ALL THESE BITS AND CLEARING
;;THE BIT FOR THE CHANNEL
MOVB #12, @ADST1 ;;RE-CLOCK 'QUAD HOLD BUFFER LATCH'
;;IN THE MNCAG 'HOLD' LOGIC
CLR $GDDAT ;;CLEAR EXPECTED VALUE
MOV @DRVDIR, $BDDAT ;;READ TESTER
BIC #177776, $BDDAT ;;CLEAR OFF BITS
BEQ 3$ ;;BR IF BIT IS OFF
ERROR 10 ;;'HOLD' FROM MNCAG FAILED TO SET CHANNEL 12
;NOW CONVERT ON THE SELECTED CHANNEL AND CHECK 'HOLD' CLEARS
3$: INCB @STREG ;;CONVERT
4$: TSTB @STREG ;;WAIT FOR READY
BPL 4$
MOV @DRVDIR, $BDDAT ;;READ TESTER
MOV @ADBUFF, R0 ;;READ 12/D BUFFER
MOV #1, $GDDAT ;;LOAD EXPECTED
BIC #177776, $GDDAT ;;CLEAR OTHER BITS
BNE 5$ ;;BR IF BIT IS OFF
ERROR 10 ;;'MNCAG HOLD' FAILED TO CLEAR FOR CHANNEL 12
5$:
  
```

798
(3)
(3)
(2) 006752 000004
(1) 006754 012737 000100 001160
799 006762 005737 016542
(3) 006766 001470
(1) 006770 012737 007002 001110
(1) 006776 004737 014006
(1)
(1)
(1) 007002 112777 000013 172414
(1) 007010 012737 000001 001124
(1) 007016 017737 172446 001126
(1) 007024 042737 177776 001126
(1) 007032 001001
(1) 007034 104010
(1)
(1)
(1) 007036 012777 000170 172422
(1) 007044 042777 000100 172414
(1)
(1) 007052 112777 000013 172344
(1)
(1) 007060 005037 001124
(1) 007064 017737 172400 001126
(1) 007072 042737 177776 001126
(1) 007100 001401
(1) 007102 104010
(1)
(1) 007104 105277 172312
(1) 007110 105777 172306
(1) 007114 100375
(1) 007116 017737 172346 001126
(1) 007124 017700 172276
(1) 007130 012737 000001 001124
(1) 007136 042737 177776 001124
(2) 007144 001001
(1) 007146 104010
(1) 007150
800

```
*****  
*TEST 40 VERIFY 'HOLD' FROM MNCAG CHANNEL 13  
*****  
TST40: SCOPE  
MOV #100,$TIMES ;:DO 100 ITERATIONS  
TST WFAG ;:CHECK IF 'WFCHK' FOUND AN MNCAG  
BEQ TST41 ;:BR IF NO MNCAG FOUND  
MOV #1$,SLPERR ;:LOAD ERROR RETURN  
JSR PC,CLRCHT ;:DO CONVERSION ON AG CHANNELS TO INIT. THE LOGIC  
:NOW SELECT CHANNEL 13 BUT DONT TELL THE TESTER TO 'HOLD'  
:CHECK FOR FALSE 'MNCAG HOLD'  
1$: MOV #13,@ADST1 ;:LOAD MUX WITH MNCAG CHANNEL  
MOV #1,$GDDAT ;:LOAD EXPECTED DATA  
MOV @DRVDIR,$BDDAT ;:READ TESTER INPUT REGISTER  
BIC #177776,$BDDAT ;:MASK OFF OTHER BITS  
BNE 2$ ;:BR IF BIT IS ON  
ERROR 10 ;:UNEXPECTED 'HOLD' SENSED FROM M.U.T. CHANNEL 13  
:NOW TELL THE TESTER TO 'HOLD' THE CHANNEL  
AND VERIFY THAT MNCAG CHANNEL DOES HOLD  
2$: MOV #170,@DRVDOR ;:TELL TESTER TO HOLD  
BIC #100,@DRVDOR ;:BY SETTING ALL THESE BITS AND CLEARING  
;THE BIT FOR THE CHANNEL  
MOV #13,@ADST1 ;:RE-CLOCK 'QUAD HOLD BUFFER LATCH'  
;IN THE MNCAG 'HOLD' LOGIC  
CLR $GDDAT ;:CLEAR EXPECTED VALUE  
MOV @DRVDIR,$BDDAT ;:READ TESTER  
BIC #177776,$BDDAT ;:CLEAR OFF BITS  
BEQ 3$ ;:BR IF BIT IS OFF  
ERROR 10 ;:'HOLD' FROM MNCAG FAILED TO SET CHANNEL 13  
:NOW CONVERT ON THE SELECTED CHANNEL AND CHECK 'HOLD' CLEARS  
3$: INCB @STREG ;:CONVERT  
4$: TSTB @STREG ;:WAIT FOR READY  
BPL 4$  
MOV @DRVDIR,$BDDAT ;:READ TESTER  
MOV @ADBUFF,R0 ;:READ 13/D BUFFER  
MOV #1,$GDDAT ;:LOAD EXPECTED  
BIC #177776,$GDDAT ;:CLEAR OTHER BITS  
BNE 5$ ;:BR IF BIT IS OFF  
ERROR 10 ;:'MNCAG HOLD' FAILED TO CLEAR FOR CHANNEL 13  
5$:
```

802
(3)
(3)
(2) 007150 000004
(1) 007152 012737 000400 001160
803
804
805
806
807
808 007160 012737 000010 014664
809
810 007166 013700 014664
811 007172 004737 011442
812 007176 103002
813 007200 004737 011572
814 007204 005237 014664
815 007210 022737 000100 014664
816 007216 001363
817
818
819
(5)
(3)
(2) 007220 000004
(1) 007222 012737 000001 001160
820 007230 000207

```
*****  
: *TEST 41 MNCAG GAIN BITS LOGIC TESTS  
*****  
TST41: SCOPE  
MOV #400,$TIMES ;;DO 400 ITERATIONS  
:NOW TO PROVE THAT THE MNCAG LOGIC IS WORKING CORRECTLY  
1ST. WRITE CH00-77 WITH GAIN BITS = 01  
2ND. WRITE CHXX WITH GAIN BITS = 10  
3RD. READ CHXX AND CHECK GAIN BITS = 10  
4TH. READ CH00-77 EXCEPT CHXX AND CHECK GAIN STILL = 01  
MOV #10,CHXX ;PRIME THE CHANNEL UNDER TEST TO 10  
  
1$: MOV CHXX,R0 ;GET CHANNEL VALUE  
JSR PC,CHKAGC ;CHECK IF THIS IS AN MNCAG CHANNEL  
BCC 2$ ;BR IF NOT  
JSR PC,CHKAN ;READ-WRITE TEST OF GAIN BITS  
2$: INC CHXX ;UPDATE TESTED CHANNEL  
CMP #100,CHXX ;TEST IF ALL CHANNELS HAVE BEEN RUN  
BNE 1$ ;BR IF NOT  
  
*****  
: *TEST 42 END OF MNCAD, MNCAG LOGIC TESTS  
*****  
TST42: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
RTS PC
```

822
823 007232
824
(3)
(3)
(2) 007232 012737 000043 001102
(1) 007240 012737 000010 001160
825 007246 012737 007232 001106
(2) 007254 012737 007232 001110
826 007262 004537 025702
827 007266 000000
828 007270 004537 026032
829 007274 004000
830 007276 026674
831 007300 104004
832
(3)
(3)
(2) 007302 000004
(1) 007304 012737 000010 001160
833 007312 004537 025702
834 007316 000001
835 007320 004537 026032
836 007324 007344
837 007326 026702
838 007330 104004
839
(3)
(3)
(2) 007332 000004
(1) 007334 012737 000010 001160
840 007342 004537 025702
841 007346 000002
842 007350 004537 026032
843 007354 000434
844 007356 026702
845 007360 104004
846
(3)
(3)
(2) 007362 000004
(1) 007364 012737 000010 001160
847 007372 005737 016542
848 007376 001402
849 007400 000137 010034
850 007404 005737 001544
851 007410 001003
852 007412 105737 044541
853 007416 100040
854 007420 004537 025702
855 007424 000005
856 007426 004537 026032
857 007432 004000
858 007434 026674
859 007436 104004
860

.SBTTL WRAPAROUND ANALOG TEST SECTION
WRAP:

*TEST 43 TEST CHO GROUND

TST43: MOV #STN,\$STNM
MOV #10,\$TIMES ;:DO 10 ITERATIONS
MOV #TST43,\$LPADR ;:SET UP LOOP ADDRESS
MOV #TST43,\$LPERR ;:SET UP ERROR LOOP ADDRESS
JSR R5,CONVRT ;:CONVERT 8 TIMES
0
JSR R5,COMPAR ;:COMPARE RESULTS
4000 ;:NOMINAL
V12 ;:TOLLERANCE
ERROR 4 ;:ERROR ON A/D CHANNEL

*TEST 44 TEST CH1 +4.5 VOLT

TST44: SCOPE
MOV #10,\$TIMES ;:DO 10 ITERATIONS
JSR R5,CONVRT ;:CONVERT 8 TIMES
1 ;:CHANNEL 1
JSR R5,COMPAR ;:COMPARE RESULTS
7344 ;:NOMINAL
V326 ;:TOLLERANCE
ERROR 4 ;:ERROR ON A/D CHANNEL

*TEST 45 TEST CH2 -4.5 VOLT

TST45: SCOPE
MOV #10,\$TIMES ;:DO 10 ITERATIONS
JSR R5,CONVRT ;:CONVERT 8 TIMES
2 ;:CHANNEL 2
JSR R5,COMPAR ;:COMPARE RESULTS
434 ;:NOMINAL
V326 ;:TOLLERANCE
ERROR 4 ;:ERROR ON A/D CHANNEL

*TEST 46 TEST CH5 GROUND (MNCAD-TA OR TESTER EXCEPT IF MNCAG)

TST46: SCOPE
MOV #10,\$TIMES ;:DO 10 ITERATIONS
TST WFLAG ;:TEST IF TESTING MNCAG'S
BEQ 1\$;:BR IF NOT
JMP WRAPY ;:BYPASS MANY TESTS
1\$: TST WFTST ;:RUNNING ON THE TESTER ?
BNE 2\$;:BR :F YES
TSTB CHTABL+5 ;:TEST IF TESTING CH4-7 ?
BPL WRAPX ;:BYPASS SOME TESTS
2\$: JSR R5,CONVRT ;:CONVERT 8 TIMES
5 ;:CHANNEL 5
JSR R5,COMPAR ;:COMPARE RESULTS
4000 ;:NOMINAL
V12 ;:TOLLERANCE
ERROR 4 ;:ERROR ON A/D CHANNEL

862
(3)
(3)
(2) 007440 000004
(1) 007442 012737 000010 001160
863 007450 004537 025702
864 007454 000004
865 007456 004537 026032
866 007462 006020
867 007464 026702
868 007466 104004
869
870
(3)
(3)
(2) 007470 000004
(1) 007472 012737 000010 001160
871 007500 004537 025702
872 007504 000006
873 007506 004537 026032
874 007512 001760
875 007514 026702
876 007516 104004
877
878 007520
879
(3)
(3)
(2) 007520 000004
(1) 007522 012737 000010 001160
880 007530 012737 000051 001102
881 007536 012702 044544
882 007542 105712
883 007544 001450
884 007546 100045
885 007550 111237 017460
886 007554 042737 177700 017460
887 007562 022737 000001 017460
888 007570 001034
889 007572 010203
890 007574 162703 044534
891 007600 010337 001520
892 007604 012703 026704
893 007610 012337 007624
894 007614 004537 025710
895 007620 004537 026032
896 007624 005560
897 007626 026702
898 007630 104004
899 007632 022737 000077 001520
900 007640 001412
901 007642 005237 001520
902 007646 005713
903 007650 100357
904 007652 062702 000007
905 007656 000240

```
*****  
: *TEST 47 TEST CH4 +2.6 VOLTS (MNCAD-TA OR TESTER)  
*****  
TST47: SCOPE  
MOV #10,$TIMES ;DO 10 ITERATIONS  
JSR R5,CONVRT ;CONVERT 8 TIMES  
4 ;CHANNEL 4  
JSR R5,COMPAR ;COMPARE RESULTS  
6020 ;NOMINAL  
V326 ;TOLLERANCE  
ERROR 4 ;ERROR ON A/D CHANNEL
```

```
*****  
: *TEST 50 TEST CH6 -2.2 VOLTS (MNCAD-TA OR TESTER)  
*****  
TST50: SCOPE  
MOV #10,$TIMES ;DO 10 ITERATIONS  
JSR R5,CONVRT ;CONVERT 8 TIMES  
6 ;CHANNEL 6  
JSR R5,COMPAR ;COMPARE RESULTS  
1760 ;NOMINAL  
V326 ;TOLLERANCE  
ERROR 4 ;ERROR ON A/D CHANNEL
```

WRAPX:

```
*****  
: *TEST 51 TEST VOLTAGE ON SINGLE-ENDED CHANNELS (MNCAD-TA OR MNCAM-TA OR TESTER)  
*****  
TST51: SCOPE  
MOV #10,$TIMES ;DO 10 ITERATIONS  
MOV #$STN-1,$STNM ;SET UP TEST NUMBER  
MOV #CHTABL+10,R2 ;LOAD POINTER TO CHANNEL LIST  
1$: TSTB (R2) ;TEST IF EXISTANT CHANNEL  
BEQ 4$ ;BR IF NO MORE CHANNELS  
BPL 3$ ;BR IF NOT TO TEST THIS CHANNEL  
MOVB (R2),CHA ;GET TYPE OF CHANNEL  
BIC #177700,CHA ;MASK OFF OTHER BITS  
CMP #1,CHA ;TEST IF A SINGLE ENDED CHANNEL  
BNE 3$ ;BR IF NOT S.E. CHANNEL  
MOV R2,R3 ;COPY R2  
SUB #CHTABL,R3 ;CONVERT INDEX INTO CHANNEL NUMBER  
MOV R3,CHANL ;SAVE CHANNEL NUMBER  
MOV #VTABLE,R3 ;MAKE INDEX INTO EXPECTED VALUE TABLE  
5$: MOV (R3)+,2$ ;GET EXPECTED VALUE  
JSR R5,CONVRT ;CONVERT 8 TIMES  
JSR R5,COMPAR ;COMPARE RESULTS  
2$: 5560 ;VOLTAGE  
V326 ;TOLLERANCE  
ERROR 4 ;ERROR ON SINGLE ENDED A/D CHANNEL  
CMP #77,CHANL ;TEST IF LAST CHANNEL IN SYSTEM  
BEQ 4$ ;BR IF LAST  
INC CHANL ;UPDATE CHANNEL NUMBER  
TST (R3) ;TEST IF END OF LIST  
BPL 5$ ;BR IF NOT  
ADD #7,R2 ;UPDATE CHANNEL LOOKUP VALUE  
NOP
```

906 007660 000240
 907 007662 105722
 908 007664 000726
 909 007666 000240
 910
 911
 (3)
 (3)
 (2) 007670 000004
 (1) 007672 012737 000001 001160
 912
 913 007700 012702 044544
 914 007704 012737 007720 001106
 915 007712 012737 007720 001110
 916 007720 105712
 917 007722 001443
 918 007724 100040
 919 007726 111237 017460
 920 007732 042737 177700 017460
 921 007740 022737 000002 017460
 922 007746 001027
 923 007750 010203
 924 007752 162703 044534
 925 007756 010337 001520
 926 007762 012737 002220 010020
 927 007770 032703 000001
 928 007774 001405
 929 007776 005437 010020
 930 010002 042737 170000 010020
 931 010010 004537 025710
 932 010014 004537 026032
 933 010020 002220
 934 010022 026702
 935 010024 104004
 936 010026 105722
 937 010030 000733
 938 010032 000240

```

NOP
3$: TSTB (R2)+ ;BUMP CHANNEL POINTER
BR 1$ ;TEST NEXT CHANNEL
4$: NOP

*****
*TEST 52 TEST VOLTAGE ON DIFFERENTIAL CHANNELS (MNCAD-TA OR MNCAM-TA OR TESTER)
*****
TST52: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION

MOV #CHTABL+10,R2 ;LOAD POINTER TO CHANNEL LIST
MOV #1$,SLPADR ;SET UP LOOP ADDRESS
MOV #1$,SLPERR ;SET UP ERROR LOOP ADDRESS
1$: TSTB (R2) ;TEST IF EXISTANT CHANNEL
BEQ 4$ ;BR IF NOT
BPL 3$ ;BR IF NOT TO TEST THE CHANNEL
MOV# (R2),CHA ;GET CHANNEL TYPE
BIC #177700,CHA ;MASK OFF OTHER BITS
CMP #2,CHA ;TEST IF DIFFERENTIAL CHANNEL
BNE 3$ ;BR IF NOT A DIFF. CHANNEL
MOV R2,R3 ;COPY R2
SUB #CHTABL,R3 ;CREATE CHANNEL NUMBER FROM OFFSET
MOV R3,CHANL ;SAVE CHANNEL NUMBER
MOV #2220,2$ ;SET UP INITIAL EXPECTED VALUE -2.2 V
BIT #BIT0,R3 ;TEST IF ODD OR EVEN CHANNEL
BEQ 5$ ;BR IF EVEN CHANNEL
NEG 2$ ;CONVERT EXPECTED VALUE
BIC #170000,2$ ;MASK OFF OTHER BITS
5$: JSR R5,CONVTC ;CONVERT 8 TIMES
JSR R5,COMPAR ;TEST RESULTS
2$: 2220 ;NOMINAL
V326 ;TOLLERANCE
ERROR 4 ;ERROR ON A/D CHANNEL
3$: TSTB (R2)+ ;PUMP THE CHANNEL POINTER
BR 1$ ;RETEST
4$: NOP
  
```

940 010034
941
(3)
(3)
(2) 010034 000004
(1) 010036 012737 000001 001160
942 010044 012737 000053 001102
943 010052 005077 171350
944 010056 005037 001520
945 010062 004537 025716
946 010066 013704 001502
947 010072 012777 000377 171326 1S:
948 010100 004537 025716
949 010104 160437 001502
950 010110 004537 026032
951 010114 000005
952 010116 026670
953 010120 104004
954
(3)
(3)
(2) 010122 000004
(1) 010124 012737 000001 001160
955 010132 104401 027723
956 010136 004737 042342
957 010142 013746 001564
958 010146 104403
959 010150 001 000
960 010152 005037 001520
961 010156 005037 001516
962 010162 004737 012230
963 010166 104401 035727
964 010172 004737 012450
965 010176 004537 026032
966 010202 000000
967 010204 026676
968 010206 000401
969 010210 000407
970 010212 104401 034611
971 010216 004737 042334
972 010222 005237 001112
973 010226 000402
974 010230 104401 034135

WRAPY:
:*****
:*TEST 53 TEST VERNIER OFFSET DAC ON CHO
:*****
TST53: SCOPE
MOV #1,\$TIMES ;:DO 1 ITERATION
MOV #STN-1,\$STNM ;:SET UP TEST NUMBER
CLR @ADBUFF ;:SET VERNIER DAC = 0
CLR CHANL ;:SET UP TO CONVERT ON CHANNEL 0
JSR R5,CONVCD ;:CONV. CHO, DIRECT VERNIER DAC
MOV TEMP,R4 ;:SAVE VALUE IN R4
MOV #377,@ADBUFF ;:SET VERNIER DAC - 377
JSR R5,CONVCD ;:CONVERT IT
SUB R4,TEMP ;:TEMP=DIFF. BETWEEN VALUE & PREVIOUS
JSR R5,COMPAR ;:COMPARE RESULTS
S
V2
ERROR 4
:*****
:*TEST 54 OFFSET ON CHO
:*****
TST54: SCOPE
MOV #1,\$TIMES ;:DO 1 ITERATION
TYPE ,OFFSET ;:INFORM OPER. TEST NAME
JSR PC,WHICHU ;:GET UNIT #
MOV UNITBD,-(SP) ;:PUSH IT
TYPOS ;:TELL OPER.
BYTE 1,0
CLR CHANL ;:LOAD CHANNEL
CLR DUMMY ;:LOAD DUMMY
JSR PC,OFFSET ;:FIND OFFSET
TYPE ,MOFFSET ;:TYPE 'OFFSET='
JSP PC,TOFF ;:TYPE OFFSET
JSR R5,COMPAR ;:IS RESULT WITHIN LIMITS?
O
V50D
BR OFFERR ;:NO-ERROR
BR OFFOK ;:YES-OK
OFFERR: TYPE ,ERMSG
JSR PC,WHICHV ;:INDICATE BAD UNIT
INC \$ERTTL ;:UPDATE ERROR COUNT
BR TST55 ;:GO TO NEXT TEST
OFFOK: TYPE ,OKMSG

```
976          ;:*****  
(3)          ;*TEST 55      TEST RAMP RANGE, CH3  
(3)          ;:*****  
(2) 010234 000004  
977 010236 012737 000001 001160 TST55: SCOPE  
978 010244 012703 007777          MOV #1,STIMES          ;DO THIS ONCE  
979 010250 005004          MOV #7777,R3          ;INIT R3 VALUE  
980 010252 012777 001400 171142 CLR R4          ;AND R4  
981 010260 012702 047040          MOV #1400,@STREG      ;SETUP FOR CH3  
982 010264 105277 171132          MOV #20000.,R2       ;SETUP FOR 20,000 CONVERSIONS  
983 010270 105777 171126 1$: INCB @STREG  
984 010274 100375          TSTB @STREG  
985 010276 027704 171124          BPL 2$  
986 010302 003402          CMP @ADBUFF,R4  
987 010304 017704 171116          BLE 3$          ;HIT A NEW HIGH  
988 010310 027703 171112 3$: CMP @ADBUFF,R3  
989 010314 002002          BGE 4$  
990 010316 017703 171104          MOV @ADBUFF,R3      ;HIT A NEW LOW  
991 010322 005302 4$: DEC R2  
992 010324 001357          BNE 1$  
993 010326 010337 001502          MOV R3,TEMP  
994 010332 004537 026032          JSR R5,COMPAR  
995 010336 000000          O  
996 010340 026666          VO  
997 010342 104004          ERROR 4          ;RAMP DIDN'T REACH LOW END OF RANGE  
998 010344 010437 001502          MOV R4,TEMP  
999 010350 004537 026032          JSR R5,COMPAR  
1000 010354 007777          7777  
1001 010356 026666          VO  
1002 010360 104004          ERROR 4          ;RAMP DIDN'T REACH HIGH END OF RANGE  
1003  
1004          ;:*****  
(3)          ;*TEST 56      NOISE TEST, 1 EDGE      (SINGLE ENDED AND MNCAG CHANNELS ONLY)  
(3)          ;:*****  
(2) 010362 000004  
(1) 010364 012737 000001 001160 TST56: SCOPE  
1005 010372 005037 001472          MOV #1,STIMES          ;DO 1 ITERATION  
1006 010376 004737 010406          CLR WIDE          ;CLEAR ENTRY FLAG  
1007 010402 000137 011124          JSR PC,NOITST      ;RUN NOISE TEST  
          JMP NOIJMP          ;NEXT TEST
```



```

1009      :MAJOR SUBROUTINE THAT DOES THE NOISE TESTING
1010 010406 104401 027640 NOITST: TYPE ,NOIMSG
1011 010412 004737 042342 JSR PC,WHICHU ;DETERMINE UNIT #
1012 010416 013746 001564 MOV UNITBD,-(SP)
1013 010422 104403 TYPOS ;TELL OPER.
1014 010424 001 000 .BYTE 1,0
1015 010426 104401 001165 TYPE ,$CRLF
1016 010432 005737 001472 TST WIDE ;TEST IF MANUAL ENTRY
1017 010436 001010 BNE NOITS1 ;BR IF MANUAL
1018 010440 005037 001520 CLR CHANL ;INITLIZE TO CHAN 0
1019 010444 005737 016540 TST WFAM ;RUNNING MNCAM'S ON THE TESTER
1020 010450 001493 BEQ NOITS1 ;BR IF NOT
1021 010452 012737 000020 001520 MOV #20,CHANL ;TESTING AM
1022      :DETERMINE IF CHANNEL IS TO BE TESTED
1023 010460 013700 001520 NOITS1: MOV CHANL,R0 ;LOAD R0
1024 010464 005737 001472 TST WIDE ;TEST ENTRY FLAG
1025 010470 001007 BNE 2$ ;BR IF MANUAL ENTRY
1026 010472 105760 044534 TSTB CHTABL(R0) ;TEST IF EXISTANT CHANNEL
1027 010476 001001 BNF 1$ ;BR IF DONE
1028 010500 000207 RTS PC ;EXIT
1029 010502 100402 1$: BMI 2$ ;BR IF OPER SAID TO TEST THIS CHANNEL
1030 010504 000137 011056 JMP UPCHAN
1031 010510 016037 044534 011122 2$: MOV CHTABL(R0),CHANIS ;GET CHANNEL TYPE
1032 010516 042737 177700 011122 BIC #177700,CHANIS ;MASK OFF BITS
1033 010524 022737 000003 011122 CMP #3,CHANIS ;TEST IF MNCAG CHANNEL
1034 010532 001135 BNE 4$ ;BR IF NOT
1035      :CHANNEL IS A MNCAG
1036 010534 104401 033271 TYPE ,GANP5 ;TELL OPER. THAT GAIN OF .5
1037 010540 112777 000077 170656 MOVB #77,@ADST1 ;ESC.
1038 010546 112777 000000 170650 MOVB #0,@ADST1 ;LOAD GAIN BITS TO 0
1039 010554 113777 001520 170642 MOVB CHANL,@ADST1 ;SELECT CHANNEL
1040
1041 010562 004537 012006 JSR R5,RMSPEK ;DO RMS NOISE TESTING
1042 010566 020 124 .BYTE 16,.84. ;RMS VALUES
1043 010570 034066 .WORD RMSNOI ;RMS MESSAGE TEXT POINTER
1044 010572 026732 VNRAGO ;POINTER TO TOLERANCE
1045
1046 010574 004537 012006 JSR R5,RMSPEK ;DO PEAK NOISE TESTING
1047 010600 001 143 .BYTE 1,.99. ;PEAK VALUES
1048 010602 034102 .WORD PKNOI ;PEAK MESSAGE TEXT POINTER
1049 010604 026734 VNPAGO ;POINTER TO TOLERANCE
1050
1051 010606 104401 033316 TYPE ,GAN5P ;TELL OPERATOR GAIN IS NOW 5.0
1052 010612 112777 000077 170604 MOVB #77,@ADST1 ;SELECT
1053 010620 112777 000001 170576 MOVB #01,@ADST1 ;
1054 010626 113777 001520 170570 MOVB CHANL,@ADST1 ; GAIN
1055 010634 004537 012006 JSR R5,RMSPEK ;DO RMS TESTING OF 5.
1056 010640 020 124 .BYTE 16,.84. ;RMS VALUES
1057 010642 034066 .WORD RMSNOI ;RMS MESSAGE TEXT POINTER
1058 010644 026736 VNRAG1 ;POINTER TO TOLERANCE
1059
1060 010646 004537 012006 JSR R5,RMSPEK ;DO PEAK NOISE TESTING
1061 010652 001 143 .BYTE 1,.99. ;PEAK VALUES
1062 010654 034102 .WORD PKNOI ;PEAK MESSAGE TEXT POINTER
1063 010656 026740 VNPAG1 ;POINTER TO TOLERANCE
1064

```

```

1065 010660 104401 033343      TYPE      ,GAN5D      ;TELL OPERATOR GAIN IS NOW 50.
1066 010664 112777 000077 170532  MOVB      #77,@ADST1 ;SELECT
1067 010672 112777 000002 170524  MOVB      #2,@ADST1 ;
1068 010700 113777 001520 170516  MOVB      CHANL,@ADST1 ;          GAIN
1069 010706 013737 026742 020444  MOV      VRAG2A,AGCHRA ;          OF 50.
1070 010714 013737 026744 020446  MOV      VRAG2B,AGCHRB ;LOAD MSW OF RMS LIMIT
1071 010722 013737 026746 020644  MOV      VPAG2A,AGCHPA ;LOAD LSW OF RMS LIMIT
1072 010730 013737 026750 020646  MOV      VPAG2B,AGCHPB ;LOAD MSW OF PEAK LIMIT
1073 010736 004737 017464      JSR      PC,PRI4A   ;LOAD LSW OR PEAK LIMIT
1074                                ;DO NOISE TESTING USING DIFFERENT METHOD
1075 010742 104401 033371      TYPE      ,GAN5T      ;TELL OPERATOR GAIN IS NOW 500
1076 010746 112777 000077 170450  MOVB      #77,@ADST1 ;SELECT
1077 010754 112777 000003 170442  MOVB      #3,@ADST1 ;
1078 010762 113777 001520 170434  MOVB      CHANL,@ADST1 ;          GAIN
1079 010770 013737 026752 020444  MOV      VRAG3A,AGCHRA ;          OF 500
1080 010776 013737 026754 020446  MOV      VRAG3B,AGCHRB ;LOAD MSW OF RMS LIMIT
1081 011004 013737 026756 020644  MOV      VPAG3A,AGCHPA ;LOAD LSW OF RMS LIMIT
1082 011012 013737 026760 020646  MOV      VPAG3B,AGCHPB ;LOAD MSW OF PEAK LIMIT
1083 011020 004737 017464      JSR      PC,PRI4A   ;LOAD LSW OF PEAK LIMIT
1084 011024 000414      BR      UPCHAN     ;DO NOISE TESTING USING DIFFERENT METHOD
1085                                ;CHECK NEXT CHANNEL
1086                                ;CHANNEL IS A MNCAD/MNCAM
1087 011026 004537 012006      JSR      R5,RMSPEK ;DO RMS NOISE TESTING
1088 011032      020      124      .BYTE     16.,84. ;RMS VALUES
1089 011034 034066      RMSNOI ;RMS MESSAGE TEXT POINTER
1090 011036 026726      VNR     ;POINTER TO TOLERANCE
1091
1092 011040 004537 012006      JSR      R5,RMSPEK ;DO PEAK NOISE TESTING
1093 011044      001      143      .BYTE     1.,99. ;PEAK VALUES
1094 011046 034102      PKNOI ;PEAK MESSAGE TEXT POINTER
1095 011050 026730      VNP     ;POINTER TO TOLERANCE
1096 011052 104401 001165      TYPE     ,SCLRF
1097
1098                                ;NOW UPDATE CHANNEL NUMBER AND DETERMINE IF MORE CHANNELS ARE TO BE TESTED
1099 011056 005737 001472  UPCHAN: TST     WIDE ;CHECK ENTRY FLAG
1100 011062 001016      BNE     3$ ;BR IF MANUAL ENTRY
1101 011064 005237 001520      INC     CHANL ;UPDATE CHANNEL NUMBER
1102 011070 022737 000003 001520  CMP      #3,CHANL ;CHANNEL 3 (RAMP CHANNEL)?
1103 011076 001404      BEQ     1$ ;:YES
1104 011100 022737 000007 001520  CMP      #7,CHANL ;CHANNEL 7 (EDC INPUT CHANNEL)?
1105 011106 001002      BNE     2$ ;:NO
1106 011110 005237 001520      1$: INC     CHANL ;CHANNELS 3 AND 7 ARE SKIPPED
1107 011114 000137 010460      2$: JMP     NOITS1 ;NO, CONTINUE TESTING
1108 011120 000207      3$: RTS     PC ;EXIT
1109 011122 000000      (CHANIS: 0 ;CURRENT CHANNEL TYPE

```

1111
1112 011124
1113
(3)
(3)
(2) 011124 000004
(1) 011126 012737 000001 001160
1114 011134 104401 027670
1115 011140 004737 042342
1116 011144 013746 001564
1117 011150 104403
1118 011152 001 000
1119 011154 104401 001165
1120 011160 012737 000001 001506
1121 011166 012737 000002 001510
1122 011174 004737 011230
1123 011200 005737 016540
1124 011204 001410
1125 011206 012737 000024 001506
1126 011214 012737 000025 001510
1127 011222 004737 011230
1128 011226
(2) 011226 000461
1129
1130
1131 011230 005037 011370
1132 011234 005237 011370
1133 011240 022737 000006 011370
1134 011246 001444
1135 011250 013737 001510 001520
1136 011256 004537 025710
1137 011262 013737 001502 001536
1138 011270 005002
1139 011272 004737 023352
1140 011276 000756
1141 011300 004737 023352
1142 011304 000753
1143 011306 005702
1144 011310 100001
1145 011312 005402
1146 011314 010204
1147 011316 012737 000001 023500
1148 011324 004737 023220
1149 011330 023737 001510 001506
1150 011336 103413
1151 011340 013702 001506
1152 011344 013737 001510 001506
1153 011352 010237 001510
1154 011356 000724
1155 011360 012702 000377
1156 011364 000753
1157 011366 000207
1158 011370 000000

NOI JMP:
:*****
:TEST 57 INTERCHANNEL SETTLING TEST, 1 EDGE
:*****
TST57: SCOPE
MOV #1,\$TIMES ;DO 1 ITERATION
TYPE ,SETMSG ;TYPE 'SETTLING TEST'
JSR PC,WHICHU ;DETERMINE THE UNIT #
MOV UNITBD,-(SP) ;SAVE IT
TYPOS ;TYPE IT
.BYTE 1,0
TYPE ,\$CRLF
MOV #1,CH1 ;LOAD INITIAL CHANNEL NUMBER
MOV #2,CH2 ;
JSR PC,SETTLE ;RUN TEST ON CH 1-2
TST WFAM ;RUNNING MNCAM ON TESTER ?
BEQ 1\$;BR IF NOT
MOV #24,CH1 ;GET MUX CHANNEL INCASE TESTING MNCAM
MOV #25,CH2 ;GET NEXT CHANNEL
JSR PC,SETTLE ;RUN TEST ON MNCAM CH 24-25
1\$: BR TST60 ;NEXT TEST
:SUBROUTINE TO DO THE SETTLING BETWEEN TWO CHANNELS
SETTLE: CLR 20\$;CLEAR RETRY COUNT
1\$: INC 20\$;INCREMENT COUNT
CMP #6,20\$;IS COUNT = 6?
BEQ 3\$;YES
MOV CH2,CHANL ;GET EDGE VALUES
JSR R5,CONVTC ;SET UP EDGE VALUE
MOV TEMP,EDGE
CLR R2
JSR PC,SET1A ;SCALING = .02 LSB
BR 1\$;ERROR RECOVERY JUMP
JSR PC,SET1A ;MAKE IT .01 LSB
BR 1\$;ERROR RECOVERY JUMP
TST R2 ;TEST RESULTS
BPL 2\$;MAKE IT POSITIVE
MOV R2,R4
MOV #1,EDGFLG
JSR PC,TYPSET ;TYPE SETTLING INFORMATION
CMP CH2,CH1 ;DONE?
BLO 4\$;YES
MOV CH1,R2 ;SETTLE THE OTHER WAY
BR SETTLE ;
3\$: MOV #255.,R2 ;SET SETTLING TO MAX ERROR
BR 2\$;
4\$: RTS PC ;EXIT
20\$: 0

1160
1161
(3)
(3)
(2) 011372 000004
(1) 011374 012737 000001 001160
1162 011402 105727 044537
1163 011406 100010
1164 011410 005737 001176
1165 011414 001405
1166 011416 012737 000003 017460
1167 011424 004737 023740
1168
1169
(3)
(3)
(2) 011430 000004
(1) 011432 012737 000001 001160
1170 011440 000207
1171
1172
1173 011442 116037 044534 011472
1174 011450 042737 177600 011472
1175 011456 122737 000003 011472
1176 011464 001001
1177 011466 000261
1178 011470 000207
1179 011472 000000
1180
1181 011474 010146
1182 011476 010246
1183 011500 013702 001424
1184 011504 012701 000010
1185 011510 112712 000077
1186 011514 112712 000001
1187 011520 110112
1188 011522 005201
1189 011524 022701 000100
1190 011530 001367
1191 011532 012602
1192 011534 012601
1193 011536 000207
1194
1195
1196 011540 013777 001124 167654
1197 011546 017737 167650 001126
1198 011554 023737 001124 001126
1199 011562 001002
1200 011564 062716 000002
1201 011570 000002

```
*****  
*TEST 60 DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST (CHANNEL 3 ONLY AFTER  
*****  
TST60: SCOPE  
MOV #1,$TIMES ;:DO 1 ITERATION  
TSTB #CHTABL+3 ;:TESTING CHANNEL 3?  
BPL TST61 ;:BR IF NOT  
TST $PASS ;:FIRST TIME-SKIP DIFLIN  
BEQ TST61 ;:BR IF FIRST PASS  
MOV #3,CHA ;:LOAD CHANNEL TO RUN ON  
JSR PC,DIFLIN ;:RUN DIF LIN AND REL ACC ON CH 3  
  
*****  
*TEST 61 END OF WRAPAROUND ANALOG TESTS  
*****  
TST61: SCOPE  
MOV #1,$TIMES ;:DO 1 ITERATION  
RTS PC ;:RETURN TO TEST SECTION  
  
;SUBROUTINE TO CHECK IF CHANNEL IN RO IS AN 'AG' CHANNEL  
CHKAGC: MOVB CHTABL(RO),10$ ;:GET CHANNEL TYPE  
BIC #177600,10$ ;:CLEAR OFF BITS  
CMPB #3,10$ ;:TEST IF MNCAG CHANNEL  
BNE 1$ ;:BR IF NOT  
SEC ;:SET 'CARRY' BIT  
1$: RTS PC ;:EXIT  
10$: 0  
  
;SUBROUTINE TO LOAD A GAIN OF '01' INTO EACH CHANNEL 10-77  
LD01CH: MOV R1,-(SP)  
MOV R2,-(SP)  
MOV ADST1,R2 ;:LOAD ADDRESS POINTER  
MOV #10,R1 ;:LOAD INITIAL CHANNEL  
1$: MOVB #77,(R2) ;:LOAD 'ESCAPE'  
MOVB #1,(R2) ;:LOAD GAIN = 01  
MOVB R1,(R2) ;:LOAD CHANNEL #  
INC R1 ;:UPDATE CHANNEL #  
CMP #100,R1 ;:TEST IF LAST CHANNEL  
BNE 1$ ;:BR IF NOT LAST CHANNEL  
MOV (SP)+,R2  
MOV (SP)+,R1  
RTS PC ;:EXIT  
  
;SUBROUTINE FOR LOGIC TESTS  
TESTIT: MOV $GDDAT,@STREG ;:LOAD EXPECTED DATA INTO REGISTER  
TEST: MOV @STREG,$BDDAT ;:READ ACTUAL REGISTER  
CMP $GDDAT,$BDDAT ;:COMPARE RESULTS  
BNE RETERR ;:RETURN EXIT  
ADD #2,(SP) ;:CORRECT EXIT BUMPS ENTRY BY 2  
RETERR: RTI ;:EXIT
```

```

1203 ;SUBROUTINE TO DO THE LOADING AND READING OF GAIN INFO
1204 ; 1ST. LOAD CHANNEL 0-77 WITH GAIN = 01
1205 ; 2ND. WRITE CHANNEL X GAIN TO = 10
1206 ; 3RD. READ CHANNEL X GAIN AND EXPECT = 10
1207 ; 4TH. READ CHANNEL 0-77 EXCEPT CH XX AND NON-PREAMP CHS.
1208
1209 ;DO 1ST STEP
1210 011572 004737 011474 CHKGAN: JSR PC,LD01CH ;LOAD GAIN BITS TO 01
1211 011576 012737 011604 001110 MOV #1$,SLPERR ;LOAD ERROR RETURN ADDRESS
1212
1213 ;DO 2ND STEP
1214 011604 112777 000077 167612 1$: MOVB #77,@ADST1 ;LOAD 'ESC'
1215 011612 112777 000002 167604 MOVB #2,@ADST1 ;LOAD GAIN = 10
1216 011620 110077 167600 MOVB R0,@ADST1 ;LOAD CHANNEL XX
1217
1218 ;DO 3RD STEP
1219 011624 004737 011744 JSR PC,RDCHXY ;READ CHANNEL IN R0
1220 011630 012737 020000 001124 MOV #20000,$GDDAT ;LOAD EXPECTED
1221 011636 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE TO EXPECTED
1222 011644 001403 BEQ 2$ ;BR IF SAME
1223 011646 010037 001520 MOV R0,CHANL ;SAVE CHANNEL INFO
1224 011652 104012 ERROR 12 ;GAIN ON CHANNEL FAILED TO LOAD
1225
1226 011654 012700 000010 ;NOW DO 4TH STEP
1227 011660 012737 011674 001110 2$: MOV #10,R0 ;PRIME THE CHANNEL #
1228 011666 012737 010000 001124 MOV #3$,SLPERR ;LOAD ERROR RETURN ADDRESS
1229 011674 020037 014664 3$: MOV #10000,$GDDAT ;LOAD EXPECTED VALUE
1230 011700 001414 BEQ 4$ ;TEST IF R0 = CHXX
1231 ;TEST IF R0 CHANNEL IS AN 'AG' CHANNEL
1232 011702 004737 011442 JSR PC,CHKAGC ;BR IF NOT 'AG' CHANNEL
1233 011706 103011 BCC 4$ ;READ CHANNEL IN R0 STATUS
1234 011710 004737 011744 JSR PC,RDCHXY ;COMPARE
1235 011714 023737 001124 001126 CMP $GDDAT,$BDDAT ;BR IF SAME
1236 011722 001403 BEQ 4$ ;SAVE BAD CHANNEL INFO
1237 011724 010037 001520 MOV R0,CHANL ;CHANNEL GAIN BITS CHANGED IN ERROR
1238 011730 104012 ERROR 12 ;UPDATE CHANNEL
1239 011732 005200 4$: INC R0 ;TEST IF MORE CHANNELS
1240 011734 022700 000100 CMP #100,R0 ;BR IF NONE
1241 011740 001355 BNE 3$ ;EXIT
1242 011742 000207 RTS PC
1243
1244 ;SUBROUTINE TO CONVERT CHANNEL IN R0
1245 ;RETURN STATUS IN $BDDAT
1246 011744 110077 167454 RDCHXY: MOV R0,@ADST1 ;LOAD MUX REG.
1247 011750 052777 000010 167444 BIS #BIT3,@STREG ;ENABLE STATUS INFO.
1248 011756 105277 167440 INCB @STREG ;START CONVERSION
1249 011762 105777 167434 1$: TSTB @STREG ;WAIT FOR DONE
1250 011766 100375 BPL 1$
1251 011770 017737 167432 001126 MOV @ADBUFF,$BDDAT ;READ STATUS
1252 011776 042737 147777 001126 BIC #147777,$BDDAT ;MASK OFF A/D CONVERSION DATA
1253 012004 000207 RTS PC ;EXIT
1254

```

```

1256
1257
1258 012006 112537 012100
1259 012012 112537 012120
1260 012016 012537 012144
1261 012022 013537 012222
1262 012026 012737 012040 012330
1263 012034 005037 012226
1264 012040 005237 012226
1265 012044 022737 000006 012226
1266 012052 001450
1267 012054 013737 001520 001516
1268 012062 004537 025710
1269 012066 013737 001502 001536
1270 012074 004537 023540
1271 012100 000020
1272 012102 004737 012304
1273 012106 013737 001532 012224
1274 012114 004537 023540
1275 012120 000124
1276 012122 004737 012304
1277 012126 163737 001532 012224
1278 012134 012737 000001 023500
1279 012142 104401
1280 012144 034066
1281 012146 013702 012224
1282 012152 004737 025644
1283 012156 023737 012224 012222
1284 012164 003007
1285 012166 104401 034135
1286 012172 000412
1287 012174 012737 000377 012224
1288 012202 000757
1289 012204 104401 034611
1290 012210 004737 042334
1291 012214 005237 001112
1292 012220 000205
1293 012222 000000
1294 012224 000000
1295 012226 000000

;SUBROUTINE TO DO THE RMS AND PEAK NOISE TESTING
RMSPEK: MOV (R5)+,60$ ;GET 1 POINT
        MOV (R5)+,61$ ;GET 2 POINT
        MOV (R5)+,62$ ;GET TEXT POINTER
        MOV @ (R5)+,63$ ;GET TOLERANCE
        MOV #1$,ERRADR ;SET UP ERROR RETRY ADDRESS
        CLR 65$ ;CLEAR RETRY COUNT
1$: INC 65$ ;INCREMENT COUNT
   CMP #6,65$ ;IS COUNT = 6?
   BEQ 3$ ;:YES, CHANNEL TOO WIDE OR NOISY
   MOV CHANL,DUMMY ;LOAD DUMMY CHANNEL
   JSR R5,CONVTC ;GET EDGE VALUE
   MOV TEMP,EDGE ;SET UP EDGE VALUE
   JSR R5,SAR SUB ;DO SAR ROUTINE AT 16%
60$: 16.
   JSR PC,TSTDAC ;CHECK VERNIER DAC SETTING
   MOV DAC,64$ ;ADD RESULT TO RMS
   JSR R5,SAR SUB ;DO SAR ROUTINE AT 84%
E1$: 84.
   JSR PC,TSTDAC ;CHECK VERNIER DAC SETTING
   SUB DAC,64$ ;SUBTRACT RESULT FROM RMS
   MOV #1,EDGFLG
2$: TYPE
62$: RMSNOI ;TEXT POINTER
   MOV 64$,R2
   JSR PC,TYPRP ;TYPE RMS VALUES
   CMP 64$,63$ ;WITHIN LIMITS?
   BGT 4$ ;NO
   TYPE ,OKMSG
   BR 5$
3$: MOV #255.,64$ ;SET RMS TO MAX ERROR
   BR 2$
4$: TYPE ,ERMSG
   JSR PC,WHICHV ;INDICATE BAD UNIT
   INC $ERTTL ;UPDATE ERROR TOTAL
5$: RTS R5 ;EXIT
63$: 0
64$: 0
65$: 0

```

```

1297
1298
1299 ;SUBROUTINE TO FIND THE 50-50 EDGE OF THE INPUT SIGANL
1300
1301 012230 012737 004001 001536 OFFSET: MOV #4001,EDGE ;4000,4001 EDGE
1302 012236 004537 023540 JSR R5,SARSUB
1303 012242 000062 50.
1304 012244 013737 001532 001502 MOV DAC,TEMP
1305 012252 012737 004000 001536 MOV #4000,EDGE ;3777,4000 EDGE
1306 012260 004537 023540 JSR R5,SARSUB
1307 012264 000062 50.
1308 012266 063737 001532 001502 ADD DAC,TEMP
1309 012274 162737 000400 001502 SUB #400,TEMP
1310 012302 000207 RTS PC
1311
1312
1313 ; ROUTINE TO TEST DAC SETTING FROM SARSUB
1314 ; JUMPS TO ADDRESS IN ERRADR IF DAC SETTING IS EITHER 0 OR 377
1315 ; OTHERWISE RETURNS TO CALL+1
1316 012304 005737 001532 TSTDAC: TST DAC ;IS DAC = 0 ?
1317 012310 001405 BEQ 1$ ;;YES
1318 012312 022737 000377 001532 CMP #377,DAC ;IS DAC = 377 ?
1319 012320 001401 BEQ 1$ ;;YES
1320 012322 000207 RTS PC
1321 012324 005726 1$: TST (SP)+ ;POP CALL OFF STACK
1322 012326 000137 JMP @PC)+ ;JUMP TO ADDRESS IN ERRADR
1323 012330 000000 ERRADR: 0
1324
1325 ;SUBROUTINE TO HANDLE THE SINGLE ENDED-DIFFERENTIAL LOGIC TESTS
1326 012332 012537 012446 TSTSDF: MOV (R5)+,10$ ;GET 1ST ARGUMENT
1327 012336 005737 001544 TST WFTST ;USING THE TESTER ?
1328 012342 001414 BEQ 1$ ;BR IF NOT
1329 012344 005737 012446 TST 10$ ;TEST THE 1ST ARG.
1330 012350 001004 BNE 23$ ;BR IF NON ZERO
1331 012352 042777 000200 167106 BIC #BIT7,@DRVDOR ;CLEAR THE BIT
1332 012360 000403 BR 24$
1333 012362 052777 000200 167076 23$: BIS #BIT7,@DRVDOR ;SET THE BIT
1334 012370 004737 016552 24$: JSR PC,STALL ;ALLOW RELAY TO CHANGE
1335 012374 012537 001124 1$: MOV (R5)+,$GDDAT ;GET 2ND ARG. <EXPECTED DATA>
1336 012400 012577 167016 MOV (R5)+,@STREG ;GET 3RD ARG. <CHANNEL TO USE>
1337 012404 105277 167012 INCB @STREG ;START CONVERSION
1338 012410 105777 167006 2$: TSTB @STREG ;WAIT FOR DONE
1339 012414 100375 BPL 2$ ;
1340 012416 017737 167004 001126 MOV @ADBUFF,$BDDAT ;READ RESULT
1341 012424 042737 157777 001126 BIC #157777,$BDDAT ;MASK OFF OTHER BITS
1342 012432 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1343 012440 001401 BEQ 3$ ;;BR IF SAME
1344 012442 104001 ERROR 1 ;INCORRECT VALUE TO SINGLE ENDED-DIFFERENTIAL MODE
1345 012444 000205 3$: RTS R5 ;EXIT
1346 012446 000000 10$: 0

```

1348
 1349
 1350 012450 013702 001502
 1351 012454 100402
 1352 012456 104401 034607
 1353 012462 104416
 1354 012464 104401 035742
 1355 012470 000207
 1356
 1357
 1358
 1359 012472 005303
 1360 012474 001005
 1361 012476 012703 000005
 1362 012502 104401 001165
 1363 012506 000402
 1364 012510 104401 034017
 1365 012514 005037 001534
 1366 012520 005077 166420
 1367 012524 105777 166414
 1368 012530 100404
 1369 012532 005237 001534
 1370 012536 001372
 1371 012540 000416
 1372 012542 005777 166400
 1373 012546 012777 000100 166370
 1374 012554 004537 026032
 1375 012560 000000
 1376 012562 026672
 1377 012564 000402
 1378 012566 062716 000002
 1379 012572 062716 000002
 1380 012576 000207

```

;SUBROUTINE TO INSERT '+' AND TYPE # ON THE STACK
TOFF:  MOV    TEMP,R2
      BMI    1$      ;; IS THE NUMBER POSITIVE?
      TYPE  ,POSITV
1$:    TYPDC
      TYPE  ,MLSB    ;TYPE ASCIZ STRING
      RTS   PC

;SUBROUTINE TO WAIT FOR OPERATOR'S 'RETURN' THEN CHECK TOLERANCES
TCHK:  DEC    R3      ;DECREMENT COUNT
      BNE   1$      ;;
      MOV   #5,R3    ;RESET COUNT
      TYPE ,SCLF    ;TYPE A CARRIAGE RETURN AND LINE FEED
      BR   2$      ;;
1$:    TYPE  ,SPACE  ;TYPE FOUR (4) SPACES
2$:    CLR   DELAY   ;CLEAR DELAY
      CLR  @STKS    ;CLEAR INTERRUPT ENABLE
3$:    TSTB  @STKS   ;IS KEYBOARD FLAG SET?
      BMI  4$      ;; YES
      INC  DELAY   ;IS DELAY ZERO?
      BNE  3$      ;; NO
      BR  5$      ;;
4$:    TST  @STKB   ;CLEAR FLAG
      MOV  #100,@STKS ;SET INTERRUPT ENABLE
      JSR R5,COMPARE ;TEST LAST CONVERSION
      OR  0
      V10
      BR   5$      ;TOLERANCE .10 LSB
      ADD #2,(SP)  ;BUMP RETURN ADDRESS
5$:    ADD  #2,(SP)  ;BUMP RETURN ADDRESS 2 WORDS
6$:    RTS   PC
  
```



```

1382 .SBTTL MNCAD CALIBRATION SECTION
1383 012600 104401 034146 BEGINC: TYPE ,CCHAN ;ASK FOR CHANNEL
1384 012604 104413 RDOCT ;READ CHANNEL NUMBER
1385 012606 012637 001520 MOV (SP)+,CHANL ;STORE CHANNEL NUMBER
1386 012612 013737 001520 001516 MOV CHANL,DUMMY ;LOAD DUMMY
1387 012620 104401 034234 1$: TYPE ,SEL ;SELECT OFFSET OR GAIN ADJUST
1388 012624 104412 RDLIN ;GET TEST
1389 012626 012600 MOV (SP)+,R0 ;MOVE POINTER TO R0
1390 012630 121027 000117 CMPB (R0),#0 ;IS IT '0'?
1391 012634 001406 BEQ AJOFF ;:YES, GO TO ADJUST OFFSET
1392 012636 121027 000107 CMPB (R0),#G ;IS IT 'G'?
1393 012642 001430 BEQ AJGAIN ;:YES, GO TO ADJUST GAIN
1394 012644 104401 001164 TYPE ,SQUES ;TYPE '?'
1395 012650 000763 BR 1$ ;
1396
1397 ;SUBROUTINE TO CHECK OFFSET ADJUSTMENT VALUES
1398 012652 104401 034427 AJOFF: TYPE ,IGND ;GROUND CHANNEL
1399 012656 104412 RDLIN ;WAIT FOR CR
1400 012660 005726 TST (SP)+ ;POP 1 WORD OFF STACK
1401 012662 104401 034325 1$: TYPE ,XADJ ;ADJUST MESSAGE
1402 012666 012703 000005 MOV #5,R3 ;SET UP COUNT
1403 012672 004737 012230 2$: JSR PC,OFFSET ;TEST AND TYPE OFFSET ERROR
1404 012676 004737 012450 JSR PC,TOFF ;TYPE OFFSET
1405 012702 004737 012472 JSR PC,TCHK ;CHECK FOR A CHARACTER AND DELAY
1406 012706 000771 BR 2$ ;
1407 012710 000402 BR 3$ ;:NOT WITHIN TOLLERANCE, TRY AGAIN
1408 012712 000137 001634 JMP BEG2
1409 012716 104401 034611 3$: TYPE ,ERMSG ;TELL OPER. 'ERROR'
1410 012722 000757 BR 1$
1411 ;SUBROUTINE TO CHECK THE GAIN ADJUSTMENT
1412 012724 104401 034526 AJGAIN: TYPE ,IVOLT ;INPUT +5.115 VOLTS ON CHANNEL
1413 012730 104401 034466 TYPE ,CRWR
1414 012734 104412 RDLIN ;WAIT FOR CR
1415 012736 005726 TST (SP)+ ;POP 1 WORD OFF STACK
1416 012740 104401 034572 1$: TYPE ,YADJ ;ADJUST MESSAGE
1417 012744 104401 034341 TYPE ,M0LSB ;TYPE '' FOR 0.00 LSB ERROR''
1418 012750 012703 000005 MOV #5,R3 ;SET UP COUNT
1419 012754 012737 007777 001536 2$: MOV #7777,EDGE ;LOOK FOR 7776,7777 EDGE
1420 012762 004537 023540 JSR R5,SARSUB
1421 012766 000062 50.
1422 012770 013737 001532 001502 MOV DAC,TEMP ;SAVE DAC
1423 012776 012737 007776 001536 MOV #7776,EDGE ;LOOK FOR 7775,7776 EDGE
1424 013004 004537 023540 JSR R,SARSUB
1425 013010 000062 50.
1426 013012 063737 001532 001502 ADD DA,TEMP ;ADD RESULTS
1427 013020 162737 000400 001502 SUB #400,TEMP ;OFFSET RESULT
1428 013026 004737 012450 JSR PC,TOFF ;TYPE GAIN
1429 013032 004737 012472 JSR PC,TCHK ;CHECK FOR CHARACTER AND DELAY
1430 013036 000746 BR 2$ ;
1431 013040 000402 BR 3$ ;:NOT WITHIN TOLLERANCE, TRY AGAIN
1432 013042 000137 001634 JMP BEG2
1433 013046 104401 034611 3$: TYPE ,ERMSG ;TELL OPER. 'ERROR'
1434 013052 000732 BR 1$

```

```

1436
1437 013054 004737 023024
1438 013060 104401 034146
1439 013064 104413
1440 013066 012600
1441 013070 010037 001520
1442 013074 000300
1443 013076 052700 000010
1444 013102 010077 166314
1445 013106 104401 032454
1446 013112 104401 033155
1447 013116 012737 040000 001124
1448 013124 104417
1449 013126 104011
1450 013130 104401 033176
1451 013134 012737 050000 001124
1452 013142 104417
1453 013144 104011
1454 013146 104401 033221
1455 013152 012737 060000 001124
1456 013160 104417
1457 013162 104011
1458 013164 104401 033244
1459 013170 012737 070000 001124
1460 013176 104417
1461 013200 104011
1462 013202 104401 033155
1463 013206 104401 032525
1464 013212 012737 100000 001124
1465 013220 104417
1466 013222 104011
1467 013224 104401 032575
1468 013230 012737 140000 001124
1469 013236 104417
1470 013240 104011
1471 013242 104401 001165
1472 013246 104401 032015
1473 013252 104401 034466
1474 013256 104412
1475 013260 005726
1476 013262 104401 033740
1477 013266 000137 001634
1478
1479 013272 104401 034466
1480 013276 104412
1481 013300 005726
1482 013302 005277 166114
1483 013306 105777 166110
1484 013312 100375
1485 013314 017737 166106 001126
1486 013322 042737 007777 001126
1487 013330 023737 001124 001126
1488 013336 001002
1489 013340 062716 000002
1490 013344 000002

      .SBTTL SWITCH GAIN MANUAL INTERVENTION TEST
BEGINF: JSR PC, FIXONE ;ENSURE INITIAL BUS ADDRESS OF UNIT
        TYPE ,CCHAN ;ASK FOR CHANNEL
        RDOCT ;READ CHANNEL NUMBER
        MOV (SP)+, R0 ;GET CHANNEL NUMBER
        MOV R0, CHANL ;LOAD CHANNEL FOR ERROR REPORT
        SWAB R0 ;PUT CHANNEL NUMBER IN HIGH BYTE
        BIS #BIT3, R0 ;SET STATUS ENABLE BIT
        MOV R0, @STREG ;LOAD CHANNEL AND STATUS ENABLE
        TYPE ,SCM ;ASK MODE BE SET TO CURRENT
        TYPE ,GHLF ;ASK GAIN BE SET TO .5
        MOV #BIT14, $GDDAT ;SET UP EXPECTED
        TESTID ;GO TEST FOR ID CODE
        ERROR 11
        TYPE ,GAIN5 ;ASK GAIN BE SET TO 5
        MOV #BIT14!BIT12, $GDDAT ;LOAD EXPECTED
        TESTID ;GO TEST ID CODE
        ERROR 11
        TYPE ,GAIN50 ;ASK GAIN BE SET TO 50
        MOV #BIT14!BIT13, $GDDAT ;LOAD EXPECTED
        TESTID ;GO TEST ID CODE
        ERROR 11
        TYPE ,GAIN5M ;ASK GAIN BE SET TO 500
        MOV #BIT14!BIT13!BIT12, $GDDAT ;LOAD EXPECTED
        TESTID ;GO TEST ID CODE
        ERROR 11
        TYPE ,GHLF ;SET RANGE SWITCH
        TYPE ,SRM ;ASK MODE BE SET TO RESISTANCE
        MOV #100000, $GDDAT ;LOAD EXPECTED VALUE
        TESTID
        ERROR 11 ;RESISTANCE MODE SWITCH VALUE IN ERROR
        TYPE ,SVM ;ASK MODE BE SET TO VOLTS
        MOV #140000, $GDDAT ;LOAD EXPECTED VALUE
        TESTID
        ERROR 11 ;VOLTAGE MODE SWITCH VALUE IN ERROR
        TYPE ,$CRLF
        TYPE ,SAGTST ;TELL OPER. TO SET SWITCHES
        TYPE ,CRWR
        RDLIN
        TST (SP)+ ;POP RETURN OFF STACK
        TYPE ,ENDTST ;TELL OPER 'THATS ALL FOLKS'
        JMP BEG2

TPRMP: TYPE ,CRWR ;ASK FOR CR WHEN READY
        RDLIN ;WAIT FOR CR
        TST (SP)+ ;POP 1 WORD OFF STACK
        INC @STREG ;START A CONVERSION
        1$: TSTB @STREG ;WAIT TILL DONE
        BPL 1$
        MOV @ADDBUFF, $BDDAT ;GET RESULTS
        BIC #7777, $BDDAT ;CLEAR CONVERTED VALUE
        CMP $GDDAT, $BDDAT ;IS ID RIGHT?
        BNE 2$ ;NO, TAKE ERROR RETURN
        ADD #2, (SP) ;BUMP RETURN ADDRESS
        2$: RTI
  
```

```

1492 .SBTTL MNCAG TEST MODULE INTERACTIVE TESTS
1493 013346 004737 023024 BEGINT: JSR PC, FIXONE ;ENSURE CORRECT ADDRESSES
1494 013352 104401 001165 TYPE ,SCLF
1495 013356 104401 032015 TYPE ,SAGTST ;TELL OPER. TO SET AG TO 'P'
1496 013362 104401 034146 TYPE ,CCHAN ;GET CHANNEL NUMBER
1497 013366 104413 RDOCT
1498 013370 012637 001506 MOV (SP)+, CH1 ;GET CHANNEL # FROM OPER.
1499 013374 004737 014014 JSR PC, CLRCHS ;CONVERT EACH CHANNEL OF THIS MNCAG
1500 ;FIRST - TEST MNCAG-TA HOLD LOGIC FOR THESE CHANNELS
1501 013400 004537 014246 JSR R5, TSTHLD ;TEST HOLD FOR 1ST CHANNEL OF THIS AG
1502 013404 000 005 .BYTE 0,5 ;CHANNEL OFFSET, SWITCH NUMBER TO PUSH
1503 013406 004537 014246 JSR R5, TSTHLD ;
1504 013412 001 006 .BYTE 1,6 ;
1505 013414 004537 014246 JSR R5, TSTHLD ;
1506 013420 002 007 .BYTE 2,7 ;
1507 013422 004537 014246 JSR R5, TSTHLD ;
1508 013426 003 010 .BYTE 3,8 ;
1509
1510 ;MNCAG PART 1
1511 013430 004537 013610 JSR R5, TSETUP ;GO DO THE WORK
1512 013434 002 003 002 .BYTE 2,3,2,3 ;FRONT PANEL EXPECTED CODE
1513 013440 032761 .WORD TXTP2 ;POS. OF TEST MODULE SWITCH
1514 013442 000 002 .BYTE 0,2 ;GAIN, SPREAD
1515 013444 004002 .WORD 4002 ;CHANNEL A - C EXPECTED VALUE
1516 013446 001 002 .BYTE 1,2 ;GAIN, SPREAD
1517 013450 004024 .WORD 4024 ;CHANNEL B - D EXPECTED VALUE
1518 013452 002 004 .BYTE 2,4 ;GAIN, SPREAD
1519 013454 004310 .WORD 4310 ;CHANNEL A - C EXPECTED VALUE
1520 013456 003 050 .BYTE 3,50 ;GAIN, SPREAD
1521 013460 007720 .WORD 7720 ;CHANNEL B - D EXPECTED VALUE
1522
1523 ;MNCAG PART 2
1524 013462 004537 013610 JSR R5, TSETUP ;GO DO THE WORK
1525 013466 003 002 003 .BYTE 3,2,3,2 ;FRONT PANEL EXPECTED CODE
1526 013472 000000 .WORD 0 ;NO TEST MODULE CHANGES
1527 013474 000 002 .BYTE 0,2 ;GAIN, SPREAD
1528 013476 004002 .WORD 4002 ;CHANNEL A - C EXPECTED VALUE
1529 013500 001 002 .BYTE 1,2 ;GAIN, SPREAD
1530 013502 004024 .WORD 4024 ;CHANNEL B - D EXPECTED VALUE
1531 013504 002 004 .BYTE 2,4 ;GAIN, SPREAD
1532 013506 004310 .WORD 4310 ;CHANNEL A - C EXPECTED VALUE
1533 013510 003 050 .BYTE 3,50 ;GAIN, SPREAD
1534 013512 007720 .WORD 7720 ;CHANNEL B - D EXPECTED VALUE
1535
1536 ;MNCAG PART 3
1537 013514 004537 013610 JSR R5, TSETUP ;GO DO THE WORK
1538 013520 001 002 001 .BYTE 1,2,1,2 ;FRONT PANEL EXPECTED CODE
1539 013524 033057 .WORD TXTP3 ;TEST MODULE SWITCH POS.
1540 013526 000 002 .BYTE 0,2 ;GAIN, SPREAD
1541 013530 004024 .WORD 4024 ;CHANNEL A - C EXPECTED VALUE
1542 013532 001 006 .BYTE 1,6 ;GAIN, SPREAD
1543 013534 004310 .WORD 4310 ;CHANNEL B - D EXPECTED VALUE
1544 013536 002 053 .BYTE 2,53 ;GAIN SPREAD

```

```

1545 013540 007720          .WORD 7720
1546 013542      000      000          .BYTE 0,0          ;NULL
1547 013544 000000          .WORD 0          ;NULL CHANNEL B - D
1548
1549          ;MNCAG PART 4
1550 013546 004537 013610          JSR R5,TSETUP          ;GO DO THE WORK
1551 013552      002      001      002          .BYTE 2,1,2,1          ;FRONT PANEL EXPECTED CODE
1552 013555      001
1552 013556 000000          .WORD 0          ;NO TEST MODULE CHANGES
1553 013560      000      002          .BYTE 0,2          ;GAIN, SPREAD
1554 013562 004024          .WORD 4024          ;CHANNEL A - C EXPECTED VALUE
1555 013564      001      006          .BYTE 1,6          ;GAIN, SPREAD
1556 013566 004310          .WORD 4310          ;CHANNEL B - D EXPECTED VALUE
1557 013570      002      053          .BYTE 2,53          ;GAIN, SPREAD
1558 013572 007720          .WORD 7720          ;CHANNEL A - C EXPECTED VALUE
1559 013574      000      000          .BYTE 0,0          ;NULL
1560 013576 000000          .WORD 0          ;CHANNEL B - D NULL
1561
1562 013600 104401 033740          TYPE ,ENDTST          ;TELL OPERATOR IT'S DONE
1563 013604 000137 001634          JMP BEG2          ;EXIT
1564
1565          ;SUBROUTINE TO DO MOST OF THE WORD FOR BEGINT
1566 013610 112500          TSETUP: MOVB (R5)+,R0          ;GET 1ST ARG.
1567 013612 104401 032645          TYPE ,CHAPOS          ;TELL OPER 'A' CHANNEL
1568 013616 004737 014364          JSR PC,TYPITA          ;CONVERT AND TYPE IT
1569 013622 010037 014654          MOV R0,CHANA          ;SAVE CHANNEL 'A' EXPECTED VALUE
1570 013626 112500          MOVB (R5)+,R0          ;GET 2ND ARG.
1571 013630 104401 032670          TYPE ,CHBPOS          ;TELL OPER 'B' CHANNEL
1572 013634 004737 014364          JSR PC,TYPITA          ;CONVERT AND TYPE IT
1573 013640 010037 014656          MOV R0,CHANB          ;SAVE CHANNEL 'B' EXPECTED VALUE
1574 013644 112500          MOVB (R5)+,R0          ;GET 3RD ARG.
1575 013646 104401 032713          TYPE ,CHCPOS          ;TELL OPER 'C' CHANNEL
1576 013652 004737 014364          JSR PC,TYPITA          ;CONVERT AND TYPE IT
1577 013656 010037 014660          MOV R0,CHANC          ;SAVE CHANNEL 'C' EXPECTED VALUE
1578 013662 112500          MOVB (R5)+,R0          ;GET 4TH ARG.
1579 013664 104401 032736          TYPE ,CHDPOS          ;TELL OPER 'D' CHANNEL
1580 013670 004737 014364          JSR PC,TYPITA          ;CONVERT AND TYPE IT
1581 013674 010037 014662          MOV R0,CHAND          ;SAVE CHANNEL 'D' EXPECTED VALUE
1582          ;NOW TELL OPERATOR ABOUT MNCAG (PREAMP) TEST MODULE POSITIONS
1583 013700 012537 013710          MOV (R5)+,60$          ;GET 5TH ARG.
1584 013704 001402          BEQ 20$          ;BR IF NONE
1585 013706 104401          TYPE          ;TELL OPER
1586 013710 000000          60$: 0
1587          ;NOW TELL OPER. TO TYPE 'RETURN' KEY WHEN READY
1588 013712 104401 034466          20$: TYPE ,CRWR          ;WAIT FOR 'RETURN'
1589 013716 104412          RDLIN          ;WAIT FOR OPERATOR
1590 013720 005726          TST (SP)+          ;POP STACK

```

```
1592 :NOW CONVERT CHANNEL AND CHECK OPER SET CORRECT FRONT PANEL POS.  
1593 :IF FRONT PANEL SWITCH IS WRONG TELL THE OPERATOR  
1594 :IF OK, TEST THE VALUES  
1595 013722 013737 001506 001510      MOV    CH1,CH2      ;REPRIME THE CHANNEL VALUE  
1596 013730 004537 014426              JSR    R5,CONTA1    ;CONVERT AND CHECK CHANNEL 'A' FRONT PANEL SWITCH  
1597 013734 014654                      CHANA  
1598 013736 005237 001510              INC    CH2          ;DO NEXT CHANNEL  
1599 013742 004537 014426              JSR    R5,CONTA1    ;CONVERT AND CHECK CHANNEL 'B'  
1600 013746 014656                      CHANB  
1601 013750 005237 001510              INC    CH2          ;DO NEXT CHANNEL  
1602 013754 004537 014426              JSR    R5,CONTA1    ;CONVERT AND CHECK CHANNEL 'C'  
1603 013760 014660                      CHANC  
1604 013762 005237 001510              INC    CH2          ;DO NEXT CHANNEL  
1605 013766 004537 014426              JSR    R5,CONTA1    ;CONVERT AND CHECK CHANNEL 'D'  
1606 013772 014662                      CHAND  
1607 013774 004737 014072              JSR    PC,TSRT1     ;CONVERT CHANNELS AND VERIFY DATA  
1608 014000 004737 014072              JSR    PC,TSRT1     ;SECOND SECTION  
1609 014004 000205                      RTS     R5          ;EXIT  
1610 :SUBROUTINE TO DO A CONVERSION ON EACH MNCAG CHANNEL  
1611 014006 012737 000010 001506      CLRCHT: MOV    #10,CH1      ;LOAD 1ST CHANNEL #  
1612 014014 113777 001506 165402      CLRCHS: MOVB   CH1,@ADST1   ;SELECT CHANNEL  
1613 014022 004737 014050              JSR    PC,21$        ;CONVERT CHANNEL  
1616 014026 004737 014044              JSR    PC,20$        ;INCR. CHANN NUMBER AND CONVERT  
    (1) 014032 004737 014044              JSR    PC,20$        ;INCR. CHANN NUMBER AND CONVERT  
    (1) 014036 004737 014044              JSR    PC,20$        ;INCR. CHANN NUMBER AND CONVERT  
1617 014042 000207                      RTS     PC          ;EXIT  
1618 014044 105277 165354 20$:      INCB   @ADST1       ;UPDATE TO NEXT CHANNEL  
1619 014050 112777 000001 165344 21$:      MOVB   #1,@STREG     ;CONVERT CHANNEL  
1620 014056 105777 165340 22$:      TSTB   @STREG       ;WAIT FOR DONE  
1621 014062 100375                      BPL    22$          ;FALSE READ  
1622 014064 005777 165336              TST    @ADBUFF  
1623 014070 000207                      RTS     PC          ;EXIT
```

```

1625 ;SUBROUTINE TO SETUP FOR CONVERTING DIFFERENT CHANNELS
1626 014072 112537 014702 TSRT1: MOV (R5)+,PRIAC ;GET INITIAL GAIN FOR A/C
1627 014076 112537 014676 MOV (R5)+,SPRAC ;GET INITIAL SPREAD FOR A/C
1628 014102 012537 014654 MOV (R5)+,CHANA ;GET CHANNEL A/C EXPECTED VALUE
1629 014106 112537 014704 MOV (R5)+,PRIBD ;GET INITIAL GAIN FOR B/D
1630 014112 112537 014700 MOV (R5)+,SPRBD ;GET INITIAL SPREAD FOR B/D
1631 014116 012537 014656 MOV (R5)+,CHANB ;GET CHANNEL B/D EXPECTED VALUE
1632
1633 014122 013737 001506 014664 MOV CH1,CHXX ;PRIME THE CHANNEL VALUE
1634 014130 013737 014702 014666 MOV PRIAC,CHPRIM ;PRIME THE A/C GAIN VALUE
1635 014136 013737 014676 001530 MOV SPRAC,SPREAD ;PRIME THE SPREAD TOLERANCE
1636 014144 013737 014654 001124 MOV CHANA,$GDDAT ;PRIME THE EXPECTED VALUE
1637 014152 004737 014534 JSR PC,CON4T ;CONVERT CHANNEL AND TEST RESULT
1638
1639 014156 062737 000002 014664 ADD #2,CHXX ;UPDATE TO CHANNEL 'C'
1640 014164 004737 014534 JSR PC,CON4T ;CONVERT CHANNEL AND TEST RESULT
1641 ;NOW DO CHANNEL B/D
1642 014170 013737 014656 001124 MOV CHANB,$GDDAT ;TEST IF ANY CHANNEL 'B/D' EXPECTED VALUE
1643 014176 001422 BEQ 1$ ;BR IF NONE
1644 014200 013737 001506 014664 MOV CH1,CHXX ;PRIME INIT 'A' CHANNEL
1645 014206 005237 014664 INC CHXX ;MAKE IT 'CHANNEL B'
1646 014212 013737 014704 014666 MOV PRIBD,CHPRIM ;PRIME THE B/D GAIN VALUE
1647 014220 013737 014700 001530 MOV SPRBD,SPREAD ;PRIME THE SPREAD TOLERANCE
1648 014226 004737 014534 JSR PC,CON4T ;CONVERT CHANNEL 'B'
1649
1650 014232 062737 000002 014664 ADD #2,CHXX ;UPDATE TO CHANNEL 'D'
1651 014240 004737 014534 JSR PC,CON4T ;CONVERT CHANNEL AND TEST RESULT
1652 014244 000207 1$: RTS PC ;EXIT SUBROUTINE
1653 ;SUBROUTINE TO HANDLE THE MNCAG-TA HOLD TEST
1654 014246 112537 014362 TSTHLD: MOV (R5)+,10$ ;GET CHANNEL OFFSET FROM CH1
1655 014252 063737 001506 014362 ADD CH1,10$ ;ADD CH1 VALUE
1656 014260 113777 014362 165136 MOVB 10$,@ADST1 ;LOAD MUX TO ENSURE THE LED IS ON
1657 014266 104401 030735 TYPE ,LEDON ;TELL OPERATOR THE LED SHOULD BE ON
1658 014272 112537 031047 MOVB (R5)+,AGTASW ;LOAD WHICH SWITCH TO PUSH NOW
1659 014276 152737 000060 031047 BISB #60,AGTASW ;MAKE CHARACTER AN ASCII NUMBER
1660 014304 104401 031006 TYPE ,PUSHAG ;TELL OPERATOR TO PUSH SWITCH 5,6,7 OR 8
1661 014310 104401 034466 TYPE ,CRWR ;AND DEPRESS 'RETURN'
1662 014314 104412 RDLIN ;WAIT FOR OPERATOR
1663 014316 005726 TST (SP)+ ;CLEAN STACK
1664 014320 113777 014362 165076 MOVB 10$,@ADST1 ;LOAD MUX AGAIN, LED SHOULD GO OUT
1665 014326 104401 030761 TYPE ,LEDOFF ;TELL OPERATOR LED SHOULD BE OUT
1666 014332 104401 034466 TYPE ,CRWR ;AND DEPRESS 'RETURN'
1667 014336 104412 RDLIN ;WAIT FOR OPER.
1668 014340 005726 TST (SP)+ ;WAIT FOR OPER.
1669 014342 105277 165054 INCB @STREG ;CONVERT THE SELECTED CHANNEL
1670 014346 105777 165050 1$: TSTB @STREG ;WAIT FOR A/D DONE
1671 014352 100375 BPL 1$ ;
1672 014354 017700 165046 MOV @ADBUFF,R0 ;READ VALUE TO CLEAR DONE FLAG
1673 014360 000205 RTS R5 ;EXIT
1674 014362 000000 10$: 0

```

```

1676
1677 ;SUBROUTINE TO CONVERT FRONT PANEL VALUE AND TYPE OUT JPER. COMMANDS
1678 014364 010001 TYPITA: MOV R0,R1 ;COPY R0
1679 014366 006301 ASL R1 ;MAKE WORD VALUE
1680 014370 016137 014416 014400 MOV FPNL(R1),10$ ;GET TEST POINTER
1681 014376 104401 TYPE ;TELL OPERATOR THE CHANNEL POSITION
1682 014400 000000 10$: 0
1683 014402 006000 ROR R0 ;CONVERT BITS
1684 014404 006000 ROR R0
1685 014406 006000 ROR R0
1686 014410 042700 037777 BIC #37777,R0 ;MASK OFF OTHER BITS
1687 014414 000207 RTS PC ;EXIT
1688
1689 014416 000000 FPNL: 0
1690
1691 014420 032454 SCM ;POINTER TO SET CURRENT MODE TEXT
1692 014422 032525 SRM ;RESISTANCE
1693 014424 032575 SVM ;VOLTAGE
1694
1695 ;SUBROUTINE TO CONVERT CHANNEL IN "CH2"
1696
1697 014426 013537 001124 CONTA1: MOV @ (R5)+,$GDDAT ;LOAD EXPECTED VALUE
1698 014432 012737 014440 001110 MOV #10$,SLPERR ;LOAD ERROR RETURN
1699 014440 113777 001510 164756 10$: MOVB CH2,@ADST1 ;LOAD MUX CHANNEL
1700 014446 052777 000010 164746 BIS #BIT3,@STREG ;ENABLE STATUS
1701 014454 052777 000001 164740 BIS #BIT0,@STREG ;CONVERT CHANNEL
1702 014462 105777 164734 1$: TSTB @STREG ;WAIT FOR READY
1703 014466 100375 BPL 1$
1704 014470 017737 164732 001126 MOV @ADBUFF,$BDDAT ;READ CONVERSION
1705 014476 042737 037777 001126 BIC #37777,$BDDAT ;MASK OFF DATA BITS
1706 014504 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES
1707 014512 001407 BEQ 2$ ;BR IF SAME
1708 014514 013737 001510 001520 MOV CH2,CHANL ;GET CHANNEL VALUE
1709 014522 113737 014666 001521 MOVB CHPRIM,CHANL+1 ;GET GAIN INFO
1710 014530 104011 ERROR 11 ;INCORRECT FRONT PANEL SWITCH POSITION
1711 014532 000205 2$: RTS R5 ;EXIT
  
```

```

1713          :SUBROUTINE TO CONVERT CHANNEL USING GAIN
1714
1715 014534 012737 014542 001110 CON4T: MOV #10$, $LPERR ;LOAD ERROR RETURN
1716 014542 012700 000004 10$: MOV #4, R0 ;LOAD LOOP COUNTER
1717 014546 005001 CLR R1 ;CLEAR SUM VALUE
1718 014550 005077 164646 CLR @STREG ;ENSURE CLEAR STATUS
1719 014554 112777 000077 164642 MOV# #77, @ADST1 ;START ESCAPE
1720 014562 113777 014666 164634 MOV# CHPRIM, @ADST1 ;LOAD GAIN DATA
1721 014570 113777 014664 164626 MOV# CHXX, @ADST1 ;LOAD GAIN CHANNEL
1722 014576 105277 164620 1$: INCB @STREG ;CONVERT CHANNEL
1723 014602 105777 164614 2$: TSTB @STREG ;WAIT FOR READY
1724 014606 100375 BPL 2$
1725 014610 067701 164612 ADD @ADBUFF, R1 ;UPDATE SUM
1726 014614 005300 DEC R0 ;FINISHED ?
1727 014616 001367 BNE 1$ ;BR IF NOT
1728 014620 006201 ASR R1 ;RES'ORE
1729 014622 006201 ASR R1
1730 014624 010137 001126 MOV R1, $BDDA* ;LOAD ACTUAL CONVERTED VALUE
1731 014630 013737 014664 001520 MOV CHXX, CHANL ;LOAD CHANNEL VALUE IF ERROR
1732 014636 113737 014666 001521 MOV# CHPRIM, CHANL+1 ;LOAD GAIN INFO IF ERROR
1733 014644 004537 026050 JSR R5, COMPRA ;TEST AGAINST EXPECTED +- SPREAD
1734 014650 104004 ERROR 4 ;INCORRECT VALUE FROM TEST MODULE
1735 014652 000207 RTS PC ;EXIT
1736
1737
1738 014654 000000 CHANA: 0
1739 014656 000000 CHANB: 0
1740 014660 000000 CHANC: 0
1741 014662 000000 CHAND: 0
1742 014664 000000 CHXX: 0
1743 014666 000000 CHPRIM: 0
1744 014670 000000 GLD0: 0
1745 014672 000000 GLD1: 0
1746 014674 000000 GLD2: 0
1747 014676 000000 SPRAC: 0
1748 014700 000000 SPRBD: 0
1749 014702 000000 PRIAC: 0
1750 014704 000000 PRIBD: 0
1751
1752
1753
  
```



```

1755          .SBTTL      PRINT VALUES ROUTINE
1756 014706 005077 164510 BEGINP: CLR      @STREG      ;CLEAR STATUS REGISTER
1757 014712 104401 034146          TYPE      .CCHAN      ;ASK FOR CHANNEL NUMBER
1758 014716 104413          RDOCT
1759 014720 012600          MOV      (SP)+,R0      ;GET CHANNEL #
1760 014722 042700 177700          BIC      #177700,R0      ;MASK OFF OTHER BITS
1761 014726 104401 031201          TYPE      .GCHAN      ;ASK FOR CHANNEL GAIN
1762 014732 104413          RDOCT
1763 014734 012601          MOV      (SP)+,R1
1766 014736 006101          ROL      R1          ;MOVE LEFT
(1) 014740 006101          ROL      R1          ;MOVE LEFT
(1) 014742 006101          ROL      R1          ;MOVE LEFT
(1) 014744 006101          ROL      R1          ;MOVE LEFT
(1) 014746 006101          ROL      R1          ;MOVE LEFT
(1) 014750 006101          ROL      R1          ;MOVE LEFT
1767 014752 042701 177477          BIC      #177477,R1      ;MASK OFF OTHER BITS
1768 014756 050100          BIS      R1,R0          ;ADD TOGETHER
1769 014760 110077 164154          MOVB     RO,@SWR      ;LOAD SWITCH REGISTER
1770 014764 017700 164150          10$: MOV      @SWR,R0      ;GET SWITCH VALUE
1771 014770 010001          MOV      RO,R1          ;COPY RO
1772 014772 042700 177700          BIC      #177700,R0      ;MASK TO ALL BUT CHANNEL VALUE
1775 014776 006001          ROR      R1          ;MOVE RIGHT
(1) 015000 006001          ROR      R1          ;MOVE RIGHT
(1) 015002 006001          ROR      R1          ;MOVE RIGHT
(1) 015004 006001          ROR      R1          ;MOVE RIGHT
(1) 015006 006001          ROR      R1          ;MOVE RIGHT
(1) 015010 006001          ROR      R1          ;MOVE RIGHT
1776 015012 042701 177760          BIC      #177760,R1      ;MASK TO ALL BUT GAIN BITS
1777 015016 112777 000077 164400          MOVB     #77,@ADST1      ;START SEQUENCE
1778 015024 110177 164374          MOVB     R1,@ADST1      ;LOAD GAIN
1779 015030 110077 164370          MOVB     RO,@ADST1      ;LOAD SELECTED CHANNEL
1780 015034 005046          CLR      -(SP)          ;CLEAR PSW
1781 015036 012746 015044          MOV      #1$,-(SP)
1782 015042 000002          RTI
1783 015044 032777 020000 164066          1$: BIT      #BIT13,@SWR      ;IS BIT 13 SET?
1784 015052 001005          BNE      2$             ;;YES,SKIP TYPEOUT
1785 015054 104401 034014          TYPE      .CH
1786 015060 010046          MOV      RO,-(SP)      ;;SAVE RO FOR TYPEOUT
(1)          TYPOS
(1) 015062 104403          .BYTE      2           ;;TYPE CHANNEL
(1) 015064 002          .BYTE      0           ;;GO TYPE--OCTAL ASCII
(1) 015065 000          .BYTE      0           ;;TYPE 2 DIGIT(S)
1787 015066 012777 001610 164334          2$: MOV      #RETURN,@VECTOR ;;SUPPRESS LEADING ZEROS
1788 015074 010003          MOV      RO,R3          ;ADDRESS AFTER INTRPT.
1789 015076 000303          SWAB     R3             ;SWITCH BYTES
1790 015100 052703 000100          BIS      #BIT6,R3
1791 015104 010377 164312          MOV      R3,@STREG      ;LOAD THE CHANNEL
1792 015110 012702 000010          MOV      #10,R2         ;TYPEOUT COUNTER
1793 015114 012701 000010          6$: MOV      #8.,R1      ;LOAD LOOP COUNTER
1794 015120 005003          CLR      R3             ;CLEAR AVERAGE
1795 015122 005277 164274          3$: INC      @STREG      ;START CONVERSION
1796 015126 000001          WAIT
1797 015130 067703 164272          ADD      @ADBUFF,R3     ;WAIT FOR INTRPT.
1798 015134 005301          DEC      R1             ;READ CONVERTED VALUE
1799 015136 001371          BNE      3$             ;FINISHED COUNT
1800 015140 006203          ASR      R3             ;BR IF NOT
                          ;RESTORE

```

```
1801 015142 006203      ASR      R3      ; CONVERTED DATA
1802 015144 006203      ASR      R3      ; INTO CORRECT POSITION
1803 015146 005503      ADC      R3
1804 015150 042703 170000  BIC      #170000,R3 ;ENSURE 12 BIT DATA
1805 015154 032777 020000 163756 BIT      #BIT13,@SWR ;IS BIT 13 SET?
1806 015162 001403      BEQ      4$      ;NOT SET, TYPE OUT LIST
1807 015164 010377 163752  MOV      R3,@DISPLAY ;PUT VALUE IN DISPLAY FOR DISPLAY CONTROL
1808 015170 000675      BR      10$     ;REPEAT CONVERSION
1809 015172 104401 034017 4$: TYPE   ,SPACE
1810 015176 010346      MOV      R3,-(SP) ;:SAVE R3 FOR TYPEOUT
(1)                                     ;:PRINT OCTAL CONVERTED VALUE
(1) 015200 104403      TYPOS
(1) 015202 004        .BYTE   4        ;:GO TYPE--OCTAL ASCII
(1) 015203 001        .BYTE   1        ;:TYPE 4 DIGIT(S)
1811 015204 012701 010000  MOV      #10000,R1 ;:TYPE LEADING ZEROS
1812 015210 005301      DEC      R1
1813 015212 001376      BNE      5$
1814 015214 005302      DEC      R2      ;DECREMENT THE COUNTER
1815 015216 001336      BNE      6$      ;NO CARRIAGE RETURN
1816 015220 104401 001165  TYPE   ,$CRLF   ;CARRIAGE RETURN
1817 015224 000657      BR      10$     ;REPEAT CONVERSION
```

```
1819 .SBTTL LOGIC TEST SECTION START-UP
1820 015226 004737 016414 BEGL: JSR PC,WFCHK ;CHECK I D CODE IF WESTFIELD MODE
1821 015232 012737 015240 027274 MOV #2$,AGTST ;LOAD EOP RETURN IF NO A/D
1822 015240 004737 003242 2$: JSR PC,TFSTAD ;SIZE THE NUMBER OF MNCAD'S
1823 015244 004737 016606 1$: JSR PC,TCHANK ;SIZE AND REPORT THE MNCAD CONFIGURATION
1824 ;ASK IF MNCXX-TA ARE AVAILABLE
1825 015250 004737 003536 JSR PC,BEGINL ;LOGIC TESTS ON MNCAD, MNCAG
1826 015254 004737 022724 JSR PC,BUMPAD ;MORE TO TEST?
1827 015260 000771 BR 1$ ;TEST NEXT A/D
1828 015262 012737 015244 027274 MOV #1$,AGTST ;ADDRESS FOR EOP
1829 015270 000137 027076 JMP $EOP ;TYPE END OF PASS
1830 .SBTTL AUTO TEST START-UP
1831 015274 004737 003242 BEGINA: JSR PC,TESTAD ;SIZE THE # OF MNCAD'S
1832 015300 004737 016414 JSR PC,WFCHK ;CHECK I D CODE IF WESTFIELD MODE
1833 015304 004737 016620 1$: JSR PC,TCHANL ;SIZE AND REPORT THE MNCAD CONFIGURATION
1834 ;ASK IF MNCXX-TA ARE AVAILABLE
1835 015310 004737 003536 JSR PC,BEGINL ;LOGIC TESTS ON MNCAD, MNCAG
1836 015314 004737 007232 JSR PC,WRAP ;RUN THE ANALOG TESTS
1837 015320 004737 022724 JSR PC,BUMPAD ;BUMP THE ADDRESSES
1838 015324 000767 BR 1$ ;BR AND DO NEXT UNIT
1839 015326 012737 015304 027274 MOV #1$,AGTST ;ADDRESS FOR EOP
1840 015334 000137 027076 JMP $EOP ;TYPE END OF PASS
1841 .SBTTL WRAPAROUND TEST START-UP
1842 015340 004737 003242 BEGINW: JSR PC,TESTAD ;SIZE THE # OF MNCAD'S
1843 015344 004737 016414 JSR PC,WFCHK ;CHECK I D CODE IF WESTFIELD MODE
1844 015350 004737 016620 1$: JSR PC,TCHANL ;SIZE AND REPORT THE A/D CONFIG.
1845 ;ASK IF MNCXX-TA ARE AVAILABLE
1846 015354 004737 007232 JSR PC,WRAP ;WRAPAROUND TESTS
1847 015360 004737 022724 JSR PC,BUMPAD ;UPDATE BUS ADDRESSES
1848 015364 000771 BR 1$ ;BR AND TEST NEXT UNIT
1849 015366 012737 015350 027274 MOV #1$,AGTST
1850 015374 000137 027076 JMP $EOP ;INCREMENTS $PASS
1851 .SBTTL NOISE TEST START-UP
1852 015400 004737 023024 BEGINN: JSR PC,FIXONE ;ENSURE BASE AND VECTOR SETUP
1853 015404 004737 016606 JSR PC,TCHANK ;SIZE AND REPORT THE MNCAD CONFIG.
1854 015410 005037 001514 CLR NMBEXT ;CLEAR MULTIPLE UNIT FLAG
1855 015414 104401 027562 TYPE ,SCHAN ;ASK FOR STARTING NOISE CHANNEL
1856 015420 104413 RDOCT ;GET OPER. CHANNEL INPUT
1857 015422 012637 001440 MOV (SP)+,BASECH ;SAVE 1ST CHANNEL
1858 015426 104401 027612 TYPE ,ECHAN ;ASK FOR END NOISE CHANNEL
1859 015432 104413 RDOCT ;GET OPER. CHANNEL INPUT
1860 015434 012637 001442 MOV (SP)+,BASEND ;SAVE LAST CHANNEL
1861 015440 001003 BNE 1$ ;BR IF NON-ZERO
1862 015442 013737 001440 001442 MOV BASECH,BASEND ;TAKE CARE IF ONLY 1 CHANNEL
1863 015450 013737 001440 001520 1$: MOV BASECH,CHANL ;INIT THE STARTING CHANNEL
1864 015456 012737 000001 001472 MOV #1,WIDE ;SET MANUAL ENTRY FLAG
1865 015464 004737 010406 JSR PC,NOITST ;RUN NOISE TEST
1866 015470 023737 001520 001442 2$: CMP CHANL,BASEND ;LAST CHANNEL
1867 015476 001405 BEQ 3$ ;BR IF FINISHED
1868 015500 005237 001520 INC CHANL ;BUMP TO NEXT CHANNEL
1869 015504 004737 010460 JSR PC,NOITS1 ;RUN NOISE TEST AGAIN
1870 015510 000767 BR 2$
1871 015512 012737 015450 027274 3$: MOV #1$,AGTST ;LOAD RETRURN POINTER
1872 015520 000137 027076 JMP $EOP ;AND REPORT END OF PASS
```

```
1874 .SBTTL MNCAG COMMON MODE REJECTION TEST
1875 015524 104401 034650 BEGINM: TYPE ,COMOD1 ; TELL OPERATOR THE TEST NAME
1876 015530 104401 034146 TYPE ,CCHAN ; ASK FOR CHANNEL TO USE
1877 015534 104413 RDOCT ; GET INPUT
1878 015536 012600 MOV (SP)+,R0 ; GET HIS ANSWER
1879 015540 010037 001520 MOV R0,CHANL ; SAVE CHANNEL TO TEST
1880 015544 112777 000077 163652 MOVB #77,@ADST1 ; ENSURE MNCAG GAIN OF .5
1881 015552 112777 000000 163644 MOVB #0,@ADST1 ; FOR
1882 015560 110077 163640 MOVB R0,@ADST1 ; THIS TEST
1883 015564 010037 001516 MOV R0,DUMMY ; LOAD DUMMY CHANNEL
1884 015570 104401 015576 TYPE ,65$ ; TYPE ASCIZ STRING
(1) 015574 000424 BR 64$ ; GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <15><12>/SET COMMON MODE VOLTAGE TO + 10 VOLTS/
(1) 64$:
1885 015646 104401 034466 TYPE ,CRWR ; CRLF MESSAGE
1886 015652 104412 RDLIN ; WAIT FOR CARRIAGE RETURN
1887 015654 005726 TST (SP)+ ; POP ADDRESS OFF STACK
1888 015656 004537 025710 JSR R5,CONVTC ; GET CONVERSION VALUE
1889 015662 013737 001502 001536 MOV TEMP,EDGE ; GET VALUE TO FIND EDGE OF
1890 015670 004537 023540 JSR R5,SARSUB ; GET EDGE
1891 015674 000062 50. ; 50% EDGE
1892 015676 013737 001532 001502 MOV DAC,TEMP ; SAVE DAC SETTING IN TEMP
1893 015704 104401 015712 TYPE ,67$ ; TYPE ASCIZ STRING
(1) 015710 000424 BR 66$ ; GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <15><12>/SET COMMON MODE VOLTAGE TO - 10 VOLTS/
(1) 66$:
1894 015762 104401 034466 TYPE ,CRWR ; CRLF MESSAGE
1895 015766 104412 RDLIN ; WAIT FOR CARRIAGE RETURN
1896 015770 005726 TST (SP)+ ; POP ADDRESS OFF STACK
1897 015772 004537 023540 JSR R5,SARSUB ; GET EDGE
1898 015776 000062 50. ; 50% EDGE
1899 016000 163737 001502 001532 SUB TEMP,DAC ; GET DIFFERENCE
1900 016006 104401 034650 TYPE ,COMOD1 ; OUTPUT TEXT
1901 016012 013702 001532 MOV DAC,R2 ; GET NUMBER INTO R2
1902 016016 104416 TYPDC ; TYPE DECIMAL NUMBER
1903 016020 104401 035742 TYPE ,MLSB ; ADD LSB TEXT
1904 016024 013702 001532 MOV DAC,R2 ; GET RESULT
1905 016030 100001 BPL 1$ ; BR IF POSITIVE
1906 016032 005402 NEG R2 ; INVERT IF NEGATIVE
1907 016034 020237 026762 1$: CMP R2,VCM ; TEST AGAINST LIMIT
1908 016040 003403 BLE 2$ ; BR IF WITHIN LIMIT
1909 016042 104401 034611 TYPE ,ERMSG ; TELL OPER. ERROR
1910 016046 000402 BR 3$
1911 016050 104401 034135 2$: TYPE ,OKMSG ; TELL OPER. OK
1912 016054 104401 033740 3$: TYPE ,ENDTST
1913 016060 000137 001634 JMP BEG2 ; GO BACK TO SELECT TEST
```

```

1915
1916 016064 004737 023024
1917 016070 004737 016606
1918 016074 005037 001514
1919 016100 104401 030440
1920 016104 104401 027562
1921 016110 104413
1922 016112 012637 001440
1923 016116 104401 027612
1924 016122 104413
1925 016124 012637 001442
1926 016130 001003
1927 016132 013737 001440 001442
1928 016140 013737 001440 017460 1$:
1929 016146 112777 000077 163250 2$:
1930 016154 112777 000000 163242
1931 016162 113777 017460 163234
1932 016170 004737 023740
1933 016174 023737 017460 001442
1934 016202 001403
1935 016204 005237 017460
1936 016210 000756
1937 016212 012737 016140 027274 3$:
1938 016220 000137 027076
1939
1940 016224 004737 023024
1941 016230 004737 016606
1942 016234 005037 001514
1943 016240 104401 016342
1944 016244 104413
1945 016246 012637 016336
1946 016252 104401 016375
1947 016256 104413
1948 016260 012637 016340
1949 016264 042737 177700 016336
1950 016272 042737 177700 016340
1951 016300 104401 001165 1$:
1952 016304 013737 016336 001506
1953 016312 013737 016340 001510
1954 016320 004737 011230
1955 016324 012737 016300 027274
1956 016332 000137 027076
1957 016336 000000 2$:
1958 016340 000000 3$:
1959
1960 016342 051600 052105 046124 10$:
1961 016375 101 042116 041440 11$:
1962
1963
1964

```

```

.SBTTL DIFFERENTIAL LINEARITY AND REL. ACC. START-UP
BEGIN: JSR PC, FIXONE ;ENSURE BASE AND VECTOR SETUP
        JSR PC, TCHANK ;SIZE AND REPORT A/D CONFIG
        CLR NMBEXT ;ENSURE ONLY 1 MNCAD
        TYPE ,RMPTXT ;TELL OPERATOR ABOUT SETTING MNCAG-TA SWITCHES
        TYPE ,SCHANK ;ASK OPER. THE STARTING CHANNEL
        RDOCT ;GET OPER INPUT
        MOV (SP)+, BASECH ;SAVE 1ST CHANNEL
        TYPE ,ECHAN ;ASK OPER. THE LAST CHANNEL
        RDOCT ;GET OPER INPUT
        MOV (SP)+, BASEND ;SAVE LAST CHANNEL
        BNE 1$ ;BR IF THERE WAS ONE
        MOV BASECH, BASEND ;ELSE ENSURE ONLY 1ST RUNS
        MOVB #77, @ADST1 ;LOAD CHANNEL TO RUN ON
        MOVB #0, @ADST1 ;ENSURE MNCAG GAIN
        MOVB CHA, @ADST1 ; OF .5
        JSR ^C, DJFLIN ; ON THIS CHANNEL
        CMP CHA, BASEND ;RUN DIF LIN AND REL ACC.
        BEQ 3$ ;TEST IF LAST CHANNEL
        INC CHA ;BR IF FINISHED
        BR 2$ ;UPDATE CHANNEL NUMBER
        MOV #1$, AGTST ;AND RUN ANOTHER TIME
        JMP $EOP ;LOAD RETURN ADDRESS
        ;TYPE END OF PASS

.SBTTL SETTling TEST START-UP
BEGINS: JSR PC, FIXONE ;ENSURE BASE AND VECTOR SETUP
        JSR PC, TCHANK ;SIZE AND REPORT A/D CONFIG
        CLR NMBEXT ;ENSURE ONLY 1 MNCAD
        TYPE ,10$ ;ASK FOR 1ST CHANNEL
        RDOCT ;GET OPER. INPUT
        MOV (SP)+, 2$ ;AND SAVE IT
        TYPE ,11$ ;ASK FOR 2ND CHANNEL
        RDOCT ;GET OPER INPUT
        MOV (SP)+, 3$ ;AND SAVE IT
        BIC #177700, 2$ ;ENSURE GOOD CHANNEL VALUE
        BIC #177700, 3$
        TYPE , $CRLF ; FRESH LINE
        MOV 2$, CH1 ;LOAD 1ST CHANNEL VALUE
        MOV 3$, CH2 ;LOAD 2ND CHANNEL VALUE
        JSR PC, SETTLE ;RUN SETTling TEST
        MOV #1$, AGTST ;LOAD RETURN ADDRESS
        JMP $EOP ;AND REPORT END OF PASS

        0
        0
        .NLIST BEX
        .ASCIZ <200>/SETTLE BETWEEN CHANNEL /
        .ASCIZ /AND CHANNEL /
        .EVEN
        .LIST BEX

```

```

: *ROUTINE TO CHECK FOR PROPER I D CODE IF TESTER MODE
1966
1967
1968 016414 005037 016536      WFCHK: CLR      WFAD      ;CLEAR TESTING MNCAD FLAG
1969 016420 005037 016540      CLR      WFAM      ;CLEAR TESTING MNCAM FLAG
1970 016424 005037 016542      CLR      WFAG      ;CLEAR TESTING MNCAG FLAG
1971 016430 005737 001544      TST      WFTEST     ;RUNNING ON TESTER?
1972 016434 100037              BPL      4$         ;BR IF NOT
1973 016436 017700 163026      MOV      @DRVDIR,RO ;READ TESTER (I.D. LINES)
1974 016442 042700 177417      BIC      #177417,RO ;CLEAR OFF OTHER BITS
1975 016446 010037 001126      MOV      RO,$BDDAT ;LOAD VALUE READ FROM TESTER
1976 016452 023700 016544      CMP      K60,RO    ;TEST IF VALID I.D. CODE
1977 016456 001005              BNE      1$         ;BR IF NOT MNCAD CODE
1978 016460 005237 016536      INC      WFAD      ;SET TESTING MNCAD FLAG
1979 016464 104401 031650      TYPE    ,TSTAD    ;TYPE TESTING A/D MESSAGE
1980 016470 000421              BR       4$
1981 016472 023700 016550      1$:  CMP      K340,RO ;TEST IF VALID I.D. CODE FOR AM
1982 016476 001005              BNE      2$         ;NR IF NOT MNCAM CODE
1983 016500 005237 016540      INC      WFAM      ;SET TESTING MNCAM FLAG
1984 016504 104401 031672      TYPE    ,TSTADM   ;TYPE TESTING A/D AND AM MESSAGE
1985 016510 000411              BR       4$
1986 016512 023700 016546      2$:  CMP      K20,RO  ;TEST IF VALID I.D. CODE
1987 016516 001005              BNE      3$         ;BR IF NOT MNCAG
1988 016520 005237 016542      INC      WFAG      ;SET TESTING MNCAG FLAG
1989 016524 104401 031714      TYPE    ,TSTAG    ;TYPE TESTING AG MESSAGE
1990 016530 000401              BR       4$
1991 016532 104007              3$:  ERROR   7        ;INCORRECT I.D. CODE FOR MODULE
1992 016534 000207              4$:  RTS      PC     ;RETURN
1993
1994 016536 000000      WFAD:  0
1995 016540 000000      WFAM:  0
1996 016542 000000      WFAG:  0
1997
1998 016544 000060      K60:   60          ;MNCAD ID VALUE
1999 016546 000020      K20:   20          ;MNCAG  .. ..
2000 016550 000340      K340:  340        ;MNCAM  .. ..
2001
2002      ;SUBROUTINE TO DELAY A FIX AMOUNT OF TIME
2003 016552 013700 001400      STALL: MOV      BARFO,RO ;PRINE THE DELAY
2004 016556 005300      1$:  DEC      RO      ;DELAY
2005 016560 001376              BNE      1$
2006 016562 000207              RTS      PC
2007
2008      ;SUBROUTINE TO TEST IF FIRST PASS OR AUTO MODE
2009      ; IF TRUE EXIT, IF NOT BUMP ENTRY BY 1 WORD AND THEN EXIT
2009 016564 005737 001176      AFIRST: TST     $PASS  ;TEST IF FIRST PASS
2010 016570 001005              BNE      1$         ;BR IF NOT FIRST
2011 016572 105737 001134      TSTB    $AUTOB     ;TEST IF AUTO MODE
2012 016576 001002              BNE      1$         ;BR IF AUTO MODE
2013 016600 062716 000002      ADD     #2,(SP)    ;ADJUST RETURN VALUE
2014 016604 000207      1$:  RTS      PC     ;EXIT

```

```

2016 :PART 1 *ROUTINE TO TYPE OUT A/D CONFIGURATION
2017 :PART 2 *IF RUNNING IN TEST MODULE MODE, ASK FOR CHANNELS TO TEST
2018 016606 005237 017460 TCHANK: INC CHA ;SET LOGIC TEST ENTRY FLAG
2019 016612 000404 BR TCHANM ;BR
2020 016614 000137 017210 TCHANN: JMP TCHANE ;BR TO EXIT
2021 016620 005037 017460 TCHANL: CLR CHA ;CLEAR LOGIC TEST ENTRY FLAG
2022 016624 004737 011474 TCHANM: JSR PC,LD01CH ;PRESET MNCAG CHANNELS
2023 016630 005737 001176 TST $PASS ;TEST IF FIRST PASS
2024 016634 001367 BNE TCHANN ;BR AND EXIT IF NOT FIRST PASS
2025 016636 005077 162560 CLR @STREG ;CLEAR A/D STATUS
2026 016642 005037 017462 CLR CHB ;CLEAR MNCAG COUNTER
2027 016646 012700 044534 MOV #CHTABL,RO ;LOAD POINTER
2028 016652 005020 1$: CLR (RO)+ ;CLEAR CHANNEL TYPE TABLE
2029 016654 022700 044634 CMP #CHTABL+100,RO ;TEST IF FINISHED
2030 016660 001374 BNE 1$ ;BR IF NOT DONE CLEARING BUFFER
2031 016662 005000 CLR RO ;INIT RO
2032 016664 005001 CLR R1 ;INIT R1
2033 016666 004737 016564 JSR PC,AFIRST ;TEST IF FIRST PASS
2034 016672 000422 BR 3$ ;BR IF NOT
2035 016674 104401 031377 TYPE ,VTMSG ;REPORT UNIT #
2036 016700 004737 042342 JSR PC,WHICHU ;DETERMINE ASCII UNIT #
2037 016704 013746 001564 MOV UNITBD,-(SP)
2038 016710 104403 TYPOS
2039 016712 001 000 .BYTE 1,0
2040 016714 104401 001165 TYPE ,$CRLF ;LEAVE A BLANK LINE
2041 016720 004737 016564 2$: JSR PC,AFIRST ;TEST IF FIRST PASS
2042 016724 000405 BR 3$ ;BR IF NOT
2043 016726 010146 MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
(1) 016730 104403 TYPOS ;GO TYPE--OCTAL ASCII
(1) 016732 002 .BYTE 2 ;TYPE 2 DIGIT(S)
(1) 016733 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
2044 016734 104401 030237 TYPE ,MDASH ;TYPE A DASH
2045 016740 005277 162456 3$: INC @STREG ;START CONVERSION
2046 016744 105777 162452 4$: TSTB @STREG ;WAIT FOR DONE
2047 016750 100375 BPL 4$ ;BR IF NOT
2048 016752 017700 162450 MOV @ADDBUFF,RO ;GET CONVERTED VALUE
2049 016756 042700 007777 BI #7777,RO ;IS CHANNEL SINGLE ENDED
2050 016762 001007 BNL 5$ ;CHANNEL IS NOT SINGLE ENDED
2051 016764 012737 031115 017100 MOV #MSE,12$ ;LOAD MESSAGE POINTER
2052 016772 004537 022540 JSR R5,LODTAB
2053 016776 001 010 .BYTE 1,10 ;LOAD SINGLE ENDED CODE, LOAD NUMBER OF CHAN
2054 017000 000423 BR 10$
2055 017002 032700 140000 5$: BIT #140000,RO ;TEST IF MNCAG CHANNEL
2056 017006 001412 BEQ 6$ ;BR IF NOT
2057 017010 062737 000004 017462 ADD #4,CHB ;UPDATE NUMBER OF MNCAG DETECTED
2058 017016 012737 031155 017100 MOV #MPRMP,12$ ;LOAD MESSAGE POINTER
2059 017024 004537 022540 JSR R5,LODTAB
2060 017030 003 004 .BYTE 3,4 ;LOAD PREAMP CODE, LOAD NUMBER OF CHAN'S
2061 017032 000406 BR 10$
2062 017034 012737 031135 017100 6$: MOV #MDIF,12$ ;LOAD MESSAGE POINTER
2063 017042 004537 022540 JSR R5,LODTAB
2064 017046 002 004 .BYTE 2,4 ;LOAD DIFFERENTIAL CODE, LOAD NUMBER OF CHAN'S
2065 017050 022701 000100 10$: CMP #100,R1 ;IS CHANNEL > LAST POSSIBLE CHANNEL
2066 017054 101002 BHI 11$ ;NO
2067 017056 012701 000077 MOV #77,R1 ;YES, SET TO LAST CHANNEL
2068 017062 004737 016564 11$: JSR PC,AFIRST ;TEST IF FIRST PASS
  
```

2069	017066	000405				BR	13\$:BR IF NOT
2070	017070	010146				MOV	R1,-(SP)		:SAVE R1 FOR TYPEOUT
(1)	017072	104403				TYPOS			:GO TYPE--OCTAL ASCII
(1)	017074	002				.BYTE	2		:TYPE 2 DIGIT(S)
(1)	017075	000				.BYTE	0		:SUPPRESS LEADING ZEROS
2071	017076	104401				TYPE			:REPORT THE CHANNEL TYPE
2072	017100	031115			12\$:	MSE			:POINTER TO MESSAGE
2073	017102	005201			13\$:	INC	R1		:SET CHANNEL TO NEXT SET OF CHANNELS
2074	017104	022701	000100			CMP	#100,R1		:DONE?
2075	017110	001412				BEQ	14\$:YES
2076	017112	010100				MOV	R1,R0		:GET CHANNEL
2077	017114	000300				SWAB	R0		:PUT CHANNEL NUMBER IN HIGH BYTE
2078	017116	052700	000010			BIS	#BIT3,R0		:SET STATUS ENABLE BIT
2079	017122	010077	162274			MOV	R0,@STREG		:LOAD INTO A/D STATUS REGISTER
2080	017126	032777	000002	162266		BIT	#BIT1,@STREG		:IS NON-EXISTENT CHANNEL BIT SET?
2081	017134	001671				BEQ	2\$:NO
2082						:PART 2 IF USING TEST MODULE OR TESTER MODE, DO MORE TESTING			
2083						:IF NOT THEN EXIT			
2084	017136	023727	017462	000021	14\$:	CMP	CHB,#21		:TEST HOW MANY MNCAG FOUND
2085	017144	103402				BLO	15\$:BR IF LESS THAN LIMIT
2086	017146	104401	030351			TYPE	.WOWAGS		:TELL OPERATOR TOO MANY DETECTED
2087	017152	052737	100200	044534	15\$:	BIS	#100200,CHTABL		:ENSURE CH 0 + 1
2088	017160	052737	100200	044536		BIS	#100200,CHTABL+2		:AND 2 + 3 ARE TESTED
2089	017166	005737	001372			TST	ADTA		:TEST IF MNCAD-TA CONNECTED
2090	017172	001007				BNE	ASKWHO		:BR IF YES
2091	017174	005737	001374			TST	AMTA	AM	..
2092	017200	001004				BNE	ASKWHO		:BR IF YES
2093	017202	005737	001376			TST	AGTA	AG	..
2094	017206	001001				BNE	ASKWHO		:BR IF YES
2095	017210	000207				TCHANE:	RTS	PC	:EXIT IF DONE
2096						:ROUTINE TO ASK OPERATOR ABOUT MNCXX-TA BEING CONNECTED			
2097	017212	004737	016564			ASKWHO:	JSR	PC,AFIRST	:TEST IF FIRST PASS
2098	017216	000517				BR	ASKDON		:BR IF NOT
2099	017220	005737	017460			TST	CHA		:TEST IF LOGIC TEST ENTRY FLAG IS SET
2100	017224	001114				BNE	ASKDON		:BR IF IT WAS SET
2101	017226	012700	000004			MOV	#4,R0		:LOAD INITIAL CHANNEL
2102	017232	005001				CLR	R1		:INIT 2ND CHANNEL
2103						:DETERMINE IF CHANNEL (R0) IS SINGLE ENDED			
2104	017234	126027	044534	000001	ASKSE:	CMPB	CHTABL(R0),#1		:TEST IF SE
2105	017242	001027				BNE	ASKDIF		:BR IF NOT
2106	017244	062701	000007			ADD	#7,R1		:UPDATE END CHANNEL VALUE
2107	017250	120027	000004			CMPB	R0,#4		:TEST IF CHANNEL 4
2108	017254	001004				BNE	2\$:BR IF NOT
2109	017256	105737	001372		1\$:	TSTB	ADTA		:TEST IF MNCAD-TA IS CONNECTED
2110	017262	001414				BEQ	4\$:BR IF NOT
2111	017264	000406				BR	3\$		
2112	017266	120027	000010		2\$:	CMPB	R0,#10		:TEST IF CHANNEL #10
2113	017272	001771				BEQ	1\$:BR IF YES
2114	017274	105737	001374			TSTB	AMTA		:TEST IF MNCAM-TA IS CONNECTED
2115	017300	001405				BEQ	4\$:BR IF NOT
2116	017302	004737	022450		3\$:	JSR	PC,ASKC		:ASK OPERATOR
2117	017306	000402				BR	4\$:BR IF ANSWER WAS NO
2118	017310	004737	022602			JSR	PC,SETASK		:GO AND SET 'TEST THIS CHANNEL BIT'
2119	017314	005201			4\$:	INC	R1		:UPDATE TO NEXT CHANNEL
2120	017316	010100				MOV	R1,R0		:PRIME 1ST CHANNEL
2121	017320	000745				BR	ASKSE		:TEST NEXT CHANNEL


```
2122 ;DETERMINE IF THE CHANNEL IS DIFFERENTIAL (DIF)
2123 017322 126027 044534 000002 ASKDIF: CMPB CHTABL(R0),#2 ;TEST IF CHANNEL TYPE IS DIFF.
2124 017330 001024 BNE ASKAG ;BR IF NOT
2125 017332 062701 000003 ADD #3,R1 ;UPDATE TO LAST CHANNEL OF DIFF CHANNEL
2126 017336 120027 000010 CMPB R0,#10 ;TEST IF CHANNEL #10
2127 017342 001004 BNE 1$ ;BR IF NOT
2128 017344 105737 001372 TSTB ADTA ;TEST IF MNCAD-TA IS CONNECTED
2129 017350 001411 BEQ 3$ ;BR IF NOT
2130 017352 000403 BR 2$
2131 017354 105737 001374 1$: TSTB AMTA ;TEST IF MNCAM-TA IS CONNECTED
2132 017360 001405 BEQ 3$ ;BR IF NOT
2133 017362 004737 022450 2$: JSR PC,ASKC ;ASK THE OPERATOR
2134 017366 000402 BR 3$ ;BR IF ANSWER WAS NO
2135 017370 004737 022602 JSR PC,SETASK ;SET 'TEST THIS CHANNEL BIT'
2136 017374 005201 3$: INC R1 ;UPDATE CHANNEL
2137 017376 010100 MOV R1,R0 ;UPDATE 1ST CHANNEL
2138 017400 000715 BR ASKSE ;TEST NEXT CHANNEL
2139 ;DETERMINE IF THE CHANNEL IS A MNCAG
2140 017402 126027 044534 000003 ASKAG: CMPB CHTABL(R0),#3 ;TEST IF CHANNEL TYPE IS MNCAG
2141 017410 001015 BNE ASKCOOP ;BR IF NOT
2142 017412 062701 000003 ADD #3,R1 ;UPDATE TO LAST CHANNEL OF MNCAG CHANNEL
2143 017416 105737 001376 TSTB AGTA ;TEST IF MNCAG-TA IS CONNECTED
2144 017422 001405 BEQ 1$ ;BR IF NOT
2145 017424 004737 022450 JSR PC,ASKC ;ASK THE OPERATOR
2146 017430 000402 BR 1$ ;BR IF ANSWER WAS NO
2147 017432 004737 022602 JSR PC,SETASK ;SET 'TEST THIS CHANNEL BITS'
2148 017436 005201 1$: INC R1 ;UPDATE CHANNEL
2149 017440 010100 MOV R1,R0 ;UPDATE 1ST CHANNEL
2150 017442 000674 BR ASKSE ;TEST NEXT CHANNEL
2151 ;OOPS THE CHANNEL TYPE WAS NOT #1, 2, 3
2152 017444 005760 044534 ASKCOOP: TST CHTABL(R0) ;TEST IF NON-EXISTANT CHANNEL
2153 017450 001402 BEQ ASKDON ;BR IF NO MORE
2154 017452 104401 030243 TYPE ,IDONTK ;TELL OPERATOR SOME UNEXPECTED TYPE OF CHANNEL
2155 017456 000207 ASKDON: RTS PC ;EXIT
2156 017460 000000 CHA: 0
2157 017462 000000 CHB: 0
2158
2159 ;SUBROUTINE TO DO THE MNCAG NOISE TEST AT GAINS OF 50 AND 500
2160 017464 012700 045664 PRI4A: MOV #BUFFER,R0 ;CLEAR RESULT BUFFER AREA
2161 017470 005037 021042 CLR BADCAL ;CLEAR BAD CALCULATION FLAG
2162 017474 012701 010000 MOV #4C96.,R1
2163 017500 005020 1$: CLR (R0)+
2164 017502 005301 DEC R1
2165 017504 001375 BNE 1$ ;BRANCH IF NOT DONE
2166
2167 017506 013700 001520 MOV CHANL,R0 ;SETUP TO DO A CONVERSION
2168 017512 000300 SWAB R0
2169 017514 052700 000100 BIS #100,R0
2170 017520 010077 161676 MOV R0,@STREG
2171 017524 012777 001610 161676 MOV #RETURN,@VECTOR ;SETUP INTERRUPT VECTORS
2172 017532 012777 000200 161672 MOV #200,@VECTR1
2173 017540 012700 040000 MOV #16384.,R0 ;DO 16384(10) CONVERSIONS
2174 017544 005277 161652 COLECT: INC @STREG ;START CONVERSION
2175 017550 000001 WAIT ;WAIT TILL CONVERSION IS DONE
2176 017552 017701 161650 MOV @ADBUF,R1 ;READ RESULT
2177 017556 006301 ASL R1 ;GET INDEX
```

2178	017560	005261	045664		INC	BUFFER(R1)	:BUILD HISTORY TABLE
2179	017564	005300			DEC	R0	:DECREMENT NUMBER OF SAMPLES
2180	017566	001366			BNE	COLECT	:BRANCH IF NOT DONE
2181							
2182	017570	005005			CLR	R5	:SETUP INDEX
2183	017572	005037	021044		CLR	TEMPL	:SETUP TO MULTIPLY
2184	017576	005037	021046		CLR	TEMPH	
2185	017602	005037	021056		CLR	VMULH	
2186	017606	016537	045664	017632	MOV	BUFFER(R5),1\$	
2187	017614	001423			BEQ	2\$	
2188	017616	010537	021054		MOV	R5,VMULL	
2189	017622	006237	021054		ASR	VMULL	:GET CONVERTED VALUE
2190	017626	004537	021366		JSR	R5,MULTI	
2191	017632	000000			1\$:	0	
2192	017634	060037	021044		ADD	R0,TEMPL	
2193	017640	005537	021046		ADC	TEMPH	
2194	017644	060137	021046		ADD	R1,TEMPH	
2195	017650	100005			BPL	2\$:BRANCH IF NO OVERFLOW
2196	017652	004537	020652		JSR	R5,TOOBIG	:CALC. OVERFLOW
2197	017656	033611			EROVF		
2198	017660	000137	020650		JMP	TOOBAD	
2199	017664	005725			2\$:	TST	(R5)+
2200	017666	032705	020000		BIT	#BIT13,R5	:BUMP INDEX
2201	017672	001743			BEQ	XBAR	:DONE?
2202							
2203	017674	012700	000002		MOV	#2,R0	:DIVIDE BY 16384(10)
2204	017700	006337	021044		3\$:	ASL	TEMPL
2205	017704	006137	021046		ROL	TEMPH	
2206	017710	005300			DEC	R0	
2207	017712	001372			BNE	3\$	
2208							
2209	017714	005005			CLR	R5	:SETUP INDEX
2210	017716	005037	021060		CLR	V1L	:SETUP TO MULTIPLY
2211	017722	005037	021062		CLR	V1H	
2212	017726	005037	021064		CLR	V2L	
2213	017732	005037	021066		CLR	V2H	
2214	017736	016537	045664	020026	RMS2:	MOV	BUFFER(R5),2\$
2215	017744	001461			BEQ	3\$	
2216	017746	010501			MOV	R5,R1	
2217	017750	006201			ASR	R1	
2218	017752	013737	021044	021054	MOV	TEMPL,VMULL	
2219	017760	013737	021046	021056	MOV	TEMPH,VMULH	
2220	017766	160137	021056		SUB	R1,VMULH	
2221	017772	100011			BPL	1\$	
2222	017774	005137	021054		COM	VMULL	
2223	020000	005137	021056		COM	VMULH	
2224	020004	062737	000001	021054	ADD	#1,VMUL	
2225	020012	005537	021056		ADC	VMULH	
2226	020016	004737	022016		1\$:	JSR	PC,SQUARE
2227	020022	004537	021546		JSR	R5,XMULT	:EXTENDED MULTIPLICATION
2228	020026	000000			2\$:	0	
2229	020030	063737	021076	021060	ADD	XMULO,V1L	:ADD IN RESULT
2230	020036	005537	021062		ADC	V1H	
2231	020042	063737	021100	021062	ADD	XMUL1,V1H	
2232	020050	005537	021064		ADC	V2L	
2233	020054	063737	021102	021064	ADD	XMUL2,V2L	

2234	020062	005537	021066		ADC	V2H	
2235	020066	063737	021104	021066	ADD	XMUL3,V2H	
2236	020074	100005			BPL	3\$:BRANCH IF NO OVERFLOW
2237	020076	004537	020652		JSR	R5,TOOBIG	:CALC. OVERFLOW
2238	020102	033611			EROVF		
2239	020104	000137	020650		JMP	TOOBAD	
2240	020110	005725		3\$:	TST	(R5)+	
2241	020112	032705	020000		BIT	#BIT13,R5	
2242	020116	001707			BEQ	RMS2	
2243	020120	012700	000002		MOV	#2,R0	:DIVIDE BY 16384(10)
2244	020124	006337	021060	4\$:	ASL	V1L	
2245	020130	006137	021062		ROL	V1H	
2246	020134	006137	021064		ROL	V2L	
2247	020140	006137	021066		ROL	V2H	
2248	020144	100005			BPL	5\$	
2249	020146	004537	020652		JSR	R5,TOOBIG	:REPORT ERROR
2250	020152	033611			EROVF		
2251	020154	000137	020652		JMP	TOOBIG	
2252	020160	005300		5\$:	DEC	R0	
2253	020162	001360			BNE	4\$	
2254	020164	062737	100000	021060	ADD	#BIT15,V1L	:ROUND OFF NUMBER
2255	020172	005537	021062		ADC	V1H	
2256	020176	005537	021064		ADC	V2L	
2257	020202	005537	021066		ADC	V2H	
2258	020206	013737	021062	021070	MOV	V1H,SQR0	:SET UP TO FIND SQUARE ROOT
2259	020214	013737	021064	021072	MOV	V2L,SQR1	
2260	020222	013737	021066	021074	MOV	V2H,SQR2	
2261	020230	013700	021070		MOV	SQR0,R0	:CHECK FOR ZERO
2262	020234	053700	021072		BIS	SQR1,R0	
2263	020240	053700	021074		BIS	SQR2,R0	
2264	020244	001005			BNE	6\$:BR IF NON-ZERO
2265	020246	004537	020652		JSR	R5,TOOBIG	:REPORT ERROR
2266	020252	033467			ERDIV		
2267	020254	000137	020650		JMP	TOOBAD	
2268	020260	005002		6\$:	CLR	R2	:GET FIRST GUESS
2269	020262	012703	004000		MOV	#2048.,R3	
2270	020266	010237	021060	SQRR:	MOV	R2,V1L	:SETUP FOR DIVISION
2271	020272	010337	021062		MOV	R3,V1H	
2272	020276	004737	021064		JSR	PC,XDIVI	:GO DO DIVISION
2273	020302	060237	021054		ADD	R2,VMULL	:GET NEXT GUESS
2274	020306	005537	021056		ADC	VMULH	
2275	020312	060337	021056		ADD	R3,VMULH	
2276	020316	006237	021056		ASR	VMULH	
2277	020322	006037	021054		ROR	VMULL	
2278	020326	023703	021056		CMP	VMULH,R3	:IS NUMBER DIFFERENT?
2279	020332	001003			BNE	1\$:YES
2280	020334	023702	021054		CMP	VMULL,R2	
2281	020340	001414			BEQ	PRMS	:NO
2282	020342	013702	021054	1\$:	MOV	VMULL,R2	:SETUP FOR NEXT GUESS
2283	020346	013703	021056		MOV	VMULH,R3	
2284	020352	010200			MOV	R2,R0	:TEST FOR DIVISION BY ZERO
2285	020354	050300			BIS	R3,R0	
2286	020356	001343			BNE	SQRR	
2287	020360	004537	020652		JSR	R5,TOOBIG	:CALC. ERROR
2288	020364	033467			ERDI!		
2289	020366	000137	020650		JMP	TOOBAD	

```
2291 ;NOW THAT THE RMS NUMBER CRUNCHING AND COLLECTION IS DONE, TEST THE RESULTS
2292 020372 005737 021042 PRMS: TST BADCAL ;TEST IF A BAD CALCULATION OCCURRED
2293 020376 001402 BEQ 1$ ;BR IF NOT
2294 020400 000137 020650 JMP TOOBAD ;DONT TEST IF WITHIN LIMITS
2295 020404 013737 021054 021050 1$: MOV VMULL,VMULLS ;SAVE IT
2296 020412 013737 021056 021052 MOV VMULH,VMULHS
2297 020420 104401 034066 TYPE ,RMSNOI ;AND NOISE TEXT
2298 020424 004737 022212 JSR PC,PRGAIN ;TYPE OUT RESULT
2299 020430 104401 035742 TYPE ,MLSB ;ADD LSB TEXT
2300 020434 004737 025664 JSR PC,PSOINOI ;ADD CHANNEL REPORT
2301 020440 004537 020760 JSR R5,ERCHKG ;CHECK IF WITHIN LIMITS
2302 020444 000000 AGCHRA: 0 ;MSW OF RMS LIMIT
2303 020446 000000 AGCHRB: 0 ;LSW OF RMS LIMIT
2304 ;NOW TAKE THE COLLECTED DATA AND DETERMINE THE PEAK NUMBERS
2305 020450 012700 017776 MOV #<<4095.*2>,R0 ;GET OFFSET TO LAST ENTRY
2306 020454 006260 045664 1$: ASR BUFFER(R0) ;DIVIDE COUNT BY 64
2307 020460 006260 045664 ASR BUFFER(R0)
2308 020464 006260 045664 ASR BUFFER(R0)
2309 020470 006260 045664 ASR BUFFER(R0)
2310 020474 006260 045664 ASR BUFFER(R0)
2311 020500 006260 045664 ASR BUFFER(R0)
2312 020504 005300 DEC R0
2313 020506 005300 DEC R0
2314 020510 100361 BPL 1$
2315 020512 005000 CLR R0 ;NOW FOR PEAK NOISE ON THE NOISY THING
2316 020514 005760 045664 PEAKN: TST BUFFER(R0) ;WAS THERE A HIT HERE?
2317 020520 001002 BNE 1$ ;YES
2318 020522 005720 TST (R0)+ ;GO TO NEXT STATE AND TRY AGAIN
2319 020524 000773 BR PEAKN ;WILL MIRACLES EVER CEASE
2320 020526 010001 1$: MOV R0,R1 ;SAVE MIN IN R1
2321 020530 012700 017776 MOV #17776,R0 ;NOW TO FIND MAX
2322 020534 005760 045664 2$: TST BUFFER(R0) ;WAS THERE A HIT HERE?
2323 020540 001002 BNE 3$ ;YES
2324 020542 005740 TST -(R0) ;GO TO PREVIOUS STATE AND TRY AGAIN
2325 020544 000773 BR 2$ ;ANOTHER MIRACLE
2326 020546 160100 3$: SUB R1,R0 ;GET PEAK NOISE
2327 020550 006200 ASR R0
2328 020552 010037 021056 MOV R0,VMULH
2329 020556 005037 021054 CLR VMULL
2330 020562 006237 021056 ASR VMULH
2331 020566 006037 021054 ROR VMULL
2332 020572 005737 021042 TST BADCAL ;TEST IF BAD CALCULATION OCCURRED
2333 020576 001402 BEQ 4$ ;BR IF NONE
2334 020600 000137 020650 JMP TOOBAD ;IF SOME DONT TEST AGAINST LIMITS
2335 020604 013737 021054 021050 4$: MOV VMULL,VMULLS ;SAVE IT
2336 020612 013737 021056 021052 MOV VMULH,VMULHS
2337 ;REPORT THE PEAK RESULTS TO THE OPERATOR
2338 020620 104401 034102 TYPE ,PKNOI ;AND PEAK TEXT
2339 020624 004737 022212 JSR PC,PRGAIN ;TYPE OUT FANTASTIC RESULT???
2340 020630 104401 035742 TYPE ,MLSB ;ADD LSB TEXT
2341 020634 004737 025664 JSR PC,PSOINOI ;ADD CHANNEL REPORT
2342 020640 004537 020760 JSR R5,ERCHKG ;CHECK IF WITHIN LIMITS
2343 020644 000000 AGCHPA: 0 ;MSW OF PEAK LIMIT
2344 020646 000000 AGCHPB: 0 ;LSW OF PEAK LIMIT
2345 020650 000207 TOOBAD: RTS PC ;EXIT
```

```

2347      :SUBROUTINE TO HANDLE CALCULATION ERRORS
2348 020652 010537 020756      TOOBIG: MOV      R5,11$      :SAVE CALLING ADDRESS
2349 020656 162737 000004 020756      SUB      #4,11$      :CORRECT THE VALUE
2350 020664 013737 020756 021042      MOV      11$,BADCAL  :LOAD LOCATION OF ERROR INTO FLAG
2351 020672 012537 020714      MOV      (R5)+,10$   :SAVE TRAILING ARGUMENT
2352 020676 032777 020000 160234      BIT      #SW13,@SWR  :TEST IF INHIBIT REPORT IS SET
2353 020704 001017      BNE      1$         :BR IF SET
2354 020706 104401 033420      TYPE     ,EXCNOI    :REPORT EXCESSIVE NOISE CAUSED FATAL MATH ERROR
2355 020712 104401      TYPE     :TELL OPER THE BAD NEWS
2356 020714 000000      10$: 0             :POINTER TO ASCII TEXT MESSAGE
2357 020716 013746 020756      MOV      11$,-(SP)  :MOVE BAD PC TO STACK
2358 020722 104402      TYPOC    :AND ADD TO ERROR TYPEOUT
2359 020724 104401 034116      TYPE     ,CHAN      :ADD CHANNEL TEXT
2360 020730 013746 001520      MOV      CHANL,-(SP) :AND CHANNEL NUMBER
2361 020734 104403      TYPOS    :
2362 020736      002      000      .BYTE     2,0
2363 020740 104401 001165      TYPE     ,$CRLF     :ADD CRLF
2364 020744 004737 042334      1$: JSR      PC,WHICHV :DETERMINE THE FAILING UNIT MASK
2365 020750 005237 001112      INC      $ERTTL     :UPDATE ERROR TOTAL
2366 020754 000205      RTS      R5        :EXIT
2367 020756 000000      11$: 0
2368      :SUBROUTINE TO CHECK WITHIN LIMITS
2369 020760 012537 021036      ERCHKG: MOV      (R5)+,10$   :GET MSW VALUE
2370 020764 012537 021040      MOV      (R5)+,11$   :GET LSW VALUE
2371 020770 023737 021036 021052      CMP      10$,VMULHS  :COMPARE MSW
2372 020776 100410      BMI     1$         :BR IF EXCESSIVE
2373 021000 001004      BNE     3$         :BR IF OK
2374 021002 023737 021040 021050      CMP      11$,VMULLS  :COMPARE LSW
2375 021010 100403      BMI     1$         :BR IF EXCESSIVE
2376 021012 104401 034135      3$: TYPE     ,OKMSG    :REPORT ITS OK
2377 021016 000406      BR      2$
2378 021020 104401 034611      1$: TYPE     ,ERMSG    :REPORT ITS ERROR
2379 021024 004737 042334      JSR      PC,WHICHV  :DETERMINE UNIT
2380 021030 005237 001112      INC      $ERTTL     :UPDATE ERROR COUNT
2381 021034 000205      2$: RTS      R5        :EXIT
2382 021036 000000      10$: 0
2383 021040 000000      11$: 0
2384 021042 000000      BADCAL: 0          :BAD CALC. FLAG
2385 021044 000000      TEMPL: 0
2386 021046 000000      TEMPH: 0
2387 021050 000000      VMULLS: 0         :TEMP LOC. OF VMULL
2388 021052 000000      VMULHS: 0         :TEMP LOC. OF VMULH
2389 021054 000000      VMULL: 0
2390 021056 000000      VMULH: 0
2391 021060 000000      V1L: 0
2392 021062 000000      V1H: 0
2393 021064 000000      V2L: 0
2394 021066 000000      V2H: 0
2395 021070 000000      SQR0: 0
2396 021072 000000      SQR1: 0
2397 021074 000000      SQR2: 0
2398 021076 000000      XMULO: 0
2399 021100 000000      XMUL1: 0
2400 021102 000000      XMUL2: 0
2401 021104 000000      XMUL3: 0

```

```

2403
2404 ;DOUBLE PRECISION DIVIDER FOR DECIMAL DIVISION OF TWO DOUBLE
2405 ;PRECISION NUMBERS.
2406 ;
2407 ; ENTER WITH DIVIDEND IN V2 DIVISOR IN V1
2408 ; RETURNS WHOLE NUMBER IN VMULH, DECIMAL PART IN VMULL
2409 ; REMAINDER IN V2
2410 021106 012700 000020 DIVI: MOV #16,R0 ;SET UP DECIMAL COUNT
2411 021112 005037 021056 CLR VMULH ;CLEAR WHOLE PART OF RESULT
2412 021116 005037 021054 CLR VMULL ;CLEAR DECIMAL PART OF RESULT
2413 021122 005046 CLR -(SP) ;CLEAR SIGN OF RESULT
2414 021124 005737 021062 TST V1H ;IS V1 NEGATIVE?
2415 021130 100012 BPL 1$ ;NO
2416 021132 005216 INC (SP) ;INCREMENT SIGN FLAG
2417 021134 005137 021060 COM V1L ;TWO'S COMPLEMENT V1
2418 021140 005137 021062 COM V1H
2419 021144 062737 000001 021060 ADD #1,V1L
2420 021152 005537 021062 ADC V1H
2421 021156 005737 021066 1$: TST V2H ;IS V2 NEGATIVE?
2422 021162 100012 BPL 2$ ;NO
2423 021164 005316 DEC (SP) ;DECREMENT SIGN FLAG
2424 021166 005137 021060 COM V1L ;TWO'S COMPLEMENT V2
2425 021172 005137 021062 COM V1H
2426 021176 062737 000001 021060 ADD #1,V1L
2427 021204 005537 021062 ADC V1H
2428 021210 163737 021060 021064 2$: SUB V1L,V2L ;SUBTRACT V1 FROM V2
2429 021216 005637 021066 SBC V2H
2430 021222 163737 021062 021066 SUB V1H,V2H
2431 021230 100406 BMI 3$ ;BRANCH IF SUBTRACT FAILED
2432 021232 005237 021056 INC VMULH ;ADD ONE TO WHOLE NUMBER RE JLT
2433 021236 100364 BPL 2$ ;TRY ANOTHER SUBTRACTION
2434 021240 004537 020652 JSR R5,TOOBIG ;CALC. OVERFLOW
2435 021244 033467 ERDIV
2436 021246 063737 021060 021064 3$: ADD V1L,V2L ;ADD V1 TO V2
2437 021254 005537 021066 ADC V2H
2438 021260 063737 021062 021066 ADD V1H,V2H
2439 021266 005300 4$: DEC R0 ;DECREMENT DECIMAL COUNT
2440 021270 100422 BMI 5$ ;BRANCH IF DONE
2441 021272 006337 021064 ASL V2L ;MULTIPLY V2 BY 2
2442 021276 006137 021066 ROL V2H
2443 021302 006337 021054 ASL VMULL ;MULTIPLY VMULL BY 2
2444 021306 163737 021060 021064 SUB V1L,V2L ;SUBTRACT V1 FROM V2
2445 021314 005637 021066 SBC V2H
2446 021320 163737 021062 021066 SUB V1H,V2H
2447 021326 100747 BMI 3$ ;BRANCH IF SUBTRACTION FAILED
2448 021330 005237 021054 INC VMULL ;INCREMENT DECIMAL RESULT
2449 021334 000754 BR 4$ ;TRY AGAIN
2450 021336 005726 5$: TST (SP)+ ;TEST SIGN FLAG
2451 021340 001411 BEQ 6$ ;NUMBER IS POSITIVE
2452 021342 005137 021054 COM VMULL ;TWO'S COMPLEMENT RESULT
2453 021346 005137 021056 COM VMULH
2454 021352 062737 000001 021054 ADD #1,VMULL
2455 021360 005537 021056 ADC VMULH
2456 021364 000207 6$: RTS PC ;RETURN FROM DIVI
  
```

```

2458 ;ROUTINE TO MULTIPLY TWO NUMBERS
2459 ;CALL: JSR R5,MULTI
2460 ;      MULTIPLIER
2461 ;
2462 ;MULTIPLIES VMUL BY MULTIPLIER, RESULT IN R0 & R1 WITH THE LOW BYTE
2463 ;      IN R0 HIGH BYTE IN R1
2464
2465 MULTI: CLR -(SP) ;CLEAR SIGN FLAG
2466 CLR R0 ;CLEAR WORK REGISTERS
2467 CLR R1
2468 MOV #BIT15,R2 ;SETUP TEST BIT, MULTIPLIER IS UNSIGNED
2469 TST VMULH ;TEST SIGN
2470 BPL 1$ ;BRANCH IF POSITIVE
2471 INC (SP) ;INCREMENT SIGN FLAG
2472 COM VMULL ;TWO'S COMPLEMENT NUMBER
2473 COM VMULH
2474 ADD #1,VMULL
2475 ADC VMULH
2476 1$: ASL R0 ;MULTIPLY RESULT BY 2
2477 ROL R1
2478 BCC 4$ ;BR IF NO **MULTIPLICATION ERROR**
2479 JSR R5,TOOBIG ;REPORT ERROR
2480 ERMUL
2481 4$: BIT R2,(R5) ;TEST MULTIPLIER BIT
2482 BEQ 2$ ;BRANCH IF BIT IS CLEAR
2483 ADD VMULL,R0 ;ADD NUMBER TO RESULT
2484 ADC R1
2485 ADD VMULH,R1
2486 BCC 2$ ;BR IF NO **MULTIPLICATION ERROR**
2487 JSR R5,TOOBIG ;REPORT ERROR
2488 ERMUL
2489 2$: CLC ;SHIFT TEST BIT RIGHT
2490 ROR R2
2491 BNE 1$ ;BRANCH IF NOT DONE
2492 TST (R5)+ ;BUMP RETURN POINTER
2493 TST (SP)+ ;TEST SIGN FLAG
2494 BEQ 3$ ;BRANCH IF POSITIVE
2495 COM R0 ;TWO'S COMPLEMENT THE RESULT
2496 COM R1
2497 ADD #1,R0
2498 ADC R1
2499 COM VMULL ;TWO'S COMPLEMENT NUMBER
2500 COM VMULH
2501 ADD #1,VMULL
2502 ADC VMULH
2503 3$: RTS ;RETURN FROM MULTI
2504

```

```

2506 ;ROUTINE TO MULTIPLY A TRIPLE PRECISION NUMBER
2507 ;BY A SINGLE PRECISION NUMBER GIVING A QUADUPLE PRECISION RESULT
2508
2509 021546 005037 021076 XMULT: CLR XMUL0 ;CLEAR RESULT
2510 021552 005037 021100 CLR XMUL1
2511 021556 005037 021102 CLR XMUL2
2512 021562 005037 021104 CLR XMUL3
2513 021566 012537 021054 MOV (R5)+,VMULL ;SETUP FOR MULTIPLICATION
2514 021572 005037 021056 CLR VMULH
2515 021576 013737 021070 021610 MOV SQRO,1$
2516 021604 004537 021366 JSR R5,MULTI ;GET FIRST TERM
2517 021610 000000 1$: 0
2518 021612 010037 021076 MOV R0,XMUL0 ;SAVE FIRST RESULT
2519 021616 010137 021100 MOV R1,XMUL1
2520 021622 013737 021072 021634 MOV SQR1,2$ ;PREPARE FOR SECOND MULTIPLICATION
2521 021630 004537 021366 JSR R5,MULTI ;GET SECOND TERM
2522 021634 000000 2$: 0
2523 021636 060037 021100 ADD R0,XMUL1 ;ADD TO FIRST RESULT (SHIFTED)
2524 021642 005537 021102 ADC XMUL2
2525 021646 060137 021102 ADD R1,XMUL2
2526 021652 013737 021074 021666 MOV SQR2,3$ ;PREPARE FOR THIRD MULTIPLICATION
2527 021660 000240 NOP ;**FOR DEBUG**
2528 021662 004537 021366 JSR R5,MULTI ;GET THIRD TERM
2529 021666 000000 3$: 0
2530 021670 060037 021102 ADD R0,XMUL2 ;ADD TO FIRST & SECOND (SHIFTED)
2531 021674 005537 021104 ADC XMUL3
2532 021700 060137 021104 ADD R1,XMUL3
2533 021704 100003 BPL 4$ ;BR IF NO ERROR IN MULTIPLICATION
2534 021706 004537 020652 JSR R5,TOOBIG ;REPORT ERROR
2535 021712 033535 ERMUL
2536 021714 000205 4$: RTS R5 ;BACK TO WHERE WE CAME
2537
2538 ;ROUTINE TO DIVIDE A TRIPLE PRECISION NUMBER
2539 ;BY A DOUBLE PRECISION NUMBER GIVING A DOUBLE PRECISION RESULT
2540
2541 021716 013737 021072 021064 XDIVI: MOV SQR1,V2L ;SETUP FOR FIRST DIVIDE
2542 021724 013737 021074 021066 MOV SQR2,V2H
2543 021732 004737 021106 JSR PC,DIVI ;GET FIRST RESULT
2544 021736 005737 021056 TST VMULH ;DID OVERFLOW OCCUR?
2545 021742 001403 BEQ 1$ ;NO
2546 021744 004537 020652 JSR R5,TOOBIG ;REPORT ERROR
2547 021750 033467 ERDIV
2548 021752 013737 021054 021046 1$: MOV VMULL,TEMPH ;SAVE FIRST RESULT
2549 021760 063737 021070 021064 ADD SQRO,V2L ;SETUP FOR SECOND DIVIDE
2550 021766 005537 021066 ADC V2H
2551 021772 004737 021106 JSR PC,DIVI ;GET SECOND RESULT
2552 021776 063737 021046 021056 ADD TEMPH,VMULH ;ADD IN FIRST RESULT
2553 022004 100003 BPL 2$ ;BRANCH IF NO OVERFLOW
2554 022006 004537 020652 JSR R5,TOOBIG ;REPORT ERROR
2555 022012 033467 ERDIV
2556 022014 000207 2$: RTS PC

```



```
2558 ;ROUTINE TO SQUARE A 32 BIT NUMBER WITH 16 BITS AFTER POINT
2559 ;RETURNS A 32 BIT NUMBER WITH 16 BITS AFTER POINT
2560
2561 022016 013746 021056 SQUARE: MOV VMULH,-(SP) ;SAVE 32 BIT NUMBER ON STACK
2562 022022 013746 021054 MOV VMULL,-(SP)
2563 022026 005037 021056 CLR VMULH ;SETUP FOR FIRST MULTIPLICATION
2564 022032 013737 021054 022044 MOV VMULL,1$
2565 022040 004537 021366 JSR R5,MULTI
2566 022044 000000 1$: 0
2567 022046 062700 100000 ADD #BIT15,R0 ;ROUND OFF DECIMAL PART
2568 022052 005501 ADC R1
2569 022054 010137 021070 2$: MOV R1,SQR0 ;SAVE RESULT
2570 022060 005037 021072 CLR SQR1
2571 022064 005037 021074 CLR SQR2
2572 022070 012637 021054 MOV (SP)+,VMULL ;SETUP FOR SECOND MULTIPLICATION
2573 022074 005037 021056 CLR VMULH
2574 022100 011637 022110 MOV (SP),3$
2575 022104 004537 021366 JSR R5,MULTI
2576 022110 000000 3$: 0
2577 022112 006300 ASL R0 ;MULTIPLY RESULT BY 2
2578 022114 006101 ROL R1
2579 022116 060037 021070 ADD R0,SQR0
2580 022122 005537 021072 ADC SQR1
2581 022126 060137 021072 ADD R1,SQR1 ;ADD TO PREVIOUS RESULT
2582 022132 005537 021074 ADC SQR2
2583 022136 100003 BPL 4$
2584 022140 004537 020652 JSR R5,TOOBIG ;REPORT ERROR
2585 022144 033654 ERSQR
2586 022146 011637 021054 4$: MOV (SP),VMULL ;SETUP FOR LAST MULTIPLICATION
2587 022152 012637 022162 MOV (SP)+,5$
2588 022156 004537 021366 JSR R5,MULTI
2589 022162 000000 5$: 0
2590 022164 060037 021072 ADD R0,SQR1 ;ADD IN LAST FIGURE
2591 022170 005537 021074 ADC SQR2
2592 022174 060137 021074 ADD R1,SQR2
2593 022200 100003 BPL 6$
2594 022202 004537 020652 JSR R5,TOOBIG ;REPORT ERROR
2595 022206 033654 ERSQR
2596 022210 000207 6$: RTS PC ;RETURN
2597
2598 ;SUBROUTINE TO PRINT THE VOLTAGE GAIN
2599 022212 062737 000510 021054 PRGAIN: ADD #510,VMULL ;ADD .005 LSB FOR ROUNDING REASONS
2600 022220 004737 022266 JSR PC,TYPDEC ;TYPE OUT DECIMAL NUMBER
2601 022224 104401 022232 TYPE ,65$ ;:TYPE ASCII STRING
(1) 022230 000401 BR 64$ ;:GET OVER THE ASCII
(1) ;:65$: .ASCIIZ /./
(1) 64$:
2602 022234 012705 000002 1$: MOV #2,R5 ;SET UP # OF DECIMAL PLACES
2603 022240 004537 021366 JSR R5,MULTI ;MULTIPLY DECIMAL FRACTION BY 10(10)
2604 022244 000012 10.
2605 022246 010037 021054 MOV R0,VMULL ;SAVE DECIMAL PART
2606 022252 010100 MOV R1,R0 ;PUT NUMBER IN R0
2607 022254 004737 022432 JSR PC,TYPDIG ;TYPE OUT DIGIT
2608 022260 005305 DEC R5 ;DECREMENT DIGIT COUNT
2609 022262 001366 BNE 1$ ;BRANCH IF NOT DONE
2610 022264 000207 RTS PC ;RETURN FROM PRGAIN
```

```

2611
2612      ;SUBROUTINE TO TYPE OUT A DECIMAL NUMBER
2613 022266 005737 021056  TYPDEC: TST      VMULH      ;TEST NUMBER
2614 022272 001005          BNE      1$      ;BRANCH IF NUMBER NOT ZERO
2615 022274 104401 022302  TYPE     ,65$    ;:TYPE ASCIZ STRING
      (1) 022300 000401          BR      64$    ;:GET OVER THE ASCIZ
      (1)          ;:65$: .ASCIZ /0/
      (1) 022304          64$:      RTS      PC      ;RETURN FROM TYPDEC
2616 022304 000207          1$:      BPL     DFCPRT  ;BRANCH IF NUMBER POSITIVE
2617 022306 100015          TYPE     ,67$    ;:TYPE ASCIZ STRING
2618 022310 104401 022316  BR      66$    ;:GET OVER THE ASCIZ
      (1) 022314 000401          ;:67$: .ASCIZ /-/
      (1) 022320          66$:      COM     VMULL      ;TWO'S COMPLEMENT NUMBER
2619 022320 005137 021054  COM     VMULH
2620 022324 005137 021056  ADD     #1,VMULL
2621 022330 062737 000001 021054  ADC     VMULH
2622 022336 005537 021056  DECPRT: TST     VMULH      ;TEST NUMBER
2623 022342 005737 021056  BNE     1$      ;BRANCH IF NUMBER NOT ZERO
2624 022346 001001          RTS      PC      ;RETURN
2625 022350 000207          1$:      MOV     R0,-(SP) ;SAVE WORK REGISTER
2626 022352 010046          MOV     #50000,R1 ;GET TEST NUMBER
2627 022354 012701 050000  MOV     VMULH,R0 ;GET DIVIDEND
2628 022360 013700 021056  CLR     VMULH    ;CLEAR RESULT
2629 022364 005037 021056  2$:      ASL     VMULH    ;DIVIDE R0 BY 10
2630 022370 006337 021056  CMP     R0,R1   ;RESULT IN VMULH
2631 022374 020001          BMI     3$      ;REMAINDER IN R0
2632 022376 100403          SUB     R1,R0
2633 022400 160100          INC     VMULH
2634 022402 005237 021056  3$:      ASR     R1
2635 022406 006201          CMP     #5,R1   ;TEST FOR DONE
2636 022410 022701 000005  BNE     2$      ;BRANCH IF NOT DONE
2637 022414 001365          JSR     PC,DECPRT ;DO DIVISION AGAIN TILL VMULH = 0
2638 022416 004737 022342  JSR     PC,TYPDIG ;TYPE OUT DIGIT
2639 022422 004737 022432  MOV     (SP)+,R0 ;RESTORE WORK REGISTER
2640 022426 012600          RTS     PC      ;RETURN
2641 022430 000207          TYPDIG: ADD     #60,R0 ;MAKE NUMBER ASCII
2642 022432 062700 000060  MOVB   R0,ONES ;SAVE FOR TYPEOUT
2643 022436 110037 040266  TYPE   ,ONES   ;TYPE OUT NUMBER
2644 022442 104401 040266  RTS     PC      ;RETURN FROM TYPDIG
2645 022446 000207

```

```
2647 ;*SUB-ROUTINE TO ASK CHANNELS TO TEST
2648
2649 022450 104401 031574 ASKC: TYPE ,TCHAN ;TYPE 'TEST CHANNELS ''
2650 022454 010046 MOV RO,-(SP) ;:SAVE R0 FOR TYPEOUT
(1) 022456 104403 TYPOS ;:GO TYPE--OCTAL ASCII
(1) 022460 002 .BYTE 2 ;:TYPE 2 DIGIT(S)
(1) 022461 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
2651 022462 104401 030237 TYPE ,MDASH ;TYPE '' - ''
2652 022466 010146 MOV R1,-(SP) ;:SAVE R1 FOR TYPEOUT
(1) 022470 104403 TYPOS ;:GO TYPE--OCTAL ASCII
(1) 022472 002 .BYTE 2 ;:TYPE 2 DIGIT(S)
(1) 022473 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
2653 022474 104401 030233 TYPE ,QUEST ;TYPE '' ? ''
2654 022500 104412 RDLIN ;GET RESPONSE
2655 022502 012602 MOV (SP)+,R2 ;GET ADDRESS OF RESPONSE TEXT
2656 022504 142712 000040 BICB #40,(R2) ;MAKE CHARACTER UPPFR CASE
2657 022510 122712 000131 CMPB #'Y,(R2) ;IS IT A Y?
2658 022514 001406 BEQ 1$ ;:YES
2659 022516 122712 000116 CMPB #'N,(R2) ;IS IT AN N?
2660 022522 001405 BEQ 2$ ;:YES
2661 022524 104401 031615 TYPE ,YESNO ;TYPE 'TYPE Y FOR YES, N FOR NO''
2662 022530 000747 BR ASKC ;:
2663 022532 062716 000002 1$: ADD #2,(SP) ;SKIP OVER BRANCH
2664 022536 000207 2$: RTS PC ;RETURN
2665
2666 ;SUBROUTINE TO LOAD THE TYPE OF CHANNEL CODE INTO 'CHTABL' BUFFER
2667 022540 112537 022576 LODTAB: MOVB (R5)+,10$ ;GET CODE VALUE
2668 022544 112537 022600 MOVB (R5)+,11$ ;GET NUMBER OF CHANNELS
2669 022550 113761 022576 044534 1$: MOVB 10$,CHTABL(R1) ;SAVE THIS CHANNELS TYPE
2670 022556 105337 022600 DECB 11$ ;MORE CHANNELS ?
2671 022562 001402 BEQ 2$ ;BR IF DONE
2672 022564 005201 INC R1 ;UPDATE CHANNEL NUMBER
2673 022566 000770 BR 1$ ;LOAD NEXT CHANNEL TYPE
2674 022570 000240 2$: NOP
2675 022572 000240 NOP
2676 022574 000205 RTS R5 ;EXIT
2677 022576 000000 10$: 0
2678 022600 000000 11$: 0
2679
2680 ;SUBROUTINE TO SET THE 'TEST THIS CHANNEL' BIT
2681 022602 152760 000200 044534 SETASK: BISB #BIT7,CHTABL(R0) ;SET THE BIT
2682 022610 020001 CMP R0,R1 ;FINISHED LOADING
2683 022612 001402 BEQ 1$ ;BR IF DONE
2684 022614 005200 INC R0 ;UPDATE CHANNEL NUMBER
2685 022616 000771 BR SETASK ;BR BACK
2686 022620 000207 1$: RTS PC ;EXIT
```

```

2688
2689 ;SUBROUTINE TO CHANGE BASE AND VECTOR ADDRESSES
2690 022622 104401 031300 BASEXC: TYPE ,MADR ;ASK FOR MODULE ADDRESS
2691 022626 013746 001244 MOV $BASE,-(SP) ;;SAVE $BASE FOR TYPEOUT
(1) 022632 104402 TYPEOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2692 022634 104401 031372 TYPE ,ENCOM
2693 022640 104413 RDOCT
2694 022642 005726 TST (SP)+ ;DEFAULT ADDRESS ?
2695 022644 001403 BEQ 5$ ;NO BRANCH
2696 022646 016637 177776 001244 MOV -2(SP), $BASE ;SAVE ADDRESS IN $BASE
2697 022654 104401 031334 5$: TYPE ,MVCT ;ASK FOR MODULE VECTOR
2698 022660 013701 001240 MOV $VECT1,R1 ;GET VECTOR
2699 022664 010146 MOV R1,-(SP) ;;SAVE R1 FOR TYPEOUT
(1) 022666 104403 TYPEOC ;GO TYPE--OCTAL ASCII
(1) 022670 003 .BYTE 3 ;TYPE 3 DIGIT(S)
(1) 022671 001 .BYTE 1 ;TYPE LEADING ZEROS
2700 022672 104401 031372 TYPE ,ENCOM
2701 022676 104413 RDOCT
2702 022700 005726 TST (SP)+ ;TAKE DEFAULT ?
2703 022702 001403 BEQ 7$
2704 022704 016637 177776 001240 MOV -2(SP), $VECT1
2705 022712 052737 100000 001240 7$: BIS #BIT15,$VECT1 ;SET PRIORITY LEVEL
2706 022720 000137 002730 JMP MTEST1 ;RFSTART
  
```

```

2708
2712          .SBTTL      DETERMINE IF MORE MNCAD'S TO BE TESTED
2713 022724 005737 001512  BUMPAD: TST      NBEXT          ;ADDITIONAL AD'S?
2714 022730 001433          BEQ      FIXADR          ;NO-INITIALIZE ADDRESSES
2715 022732 006337 001562          ASL      MASKNM          ;MOVE BIT TO NEXT MODULE
2716 022736 005001          CLR      R1
2717 022740 013700 001562          MOV      MASKNM,R0          ;GET MASK NUMBER
2718 022744 006200          1$:  ASR      R0          ;MOVE RIGHT
2719 022746 001403          BEQ      2$          ;BR IF DONE
2720 022750 062701 000004          ADD      #4,R1          ;UPDATE INDEX VALUE
2721 022754 000773          BR      1$
2722 022756 016137 001402 001422 2$:  MOV      MNCADO(R1),STREG ;GET NEW ADDRESS
2723 022764 062701 000002          ADD      #2,R1          ;NEW NEXT INDEX
2724 022770 016137 001402 001430          MOV      MNCADO(R1),VECTOR ;GET NEW VECTOR
2725 022776 013737 001422 001424          MOV      STREG,ADST1      ;PRIME OTHER ADDRESSES
2726 023004 013737 001422 001426          MOV      STREG,ADBUFF
2727 023012 005337 001512          DEC      NBEXT          ;ONE LESS MNCAD
2728 023016 000427          BR      BYPASS
2729 023020 062716 000002          FIXADR: ADD     #2,(SP)
2730 023024 012737 027276 000004  FIXONE: MOV     #IOTRD,ERRVEC ;SET UP ERRVEC
2731 023032 012737 000001 001562          MOV     #1,MASKNM          ;INIT. MODULE ERROR TEST BIT
2732 023040 013737 001244 001422          MOV     $BASE,STREG        ;RELOAD INITIAL ADDRESSES
2733 023046 013737 001244 001424          MOV     $BASE,ADST1
2734 023054 013737 001244 001426          MOV     $BASE,ADBUFF
2735 023062 013737 001240 001430          MOV     $VECT1,VECTOR      ;GET DEFAULT VECTOR
2736 023070 013737 001514 001512          MOV     NMBEXT,NBEXT      ;RESET UNIT COUNTER
2737 023076 005237 001424          BYPASS: INC    ADST1
2738 023102 062737 000002 001426          ADD     #2,ADBUFF
2739 023110 042737 170000 001430          BIC     #170000,VECTOR
2740 023116 013737 001430 001432          MOV     VECTOR,VECTR1
2741 023124 062737 000002 001432          ADD     #2,VECTR1
2742 023132 013737 001430 001434          MOV     VECTOR,VECTR2
2743 023140 062737 000004 001434          ADD     #4,VECTR2
2744 023146 013737 001430 001436          MOV     VECTOR,VECTR3
2745 023154 062737 000006 001436          ADD     #6,VECTR3
2746          ;;LOAD .+2 AND JSR PC,R0 TRAP CATCHER;;
2747 023162 012700 000222          MOV     #222,R0          ;FILL .+2
2748 023166 012701 000220          MOV     #220,R1          ;LOAD JSR PC,R0
2749 023172 010021          1$:  MOV     R0,(R1)+
2750 023174 012721 004700          MOV     #4700,(R1)+
2751 023200 010100          MOV     R1,R0
2752 023202 005720          TST     (R0)+
2753 023204 020027 001002          CMP     R0,#1002
2754 023210 001370          BNE     1$
2755 023212 004737 042342          JSR     PC,WHICHU        ;DETERMINE UNIT #
2756 023216 00G207          RTS     PC          ;TEST NEXT A/D
  
```

```
2758 023220 104416
2759 023222 104401 034024
2760 023226 013746 001510
(1)
(1) 023232 104403
(1) 023234 002
(1) 023235 000
2761 023236 104401 034061
2762 023242 004737 023436
2763 023246 104401 034037
2764 023252 013746 001506
(1)
(1) 023256 104403
(1) 023260 002
(1) 023261 000
2765 023262 104401 034061
2766 023266 013737 001506 023306
2767 023274 012777 000200 156124
2768 023302 004537 025702
2769 023306 000000
2770 023310 013746 001502
(1)
(1) 023314 104403
(1) 023316 004
(1) 023317 001
2771 023320 020437 026764
2772 023324 003003
2773 023326 104401 034135
2774 023332 000207
2775 023334 104401 034611
2776 023340 004737 042334
2777 023344 005237 001112
2778 023350 000207
2779
2780
2781 023352 012737 023434 012330
2782 023360 013737 001510 001516
2783 023366 004537 023540
2784 023372 000062
2785 023374 004737 012304
2786 023400 063702 001532
2787 023404 013737 001506 001516
2788 023412 004537 023540
2789 023416 000062
2790 023420 004737 012304
2791 023424 163702 001532
2792 023430 062716 000002
2793 023434 000207

TYPSET: TYPDC
TYPE ,LSB
MOV CH2,-(SP) ;:SAVE CH2 FOR TYPEOUT
;:TYPE CH
;:GO TYPE--OCTAL ASCII
;:TYPE 2 DIGIT(S)
;:SUPPRESS LEADING ZEROS
;:TYPE ASCIIZ STRING

TYPOS
.BYTE 2
.BYTE 0
TYPE ,ATMSG
JSR PC,TYPEDG
TYPE ,SETCH
MOV CH1,-(SP) ;:SAVE CH1 FOR TYPEOUT
;:TYPE CH
;:GO TYPE--OCTAL ASCII
;:TYPE 2 DIGIT(S)
;:SUPPRESS LEADING ZEROS

TYPOS
.BYTE 2
.BYTE 0
TYPE ,ATMSG
MOV CH1,1$
MOV #200,@ADBUFF
JSR R5,CONVRT
1$:
MOV TEMP,-(SP) ;:SAVE TEMP FOR TYPEOUT
;:TYPE VALUE
;:GO TYPE--OCTAL ASCII
;:TYPE 4 DIGIT(S)
;:TYPE LEADING ZEROS

CMP R4,VSET
BGT ERR
TYPE ,OKMSG
RTS PC
ERR: TYPE ,ERMSG
JSR PC,WHICHV ;:INDICATE BAD UNIT
INC $ERTIL ;:UPDATE ERROR TOTAL
RTS PC

;:SUBROUTINE FOR SETTLING TESTS::
SET1A: MOV #1$,ERRADR ;:SET UP ERROR RECOVERY ADDRESS
MOV CH2,DUMMY ;:LOAD DUMMY
JSR R5,SARSUB ;:DO SAR ROUTINE AT 50%
50.
JSR PC,TSTDAC ;:CHECK VERNIER DAC SETTING
ADD DAC,R2 ;:ADD RESULT TO R2
MOV CH1,DUMMY ;:CHANGE DUMMY VALUE
JSR R5,SARSUB ;:DO SAR ROUTINE AT 50%
50.
JSR PC,TSTDAC ;:CHECK VERNIER DAC SETTING
SUB DAC,R2 ;:SUBTRACT RESULT FROM R2
ADD #2,(SP) ;:BUMP RETURN ADDRESS TO SKIP OVER BRANCH
1$: RTS PC ;:RETURN
```

```
2795  
2796 023436 013703 001536  
2797 023442 010346  
  (1)  
  (1) 023444 104403  
  (1) 023446 004  
  (1) 023447 001  
2798 023450 023727 023500 000001  
2799 023456 001407  
2800 023460 062703 000007  
2801 023464 104401 030231  
2802 023470 010346  
  (1)  
  (1) 023472 104403  
  (1) 023474 004  
  (1) 023475 001  
2803 023476 000207  
2804 023500 000000  
2805  
2806 023502 012700 000222  
2807 023506 012701 000220  
2808 023512 010021  
2809 023514 012721 004700  
2810 023520 010100  
2811 023522 005720  
2812 023524 022700 001002  
2813 023530 001370  
2814 023532 000240  
2815 023534 000240  
2816 023536 000207
```

```
;;SUBROUTINE TO TYPE EDGE VALUES;;  
TYPEDG: MOV EDGE,R3  
        MOV R3,-(SP)      ;;SAVE R3 FOR TYPEOUT  
                                ;;TYPE OCTAL VALUE OF EDGE  
                                ;;GO TYPE--OCTAL ASCII  
                                ;;TYPE 4 DIGIT(S)  
                                ;;TYPE LEADING ZEROS  
        TYPOS  
        .BYTE 4  
        .BYTE 1  
        CMP EDGFLG,#1  
        BEQ RET  
        ADD #7,R3  
        TYPE ,MINUS      ;TYPE ASCII STRING  
        MOV R3,-(SP)      ;;SAVE R3 FOR TYPEOUT  
                                ;;TYPE EDGE VALUE  
                                ;;GO TYPE--OCTAL ASCII  
                                ;;TYPE 4 DIGIT(S)  
                                ;;TYPE LEADING ZEROS  
RET:    RTS PC  
EDGFLG: 0  
;SUBROUTINE TO LOAD VECTOR AREA WITH TRAP CATCHER  
SETINT: MOV #222,R0      ;LOAD UP POINTER  
        MOV #220,R1      ;LOAD ADDRESS  
2$:    MOV R0,(R1)+      ;LOAD POINTER TO NEXT WORD  
        MOV #4700,(R1)+  ;LOAD 'BAD' INSTRUCTION  
        MOV R1,R0        ;LOAD NEW ADDRESS POINTER  
        TST (R0)+        ;BUMP VALUE  
        CMP #1002,R0     ;FINISHED?  
        BNE 2$           ;BR IF NOT  
        NOP  
        NOP  
        RTS PC          ;EXIT
```

```

2818
2819
2820      ;SUBROUTINE TO DO SUCCESSIVE APPROXIMATION ROUTINE
2821      ;CALL=JSR  R5,SARSUB
2822      ;   XXX:XXX=PERCENT
2823      ;RESULT RETURNED IN 'DAC',USES R0,R1,R4
2824 023540 012537 001552  SARSUB: MOV      (R5)+,PERCNT ;GET PERCENT
2825 023544 006337 001552      ASL      PERCNT
2826 023550 006337 001552      ASL      PERCNT
2827 023554 006337 001552      ASL      PERCNT ;RESCALE PERCENT FOR 1600.
2828 023560 006337 001552      ASL      PERCNT ;POINTS PER BURST
2829 023564 012737 000200 001540  MOV      #200,BITPNT ;INITIALIZE BIT POINTER AT MSB
2830 023572 005037 001532      CLR      DAC ;INITIALIZE DAC VALUE
2831 023576 005000      TRY:  CLR      R0
2832 023600 063737 001540 001532  ADD      BITPNT,DAC ;TRY BIT
2833 023606 013777 001532 155612  MOV      DAC,@ADBUFF
2834 023614 012701 003100      MOV      #1600.,R1 ;SET UP FOR 1600. CONVERSIONS
2835 023620 113777 001516 155576  NXTCVT: MOVB   DUMMY,@ADST1 ;PRESET MUX TO DUMMY CHANNEL
2836 023626 012777 001610 155574  MOV      #RETURN,@VECTOR ;RETURN ADDRESS
2837 023634 052777 000101 155560  BIS      #101,@STREG ;CONVERSION ON DUMMY CHANNEL
2838 023642 000001      WAIT   ;WAIT FOR INTERRUPT
2839 023644 017704 155556      MOV      @ADBUFF,R4 ;DUMMY READ
2840 023650 013704 001520      MOV      CHANL,R4
2841 023654 000304      SWAB   R4
2842 023656 052704 000101      BIS      #101,R4 ;INTERRUPT ENABLE START
2843 023662 010477 155534      MOV      R4,@STREG ;JUMP TO CHANNEL + START CONVERT
2844 023666 000001      WAIT   ;WAIT FOR INTERRUPT
2845 023670 027737 155532 001536  CMP      @ADBUFF,EDGE
2846 023676 002001      BGE     2$
2847 023700 005200      INC     R0 ;COUNT RESULTS .LT. EDGE
2848 023702 005301      2$:  DEC     R1
2849 023704 001345      BNE     NXTCVT
2850 023706 020037 001552      CMP     R0,PERCNT
2851 023712 003003      BGT     SHIFT
2852 023714 163737 001540 001532  SUB      BITPNT,DAC ;TAKE THE BIT OUT
2853 023722 006237 001540      SHIFT: ASR     BITPNT
2854 023726 001323      BNE     TRY
2855 023730 000205      RTS     R5
2856
2857      ;ROUTINE TO DELAY IF PROCESSER CAN NOT DO SOB INSTRUCTION
2858
2859 023732 005300      DELAY4: DEC     R0 ;DECREMENT R0, IS IT ZERO?
2860 023734 001376      BNE     DELAY4 ;NO
2861 023736 000002      RTI     ;RETURN
  
```



```

2863      ;;DIFFERENTIAL LINEARITY SUBROUTINE;;
2864      ;'CHA' CONTAINS THE CHANNEL NUMBER
2865 023740 104401 035214  DIFLIN: TYPE      ,MSG20      ;IDENTIFY TEST
2866 023744 004737 042342      JSR      PC,WHICHU    ;DETERMINE UNIT #
2867 023750 013746 001564      MOV      UNITBD,-(SP)
2868 023754 104403      TYPOS      ;TELL OPER. THE #
2869 023756      001      .BYTE      1,0
2870 023760 104401 034116      TYPE      ,CHAN      ;TELL OPER THE CHANNEL NUMBER
2871 023764 013746 017460      MOV      CHA,-(SP)    ;LOAD NUMBER
2872 023770 104403      TYPOS      ;TELL OPER. THE #
2873 023772      002      .BYTE      2,0
2874 023774 104401 001165      TYPE      ,$CRLF
2875 024000 012702 062341      MOV      #62341,R2    ;SET UP RANDOM NUMBER GENERATOR
2876 024004 012704 142315      MOV      #142315,R4
2877 024010 012705 127623      MOV      #127623,R5
2878 024014 012700 045664      MOV      #BUFFER,R0
2879 024020 012701 010000      MOV      #4096.,R1   ;4096 WORDS FOR HISTOGRAM
2880 024024 005020      CLEAR1: CLR      (R0)+ ;CLEAR BUFFER AREA
2881 024026 005301      DEC      R1
2882 024030 001375      BNE      CLEAR1
2883 024032 012700 045044      MOV      #DIST,R0   ;DISTRIBUTION BUFFER POINTER
2884 024036 012701 000310      MOV      #200.,R1   ;200. WORDS FOR DISTRIBUTION
2885 024042 005003      CLR      R3
2886 024044 005037 001554      CLR      OUT
2887 024050 005037 001472      CLR      WIDE
2888 024054 005037 001474      CLR      NARROW
2889 024060 005037 001476      CLR      FIRST
2890 024064 005037 001500      CLR      SKIPST
2891 024070 005020      CLEAR2: CLR      (R0)+ ;CLEAR DISTRIBUTION BUFFER AREA
2892 024072 005301      DEC      R1
2893 024074 001375      BNE      CLEAR2
2894 024076 013700 017460      MOV      CHA,R0     ;CHANNEL 3
2895 024102 000300      SWAB     R0        ;LOAD MUX BITS
2896 024104 052700 000100      BIS      #100,R0
2897 024110 010077 155306      MOV      R0,@STREG
2898 024114 012737 001440 001534      MOV      #800.,DELAY ;NOMINAL STATE WIDTH - 1 LSB
2899 024122 012777 001620 155300      MOV      #RET1,@VECTOR
2900 024130 012701 007776      AGAIN: MOV      #4094.,R1
2901 024134 060402      NEXT1: ADD      R4,R2   ;GENERATE A RANDOM NUMBER
2902 024136 060502      ADD      R5,R2
2903 024140 005502      ADC      R2
2904 024142 010200      MOV      R2,R0     ;PUT RANDOM NUMBER IN R0
2905 024144 042700 177770      BIC      #177770,R0 ;MASK IT TO 3 BITS ONLY
2906 024150 001401      BEQ      CONVR1
2907 024152 077001      DELAY1: SOB     R0,DELAY1 ;STALL TIME
2908 024154 005277 155242      CONVR1: INC     @STREG ;START CONVERSION
2909 024160 000001      WAIT
2910 024162 000240      NOP
2911 024164 017700 155236      MOV      @ADDBUFF,R0 ;GET CONVERTED VALUE
2912 024170 001416      BEQ      LODLY1    ;IGNORE IF =0
2913 024172 020027 007777      CMP      R0,#7777  ;IGNORE IF =7777
2914 024176 001416      BEQ      HIDLY1
2915 024200 006300      ASL     R0
2916 024202 005260 045664      INC     BUFFER(R0) ;MAKE HISTOGRAM
2917 024206 100016      BPL     OKAY1
2918 024210 012760 077777 045664      MOV     #077777,BUFFER(R0) ;PREVENT OVERFLOW
  
```

```

2919 024216 000412
2920 024220 005037 001502
2921 024224 000407
2922 024226 020027 007777
2923 024232 001400
2924 024234 005201
2925 024236 005263 001502
2926 024242 100766
2927 024244 005301
2928 024246 001514
2929 024250 060204
2930 024252 060504
2931 024254 005504
2932 024256 010400
2933 024260 042700 177770
2934 024264 001401
2935 024266 077001
2936 024270 005277 155126
2937 024274 000001
2938 024276 000240
2939 024300 017700 155122
2940 024304 001416
2941 024306 020027 007777
2942 024312 001416
2943 024314 006300
2944 024316 005260 045664
2945 024322 100016
2946 024324 012760 077777 045664
2947 024332 000412
2948 024334 005037 001502
2949 024340 000407
2950 024342 020027 007777
2951 024346 001400
2952 024350 005201
2953 024352 005263 001502
2954 024356 100766
2955 024360 005301
2956 024362 001446
2957 024364 060205
2958 024366 060405
2959 024370 005505
2960 024372 010500
2961 024374 042700 177770
2962 024400 001401
2963 024402 077001
2964 024404 005277 155012
2965 024410 000001
2966 024412 000240
2967 024414 017700 155006
2968 024420 001416
2969 024422 020027 007777
2970 024426 001416
2971 024430 006300
2972 024432 005260 045664
2973 024436 100016
2974 024440 012760 077777 045664

      BR      OKAY1
NOTOK1: CLR    TEMP
      BR      OKAY1
LODLY1: CMP    R0,#7777      ;EQUALIZE LOOP TIME
      BEQ    HIDLY1          ;WITH DUMMY INSTR.
HIDLY1: INC    R1
      INC    TEMP(R3)
      BMI    NOTOK1
OKAY1:  DEC    R1
      BEQ    AROUND
      ADD    R2,R4          ;GENERATE A RANDOM NUMBER
      ADD    R5,R4
      ADC    R4
      MOV    R4,R0          ;PUT RANDOM NUMBER IN R0
      BIC    #177770,R0     ;MASK IT TO 3 BITS ONLY
      BEQ    CONVR2
DELAY2: SOB    R0,DELAY2    ;STALL TIME
CONVR2: INC    @STREG       ;START CONVERSION
      WAIT
      NOP
      MOV    @ADBUFF,R0     ;GET CONVERTED VALUE
      BEQ    LODLY2          ;IGNORE IF =0
      CMP    R0,#7777      ;IGNORE IF =7777
      BEQ    HIDLY2
      ASL    R0
      INC    BUFFER(R0)     ;MAKE HISTOGRAM
      BPL    OKAY2
      MOV    #077777,BUFFER(R0) ;PREVENT OVERFLOW
      BR      OKAY2
NOTOK2: CLR    TEMP
      BR      OKAY2
LODLY2: CMP    R0,#7777      ;EQUALIZE LOOP TIME
      BEQ    HIDLY2          ;WITH DUMMY INSTR.
HIDLY2: INC    R1
      INC    TEMP(R3)
      BMI    NOTOK2
OKAY2:  DEC    R1
      BEQ    AROUND
      ADD    R2,R5          ;GENFRATE A RANDOM NUMBER
      ADD    R4,R5
      ADC    R5
      MOV    R5,R0          ;PUT RANDOM NUMBER IN R0
      BIC    #177770,R0     ;MASK IT TO 3 BITS ONLY
      BEQ    CONVR3
DELAY3: SOB    R0,DELAY3    ;STALL TIME
CONVR3: INC    @STREG       ;START CONVERSION
      WAIT
      NOP
      MOV    @ADBUFF,R0     ;GET CONVERTED VALUE
      BEQ    LODLY3          ;IGNORE IF =0
      CMP    R0,#7777      ;IGNORE IF =7777
      BEQ    HIDLY3
      ASL    R0
      INC    BUFFER(R0)     ;MAKE HISTOGRAM
      BPL    OKAY3
      MOV    #077777,BUFFER(R0) ;PREVENT OVERFLOW
  
```

2975	024446	000412		BR	OKAY3	
2976	024450	005037	001502	NOTOK3: CLR	TEMP	
2977	024454	000407		BR	OKAY3	
2978	024456	020027	007777	LODLY3: CMP	RO,#7777	;EQUALIZE LOOP TIME
2979	024462	001400		BEQ	HIDLY3	;WITH DUMMY INSTR.
2980	024464	005201		HIDLY3: INC	R1	
2981	024466	005263	001502	INC	TEMP(R3)	
2982	024472	100766		BMI	NOTOK3	
2983	024474	005301		OKAY3: DEC	R1	
2984	024476	001216		BNE	NEXT1	
2985	024500	005337	001534	AROUND: DEC	DELAY	
2986	024504	001211		BNE	AGAIN	
2987				;TAKE THE CONTENTS OF THE ACQUIRED DATA BUFFER AND TEST IF WITHIN CERTAIN LIMITS		
2988				;AND CREATE A STATE DISTRIBUTION BUFFER AND SORT THE VALUES INTO 'BINS'		
2989	024506	012700	007776	MOV	#4094.,R0	
2990	024512	012701	045666	MOV	#BUFFER+2,R1	
2991	024516	012102		READ: MOV	(R1)+,R2	;GET STATE WIDTH
2992	024520	006202		ASR	R2	;1 LSB = 800.
2993	024522	006202		ASR	R2	
2994	024524	006202		ASR	R2	
2995	024526	005502		ADC	R2	;1 LSB = 100.
2996	024530	020227	000310	CMP	R2,#200.	;OUT OF RANGE?
2997	024534	002403		BLT	INRNGE	
2998	024536	005237	001554	INC	OUT	;YES - INCREMENT COUNTER
2999	024542	000423		BR	TYPBAD	
3000	024544	006302		INRNGE: ASL	R2	
3001	024546	005262	045044	INC	DIST(R2)	;MAKE STATE WIDTH DISTRIBUTION
3002	024552	006202		ASR	R2	
3003	024554	020227	000062	CMP	R2,#50.	;IS IT 1/2 LSB?
3004	024560	002007		BGE	NOTNAR	
3005	024562	005237	001474	INC	NARROW	
3006	024566	005702		TST	R2	;IS IT A SKIPPED STATE?
3007	024570	001002		BNE	31\$	
3008	024572	005237	001500	INC	SKIPST	
3009	024576	000405		BR	TYPBAD	
3010	024600	020227	000226	NOTNAR: CMP	R2,#150.	;IS IT 1.5 LSB?
3011	024604	003425		BLE	LAST	
3012	024606	005237	001472	INC	WIDE	
3013	024612	005737	001476	TYPBAD: TST	FIRST	
3014	024616	001004		BNE	60\$	
3015	024620	005237	001476	INC	FIRST	
3016	024624	104401	033774	TYPE	,STATE	
3017	024630	010103		60\$: MOV	R1,R3	
3018	024632	162703	045666	SUB	#BUFFER+2,R3	
3019	024636	006203		ASR	R3	
3020	024640	010346		MOV	R3,-(SP)	::SAVE R3 FOR TYPEOUT
(1)						::TYPE STATE
(1)	024642	104403		TYPOS		::GO TYPE--OCTAL ASCII
(1)	024644	004		.BYTE	4	::TYPE 4 DIGIT(S)
(1)	024645	001		.BYTE	1	::TYPE LEADING ZEROS
3021	024646	104401	033770	TYPE	,DASH	
3022	024652	104416		TYPDC		
3023	024654	104401	033761	TYPE	,LSBMSG	
3024	024660	005300		LAST: DEC	RO	
3025	024662	001315		BNE	READ	

N 7

```

3027 :REPORT TO THE OPERATOR THE DIFFERENT STATE VALUES
3028 : IN THE FORM OF A GENERAL STATUS AND INDICATE OK/ERROR
3029 024664 112737 000177 040254 MOVB #177,DECPNT
3030 024672 013702 001500 MOV SKIPST,R2 ;GET NO. OF SKIPPED STATES
3031 024676 104416 TYPDC ;TYPE IT
3032 024700 104401 034626 TYPE ,SKPMSG ;TYPE MESSAGE
3033 024704 005737 001500 TST SKIPST
3034 024710 001407 BEQ 1$
3035 024712 104401 034611 TYPE ,ERMSG ;TYPE 'ERROR'
3036 024716 004737 042334 JSR PC,WHICHV ;INDICATE BAD UNIT
3037 024722 005237 001112 INC $ERTTL ;UPDATE ERROR COUNT
3038 024726 000402 BR NAR
3039 024730 104401 034135 1$: TYPE ,OKMSG ;TYPE #OK#
3040 024734 013702 001474 NAR: MOV NARROW,R2 ;GET NO. OF NARROW STATES
3041 024740 104416 TYPDC ;TYPE IT
3042 024742 104401 034705 TYPE ,NARMSG ;TYPE MESSAGE
3043 024746 013702 001472 MOV WIDE,R2
3044 024752 063702 001554 ADD OUT,R2
3045 024756 104416 TYPDC ;TYPE NO. OF WIDE STATES
3046 024760 104401 034744 TYPE ,WIDMSG ;TYPE MESSAGE
3047 024764 013702 001554 MOV OUT,R2
3048 024770 104416 TYPDC ;TYPE NO. OF STATES OUTSIDE 2 LSB
3049 024772 104401 035003 TYPE ,OUTMSG ;TYPE MESSAGE
3050 024776 005737 001554 TST OUT
3051 025002 001407 BEQ 11$
3052 025004 104401 034611 TYPE ,ERMSG ;TYPE 'ERROR'
3053 025010 004737 042334 JSR PC,WHICHV ;DETERMINE BAD UNIT
3054 025014 005237 001112 INC $ERTTL ;UPDATE ERROR COUNT
3055 025020 000402 BR HALF
3056 025022 104401 034135 11$: TYPE ,OKMSG ;TYPE 'OK'
3057 025026 013702 001474 HALF: MOV NARROW,R2
3058 025032 063702 001472 ADD WIDE,R2
3059 025036 063702 001554 ADD OUT,R2
3060 025042 010200 MOV R2,R0
3061 025044 104416 TYPDC ;TYPE NO. OF STATES OUTSIDE LIMITS
3062 025046 112737 000056 040264 MOVB #56,DECPNT
3063 025054 104401 035036 TYPE ,HAFMSG
3064 025060 020027 000051 CMP RO,#41. ;COMPARE IT TO NOMINAL
3065 025064 003407 BLE 21$
3066 025066 104401 034611 TYPE ,ERMSG ;TYPE 'ERROR'
3067 025072 004737 042334 JSR PC,WHICHV ;INDICATE BAD UNIT
3068 025076 005237 001112 INC $ERTTL ;UPDATE ERROR COUNT
3069 025102 000402 BR SWDIST
3070 025104 104401 034135 21$: TYPE ,OKMSG ;TYPE 'OK'
  
```

3072
3073
3074
3075 025110 005737 001526
3076 025114 001426
3077 025116 004737 025604
3078 025122 104401 035260
3079 025126 104401 035761
3080 025132 012700 045044
3081 025136 012701 000310
3082 025142 012002
3083 025144 004737 026300
3084 025150 005002
3085 025152 004737 026300
3086 025156 005301
3087 025160 001370
3088 025162 104401 035717
3089 025166 004737 025604

; DETERMINE IF VT55 TYPE TERMINAL IS CONNECTED
; IF NOT BYPASS THIS SECTION
; IF VT55/VT105 GRAPHIC TERMINAL REPORT THE DISTRIBUTION CURVE
SWDIST: TST VTFLAG ; BIT MAP TERMINAL AVAILABLE?
BEQ RELACC ; BR IF NOT
JSR PC,DELCLR ; WAIT AWHILE, THEN CLEAR BIT MAP TERMINAL
TYPE ,MSG16
TYPE ,BUFF1 ; TYPE BUFF1-PRINT GRID
MOV #DIST,R0 ; POINTER TO STATE WIDTH DISTRIBUTION
MOV #200.,R1 ; GO 200. TIMES UP TO 2 LSB
NXTY1: MOV (R0)+,R2
JSR PC,LOADY
CLR R2
JSF PC,LOADY
DEC R1
BNE NXTY1
TYPE ,C2 ; TYPE ASCII STRING
JSR PC,DELCLR

```

3091
3092
3093 025172 005001
3094 025174 005003
3095 025176 104401 035605
3096 025202 012700 045666
3097 025206 011002
3098 025210 162702 001440
3099 025214 060201
3100 025216 010120
3101 025220 010104
3102 025222 100001
3103 025224 005404
3104 025226 020403
3105 025230 003405
3106 025232 010403
3107 025234 010005
3108 025236 162705 045666
3109 025242 006205
3110 025244 020027 065662
3111 025250 001356
3112 025252 006203
3113 025254 006203
3114 025256 006203
3115 025260 005503
3116 025262 010302
3117 025264 104416
3118 025266 104401 035632
3119 025272 010546
(1)
(1) 025274 104403
(1) 025276 004
(1) 025277 001
3120 025300 104401 034133
3121 025304 005205
3122 025306 010546
(1)
(1) 025310 104403
(1) 025312 004
(1) 025313 001
3123 025314 020337 026766
3124 025320 003407
3125 025322 104401 034611
3126 025326 004737 042334
3127 025332 005237 001112
3128 025336 000402
3129 025340 104401 034135
3130 025344 005737 001526
3131 025350 001503
3132 025352 012700 045664
3133 025356 012701 010000

;CHANGE HISTOGRAM ERROR TO RELATIVE ACCURACY ERROR
RELACC: CLR R1 ;RUNNING ERROR = 0
CLR R3 ;MAXIMUM ERROR - 0
TYPE ,MSG2i
MOV #BUFFER+2,R0
NXTSTA: MOV (R0),R2 ;STATE WIDTH = R2
SUB #800.,R2 ;STATE WIDTH ERROR IN R2
ADD R2,R1 ;UPDATE RUNNING ERROR
MOV R1,(R0)+ ;SAVE IN BUFFER
MOV R1,R4 ;SAVE IN R4 ALSO
BPL PLUS ;IS IT POSITIVE?
NEG R4 ;NO - MAKE IT POSITIVE
PLUS: CMP R4,R3 ;CHECK AGAINST PREVIOUS MAX. ERROR
BLE NOTNEW ;NOT A NEW MAXIMUM
MOV R4,R3 ;UPDATE MAXIMUM IN R3
MOV R0,R5
SUB #BUFFER+2,R5
ASR R5 ;R5=EDGE VALUE AT MAX. RELACC
NOTNEW: CMP R0,#BUFFER+8190. ;DONE?
BNE NXTSTA ;NO - REPEAT
ASR R3 ;RESCALE FROM 1 LSB = 800. SCALING
ASR R3 ;TO 1 LSB = 100. SCALING
ASR R3
ADC R3
MOV R3,R2
TYPDC
TYPE ,LINEA
MOV R5,-(SP) ;;SAVE R5 FOR TYPEOUT
;;TYPE VALUE
;;GO TYPE--OCTAL ASCII
TYPOS
.BYTE 4 ;;TYPE 4 DIGIT(S)
.BYTE 1 ;;TYPE LEADING ZEROS
TYPE ,SLASH ;PRINT '/'
INC R5
MOV R5,-(SP) ;;SAVE R5 FOR TYPEOUT
;;TYPE VALUE
;;GO TYPE--OCTAL ASCII
TYPOS
.BYTE 4 ;;TYPE 4 DIGIT(S)
.BYTE 1 ;;TYPE LEADING ZEROS
CMP R3,VLIN
BLE 41$
TYPE ,ERMSG
JSR PC,WHICHV ;INDICATE BAD UNIT
INC $ERTTL ;UPDATE ERROR COUNT
BR 42$
41$: TYPE ,OKMSG
42$: TST VTFLAG ;BIT MAP TERMINAL ?
BEQ L02 ;BR IF NOT
MOV #BUFFER,R0
MOV #4096.,R1

```

3135	025362	011002		GETDAT:	MOV	(R0),R2		;GET RELATIVE ACCURACY ERROR SCALED 1LSB - 800.
3136	025364	006202			ASR	R2		;RESCALE IT TO 1 LSB = 100.
3137	025366	006202			ASR	R2		
3138	025370	006202			ASR	R2		
3139	025372	005502			ADC	R2		
3140	025374	062702	000166		ADD	#118.,R2		;AND MOVE IT TO MID-SCREEN
3141	025400	010220			MOV	R2,(R0)+		;PUT IT BACK INTO BUFFER
3142	025402	005301			DEC	R1		
3143	025404	001366			BNE	GETDAT		
3144	025406	012700	045664		MOV	#BUFFER,R0		
3145	025412	012704	045664		MOV	#BUFFER,R4		
3146	025416	012705	045666		MOV	#BUFFER+2,R5		
3147	025422	012701	001000		MOV	#512.,R1		
3148	025426	012702	000007	NXT8:	MOV	#7.,R2		
3149	025432	012003			MOV	(R0)+,R3		
3150	025434	010337	001542		MOV	R3,MIN		;MINIMUM
3151	025440	010337	001550		MOV	R3,MAX		;MAXIMUM
3152	025444	012003		NXTCMP:	MOV	(R0)+,R3		
3153	025446	020337	001542		CMP	R3,MIN		
3154	025452	002002			BGE	MAXTST		
3155	025454	010337	001542		MOV	R3,MIN		;NEW MINIMUM
3156	025460	020337	001550	MAXTST:	CMP	R3,MAX		
3157	025464	003402			BLE	TST8		
3158	025466	010337	001550		MOV	R3,MAX		;NEW MAXIMUM
3159	025472	005302		TST8:	DEC	R2		
3160	025474	001363			BNE	NXTCMP		
3161	025476	013724	001542		MOV	MIN,(R4)+		
3162	025502	013725	001550		MOV	MAX,(R5)+		
3163	025506	022425			CMP	(R4)+,(R5)+		;BUMP EACH ONCE MORE
3164	025510	005301			DEC	R1		
3165	025512	001345			BNE	NXT8		
3166	025514	104401	035154		TYPE	,MSG18		
3167	025520	104401	036007		TYPE	,BUFF2		;TYPE BUFF2
3168	025524	012700	045664		MOV	#BUFFER,R0		
3169	025530	004737	025562		JSR	PC,LOAD		
3170	025534	104401	035725		TYPE	,C3		;TYPE ASCIZ STRING
3171	025540	012700	045666		MOV	#BUFFER+2,R0		
3172	025544	004737	025562		JSR	PC,LOAD		
3173	025550	104401	035717		TYPE	,C2		;TYPE ASCIZ STRING
3174	025554	004737	025604		JSR	PC,DELCLR		
3175	025560	000207		LO2:	RTS	PC		
3176	025562	012701	001000	LOAD:	MOV	#512.,R1		
3177	025566	012002		LOAD0:	MOV	(R0)+,R2		
3178	025570	005720			TST	(R0)+		
3179	025572	004737	026300		JSR	PC,LOADY		
3180	025576	005301			DEC	R1		
3181	025600	001372			BNE	LOAD0		
3182	025602	000207			RTS	PC		

```
3184 025604 032777 002000 153326 DELCLR: BIT      #BIT10,@SWR      ;TEST FOR HALT FOR DISPLAY
3185 025612 001402                BEQ      1$              ;:DON'T HALT FOR DISPLAY
3186 025614 000000                HALT
3187 025616 000407                BR       3$              ;;
3188 025620 005000                1$:   CLR      R0
3189 025622 012701 000020        MOV      #20,R1        ;DELAY BEFORE CLEANING SCREEN
3190 025626 005300                2$:   DEC      R0
3191 025630 001376                BNE     2$
3192 025632 005301                DEC     R1
3193 025634 001374                BNE     2$
3194 025636 104401 036046        3$:   TYPE     ,VTINIT
3195 025642 000207                RTS     PC
3196                ;;TYPE RMS AND PEAK VALUES;;
3197 025644 005702                TYPRP: TST     R2        ;IS NOISE POSITIVE?
3198 025646 100001                BPL     POSNOI        ;YES
3199 025650 005002                CLR     R2            ;R2<0,SET R2-0
3200 025652 104416                POSNOI: TYPDC
3201 025654 104401 035750        TYPE     ,MLSBAT      ;TYPE '' LSB AT ''
3202 025660 004737 023436        JSR     PC,TYPEDG
3203 025664 104401 034116        PSONOI: TYPE     ,CHAN ;TYPE '' ON CHANNEL ''
3204 025670 013746 001520        MOV     (CHANL,-(SP)) ;:SAVE CHANL FOR TYPEOUT
(1)                ;;TYPE CHANL
(1) 025674 104403                TYPOS
(1) 025676 002                .BYTE 2                ;:GO TYPE--OCTAL ASCII
(1) 025677 000                .BYTE 0                ;:TYPE 2 DIGIT(S)
3205 025700 000207                RTS     PC            ;:SUPPRESS LEADING ZEROS-
```



```

3207      ;:ROUTINE TO AVERAGE 8 CONVERSIONS;:
3208 025702 012500      CONVRT: MOV      (R5)+,R0      ;GET CHANNEL VALUE
3209 025704 010037 001520      MOV      R0,CHANL
3210 025710 012777 000200 153510  CONVTC: MOV      #200,@ADBUFF      ;LOAD VERNIER DAC
3211 025716 113700 001520      CONVCD: MOV      CHANL,R0      ;GET CHANNEL
3212 025722 000300      SWAB      R0      ;SET UP A/D STATUS REGISTER
3213 025724 052700 000100      BIS      #100,R0      ;ENABLE INTERRUPTS
3214 025730 010077 153466      MOV      R0,@STREG
3215 025734 012700 010000      MOV      #10000,R0      ;DAC SETTLING DELAY
3216 025740 005300      1$: DEC      R0
3217 025742 001376      BNE      1$
3218 025744 005037 001502      CLR      TEMP
3219 025750 012777 001610 153452      MOV      #RETURN,@VECTOR      ;LOAD VECTOR
3220 025756 012777 000200 153446      MOV      #200,@VECTR      ;SET UP NEW PSW
3221 025764 012700 000010      MOV      #10,R0      ;SET UP COUNTER
3222 025770 005277 153426      2$: INC      @STREG      ;START CONVERSION
3223 025774 000001      WAIT     ;WAIT FOR CONVERSION
3224 025776 067737 153424 001502      ADD      @ADBUFF,TEMP      ;READ BUFFER
3225 026004 005300      DEC      R0
3226 026006 001370      BNE      2$      ;DO 8 TIMES
3227 026010 006237 001502      ASR      TEMP      ;AVERAGE VALUE
3228 026014 006237 001502      ASR      TEMP
3229 026020 006237 001502      ASR      TEMP
3230 026024 005537 001502      ADC      TEMP
3231 026030 000205      RTS      R5      ;RETURN
3232
3233      ;COMPARE $GDDAT AND $BDDAT;:
3234 026032 012537 001124      COMPAR: MOV      (R5)+,$GDDAT      ;GET GOOD DATA
3235 026036 013537 001530      MOV      @(R5)+,SPREAD      ;GET SPREAD
3236 026042 013737 001502 001126      MOV      TEMP,$BDDAT      ;GET BAD(ACTUAL) DATA
3237 026050 013701 001126      COMPRA: MOV      $BDDAT,R1
3238 026054 013700 001124      MOV      $GDDAT,R0
3239 026060 160100      SUB      R1,R0      ;GET DIFFERENCE
3240 026062 100001      BPL      7$
3241 026064 005400      NEG      R0
3242 026066 020037 001530      7$: CMP      R0,SPREAD      ;COMPARE IT TO SPREAD
3243 026072 003001      BGT      10$      ;GO TO ERROR PRINTOUT
3244 026074 005725      TST      (R5)+      ;BUMP RETURN POINTER AROUND ERROR CALL
3245 026076 000205      10$: RTS      R5
  
```

3247
3248 026100 012500
3249 026102 010037 001520
3250 026106 000300
3251 026110 005037 001502
3252 026114 010077 153332
3253 026120 012700 010000
3254 026124 005300
3255 026126 001376
3256 026130 012777 001610 153320
3257 026136 012777 000200 153314
3258 026144 012700 000010
3259 026150 152777 000101 153274 1\$
3260 026156 000001
3261 026160 067737 153270 001502
3262 026166 005300
3263 026170 001367
3264 026172 006237 001502
3265 026176 006237 001502
3266 026202 006237 001502
3267 026206 005537 001502
3268 026212 000205
3269
3270
3271
3272
3273
3274 026214 032703 004000
3275 026220 001003
3276 026222 005403
3277 026224 104401 030231
3278 026230 042703 174000
3279 026234 005002
3280 026236 012701 013424
3281 026242 012700 002000
3282 026246 030003
3283 026250 001401
3284 026252 060102
3285 026254 006201 3\$
3286 026256 006200
3287 026260 001372
3288 026262 006202
3289 026264 006202
3290 026266 005502
3291 026270 104416
3292 026272 104401 032445
3293 026276 000207
3294

```
:::ROUTINE TO AVERAGE 8 CONVERSIONS ON GOOD AD:::
GCONVT: MOV (R5)+,R0 ;GET CHANNEL VALUE
MOV R0,CHANL
SWAB R0
CLR TEMP
MOV R0,@GSTREG ;LOAD CHANNEL INTO MIX BITS
MOV #10000,R0
2$: DEC R0
BNE 2$
MOV #RETURN,@GVECT ;LOAD VECTOR
MOV #200,@GVECT+2 ;SET UP NEW PRIORITY
MOV #10,R0 ;SET UP COUNTER
3$: BISB #101,@GSTREG ;SFT INTRPT. EN., S'AR' CONV.
WAIT ;WAIT FOR CONVERSION
ADD @GADBUF,TEMP ;READ BUFFER
DEC R0
BNE 1$ ;DO 8 TIMES
ASR TEMP ;AVERAGE VALUE
ASR TEMP
ASR TEMP
ADC TEMP
RTS R5 ;RETURN
```

```
:::SUBROUTINE TO CONVERT 2.60 VOLTS TO 15.00 VOLTS:::
:::FUNNY NUMBER CALCULATED BY:
::: (15*2.56/(VOLTAGE))/0.0025
```

```
CONV15: BIT #BIT11,R3 ;IS RESULT MINUS?
BNE 1$ ;:NO
NEG R3 ;YES, MAKE IT PLUS
TYPE ,MINUS ;TYPE '-'
1$: BIC #174000,R3 ;CLEAR UPPER 5 BITS
CLR R2 ;CLEAR RESULT REGISTER
MOV #5908.,R1 ;PUT FUNNY NUMBER INTO R1
MOV #BIT10,R0 ;SETUP TEST BIT
2$: BIT R0,R3 ;MULTIPLY TEMP BY FUNNY NUMBER
BEQ 3$ ;:
ADD R,R2
3$: ASR R1
ASR R0
BNE 2$ ;:NOT FINISHED YET
ASR R2 ;SCALE TO .01 VOLTS / BIT
ASR R2
ADC R2
TYPDC ;TYPE RESULTS
TYPE ,VOLTS ;TYPE 'VOLTS'
RTS PC
```

3296
3297
3298 026300 005702
3299 026302 000001
3300 026304 005002
3301 026306 020227 000353
3302 026312 002402
3303 026314 012702 000353
3304 026320 010203
3305 026322 042702 177740
3306 026326 052702 000040
3307 026332 105777 152612
3308 026336 100375
3309 026340 110277 152606
3310 026344 006203
3311 026346 006203
3312 026350 006203
3313 026352 006203
3314 026354 006203
3315 026356 042703 177770
3316 026362 052703 000040
3317 026366 105777 152556
3318 026372 100375
3319 026374 110377 152552
3320 026400 000207
3321
3322
3323 026402 004737 016552
3324 026406 000005
3325 026410 004737 016552
3326 026414 000207

```
;SUBROUTINE LOADY:
LOADY:  TST      R2                ;ROUTINE TO LOAD VLAUE INTO R2
        BPL     PLUSR2           ;AS A VT55 Y-VALUE
        CLR     R2
PLUSR2:  CMP     R2,#235.
        BLT     LESS
        MOV     #235.,R2
LESS:   MOV     R2,R3
        BIC     #177740,R2
        BIS     #40,R2
B10:    TSTB    @STPS             ;PRINT CHARACTER
        BPL     B10
        MOVB   R2,@STPB
        ASR    R3
        ASR    R3
        ASR    R3
        ASR    R3
        ASR    R3
        BIC    #177770,R3
        BIS    #40,R3
B11:    TSTB    @STPS             ;PRINT CHARACTER
        BPL     B11
        MOVB   R3,@STPB
        RTS    PC

;SUBROUTINE TO DO A BUS RESET
ARESET: JSR     PC,STALL         ;DELAY
        RESET
        JSR     PC,STALL         ;DELAY
        RTS    PC              ;EXIT
```

```

3328      ::SUBROUTINE TO TYPE DECIMAL VALUE;;
3329      ::IN R2 AS X.XX;;
3330 026416 005702      DECTYP: TST      R2          ;TEST VALUE TO BE TYPED
3331 026420 100003      BPL      POS
3332 026422 104401 030231 TYPE      ,MINUS          ;TYPE MINUS SIGN
3333 026426 005402      NEG      R2
3334 026430 020227 023417 POS:      CMP      R2,#9999.    ;>9999. REPLACE IT WITH 9999.
3335 026434 003402      BLE      OKAYD
3336 026436 012702 023417 MOV      #9999.,R2
3337 026442 105037 040266 OKAYD:   CLRB     ONES          ;CLEAR ONES
3338 026446 105037 040265      CLRB     TENS          ;CLEAR TENS
3339 026452 105037 040263      CLRB     HUNS          ;CLEAR HUNS
3340 026456 105037 040262      CLRB     THOUS         ;CLEAR THOUS
3341 026462 005702      TESTR2: TST     R2          ;CONVERT VALUE TO A DECIMAL VALUE
3342 026464 001434      BEQ     TYP0UT
3343 026466 005302      DEC     R2
3344 026470 105237 040266      INCB    ONES
3345 026474 123727 040266 000012      CMPB   ONES,#10.
3346 026502 001367      BNE     TESTR2
3347 026504 105037 040266      CLRB   ONES
3348 026510 105237 040265      INCB   TENS
3349 026514 123727 040265 000012      CMPB   TENS,#10.
3350 026522 001357      BNE     TESTR2
3351 026524 105037 040265      CLRB   TENS
3352 026530 105237 040263      INCB   HUNS
3353 026534 123727 040263 000012      CMPB   HUNS,#10.
3354 026542 001347      BNE     TESTR2      ;;
3355 026544 105037 040263      CLRB   HUNS
3356 026550 105237 040262      INCB   THOUS
3357 026554 000742      BR     TESTR2
3358 026556 152737 000060 040262 TYP0UT: BISB   #60,THOUS      ;PREPARE FOR TYP0UT
3359 026564 152737 000060 040263      BISB   #60,HUNS
3360 026572 152737 000060 040265      BISB   #60,TENS
3361 026600 152737 000060 040266      BISB   #60,ONES
3362 026606 123727 040262 000060      CMPB   THOUS,#60
3363 026614 001403      BEQ     1$      ;;
3364 026616 104401 040262      TYPE   ,THOUS
3365 026622 000002      RTI
3366 026624 104401 040263      1$:    TYPE   ,HUNS      ;TYPE VALUE
3367 026630 000002      RTI
3368      ;SUBROUTINE TO SENSE THE 'WFTEST' FLAG AND USE WIDE/NARROW ERROR TOLERANCES
3369 026632 012701 026726      WFADJ: MOV     #VNR,R1      ;SUBROUTINE TO SET LIMITS
3370 026636 005737 001544      TST     WFTEST        ;RUNNING ON TESTER ?
3371 026642 100403      BMI     1$          ;:YES
3372 026644 012702 026772      MOV     #VARLT1,R2    ;WFTEST NOT MINUS, USE NORMAL LIMITS
3373 026650 000402      BR     2$          ;:
3374 026652 012702 027034      1$:    MOV     #VARLT2,R2 ;WFTEST MINUS, USE OPTION AREA LIMITS
3375 026656 012221      2$:    MOV     (R2)+,(R1)+ ;SET UP LIMITS
3376 026660 005711      TST     (R1)          ;DONE?
3377 026662 100375      BPL     2$          ;:NO
3378 026664 000207      RTS     PC
3379 026666 000000      V0:    0
3380 026670 000002      V2:    2
3381 026672 000012      V10:   10.
3382 026674 000012      V12:   12
3383 026676 000062      V50D:  50.
  
```

3384 026700 000144
3385 026702 000326
3386
3387 026704
3388
3389 026704 005560
3390 026706 002220
3391 026710 004670
3392 026712 003110
3393 026714 007340
3394 026716 000440
3395 026720 006450
3396 026722 001330
3397 026724 100000
3398
3399
3400 026726 000050
3401 026730 000310
3402 026732 000074
3403 026734 000257
3404 026736 000113
3405 026740 000341
3406 026742 000000
3407 026744 000000
3408 026746 000000
3409 026750 000000
3410 026752 000000
3411 026754 000000
3412 026756 000000
3413 026760 000000
3414 026762 000003
3415 026764 000144
3416 026766 000175
3417 026770 100000

V100D: 100.
V326: 326

VTABLE:
;*VOLTAGE TABLE OF EXPECTED VALUES (SINGLE ENDED) <TEST MODULE>
5560 ;+2.2 VOLTS <CH10, 20, 30 ETC>
2220 ; -2.2 VOLTS
4670 ; +1.1 VOLTS
3110 ; -1.1 VOLTS
7340 ; +4.4 VOLTS <CH14, 24, 34 ETC>
0440 ; -4.4 VOLTS
6450 ; +3.3 VOLTS
1330 ; -3.3 VOLTS <CH17, 27, 37 ETC>
BIT15 ; END INDICATOR

VNR: 40. ; RMS NOISE TEST LIMITS FOR MNCAD-MNCAM CHANNELS
VNP: 200. ; PEAK NOISE TEST LIMITS FOR MNCAD-MNCAM CHANNELS
VNRAGO: 60. ; RMS NOISE TEST LIMIT FOR .5 MNCAG CHANNELS
VNPAGO: 175. ; PEAK NOISE TEST LIMIT FOR .5 MNCAG CHANNELS
VNRAG1: 75. ; RMS NOISE TEST LIMIT FOR 5. MNCAG CHANNELS
VNPAG1: 225. ; PEAK NOISE TEST LIMIT FOR 5. MNCAG CHANNELS
VRAG2A: 0 ; MSW OF RMS NOISE TEST LIMIT FOR 50. MNCAG CHANNELS
VRAG2B: 0 ; LSW OF RMS NOISE TEST LIMIT FOR 50. MNCAG CHANNELS
VPAG2A: 0 ; MSW OF PEAK NOISE TEST LIMIT FOR 50. MNCAG CHANNELS
VPAG2B: 0 ; LSW OF PEAK NOISE TEST LIMIT FOR 50. MNCAG CHANNELS
VRAG3A: 0 ; MSW OF RMS NOISE TEST LIMIT FOR 500. MNCAG CHANNELS
VRAG3B: 0 ; LSW OF RMS NOISE TEST LIMIT FOR 500. MNCAG CHANNELS
VPAG3A: 0 ; MSW OF PEAK NOISE TEST LIMIT FOR 500. MNCAG CHANNELS
VPAG3B: 0 ; LSW OF PEAK NOISE TEST LIMIT FOR 500. MNCAG CHANNELS
VCM: 3 ; COMMON MODE TEST LIMIT FOR MNCAG CHANNELS
VSET: 100. ; SETTLING TEST LIMIT FOR MNCAD-MNCAM CHANNELS
VLIN: 125. ; RELATIVE ACCURACY TEST LIMIT
BIT15

```
3419
3420
3421 ;LIMITS FOR NON-TESTER
3422 026772 000050 VARLT1: 40. ;.4 LSB, NORMAL LIMITS FOR SYSTEM
3423 026774 000310 200. ;2 LSB, INTEGRATION AND FIELD USE ON SPEC TESTS
3424 026776 000074 60. :0.60 ;RMS VALUE FOR .5
3425 027000 000257 175. :1.75 ;PEAK VALUE FOR .5
3426 027002 000113 75. :0.75 ;RMS VALUE FOR 5
3427 027004 000341 225. :2.25 ;PEAK VALUE FOR 5
3428 027006 000001 1 :1.33 ;MSW OF RMS VALUE FOR 50
3429 027010 052173 052173 ;LSW OF RMS VALUE FOR 50
3430 027012 000004 4 :4.00 ;MSW OF PEAK VALUE FOR 50
3431 027014 000000 0 ;LSW OF PEAK VALUE FOR 50
3432 027016 000004 4 :4.00 ;MSW OF RMS VALUE FOR 500
3433 027020 000000 0 ;LSW OF RMS VALUE FOR 500
3434 027022 000014 12. :12.00 ;MSW OF PEAK VALUE FOR 500
3435 027024 000000 0 ;LSW OF PEAK VALUE FOR 500
3436 027026 000004 4 :0.04 ;COMMON MODE VALUE
3437 027030 000144 100. ;1 LSB
3438 027032 000175 125. ;1.25 LSB
3439
3440 ;LIMITS FOR TESTER
3441 027034 000041 VARLT2: 33. ;.33 LSB RMS NOISE LIMIT
3442 027036 000226 150. ;1.5 LSB PEAK NOISE LIMIT
3443 027040 000062 50. :0.50 ;RMS VALUE FOR .5
3444 027042 000226 150. :1.50 ;PEAK VALUE FOR .5
3445 027044 000074 60. :0.60 ;RMS VALUE FOR 5
3446 027046 000257 175. :1.75 ;PEAK VALUE FOR 5
3447 027050 000001 1 :1.15 ;MSW OF RMS VALUE FOR 50
3448 027052 023146 023146 ;LSW OF RMS VALUE FOR 50
3449 027054 000003 3 :3.50 ;MSW OF PEAK VALUE FOR 50
3450 027056 100000 100000 ;LSW OF PEAK VALUE FOR 50
3451 027060 000003 3 :3.66 ;MSW OF RMS VALUE FOR 500
3452 027062 124366 124366 ;LSW OF RMS VALUE FOR 500
3453 027064 000013 11. :11.0 ;MSW OF PEAK VALUE FOR 500
3454 027066 000000 0 ;LSW OF PEAK VALUE FOR 500
3455 027070 000003 3 :0.03 ;COMMON MODE VALUE
3456 027072 000132 90. ;.9 LSB INTER-CHANNEL SETTLING LIMIT
3457 027074 000144 100. ;1 LSB RELATIVE ACCURACY ERROR LIMIT
3458
```


: INTERRUPT.
: FOLLOW THE RECOMMENDED PROCEDURE
: IN THE DOCUMENT (ON THIS DIAGNOSTIC)
: FOR LOOPING ON TEST.

```
3532  
3533  
3534  
3535  
3536  
3537  
3538 027344 000002  
3539 027346 022626  
3540 027350 022626  
3541 027352 005737 001176  
3542 027356 001025  
3543 027360 104401 031377  
3544 027364 004737 042342  
3545 027370 013746 001202  
3546 027374 104405  
3547 027376 104401 031423  
3548 027402 013746 001430  
  (1) 027406 104403  
  (1) 027410 003  
  (1) 027411 001  
3549 027412 104401 031454  
3550 027416 013746 027556  
  (1) 027422 104403  
  (1) 027424 003  
  (1) 027425 001  
3551 027426 104401 031504  
3552 027432 013777 001432 151770 4$:  
3553 027440 013777 001436 151766  
3554 027446 012777 004700 151756  
3555 027454 012777 004700 151754  
3556 027462 013737 027556 001430  
3557 027470 042737 000003 001430  
3558 027476 013737 001430 001432  
3559 027504 062737 000002 001432  
3560 027512 013737 001430 001434  
3561 027520 062737 000004 001434  
3562 027526 013737 001434 001436  
3563 027534 062737 000002 001436  
3564 027542 005077 151654  
3565 027546 005777 151654  
3566 027552 000177 151330  
3567 027556 000000  
3568 027560 000000
```

RTI
3\$: CMP (SP)+,(SP)+ :POP OFF JSR TRAP
CMP (SP)+,(SP)+ :POP OFF WRONG INTR.
TST \$PASS :IS THIS THE FIRST PASS?
BNE 4\$:NO, DON'T REPORT
TYPE ,VTMSG :TYPE 'EXPECTED INTR. AT ''
JSR PC,WHICHU :DETERMINE THE UNIT #
MOV \$UNIT,-(SP)
TYPDS
TYPE ,VTMSG3 :REPORT INTR. TO
MOV VECTOR,-(SP) :SAVE VECTOR FOR TYPEOUT
TYPOS :GO TYPE--OCTAL ASCII
.BYTE 3 :TYPE 3 DIGIT(S)
.BYTE 1 :TYPE LEADING ZEROS
TYPE ,VTMSG1 :TYPE '' RECEIVED INTR. AT ''
MOV TRTO,-(SP) :SAVE TRTO FOR TYPEOUT
TYPOS :GO TYPE--OCTAL ASCII
.BYTE 3 :TYPE 3 DIGIT(S)
.BYTE 1 :TYPE LEADING ZEROS
TYPE ,VTMSG2 :TYPE 'RESTARTING TEST'
MOV VECTR1,@VECTOR
MOV VECTR3,@VECTR2
MOV #4700,@VECTR1
MOV #4700,@VECTR3
MOV TRTO,VECTOR
BIC #3,VECTOR
MOV VECTOR,VECTR1
ADD #2,VECTR1
MOV VECTOR,VECTR2
ADD #4,VECTR2
MOV VECTR2,VECTR3
ADD #2,VECTR3
CLR @STREG
TST @ADBUFF :READ A/D BUFFER TO CLEAR DONE FLAG
JMP @SLPADR :START TEST OVER AGAIN.
TRTO: .WORD 0 :CONTAINS ADDR. WE TRAPPED OR INTERRUPTED TO.
TRFRO: .WORD 0 :CONTAINS ADDR. WE TRAPPED OR INTR. FROM.

```
3570 .SBTTL ASCII MESSAGES
3571 .NLIST BEX
3572 027562 051600 040524 052122 SCHAN: .ASCIZ <200>\STARTING ON CHANNEL = \
3573 027612 042600 042116 047111 ECHAN: .ASCIZ <200>\ENDING ON CHANNEL = \
3574 027640 005015 047516 051511 NOIMSG: .ASCIZ <15><12>/NOISE TEST ON UNIT # /
3575 027670 005015 042523 052124 SETMSG: .ASCIZ <15><12>/SETTLING TEST ON UNIT # /
3576 027723 200 043117 051506 OFSET: .ASCIZ <200>/OFFSET TEST ON UNIT # /
3577 027753 111 020123 044124 DWRFAD: .ASCIZ \IS THE MNCAD (A/D) TEST MODULE CONNECTED ? \
3578 030027 111 020123 020101 DWRFAM: .ASCIZ \IS A MNCAM (MUX) TEST MODULE CONNECTED ? \
3579 030101 111 020123 020101 DWRFAG: .ASCIZ \IS A MNCAG (PREAMP) TEST MODULE CONNECTED ? \
3580 030156 051511 052040 042510 DWRFAP: .ASCIZ \IS THE CONSOLE TERMINAL A VT55 OR VT105 ? \
3581 030231 055 000 MINUS: .BYTE 55,0
3582 030233 040 077 040 QUEST: .BYTE 40,77,40,0
3583 030237 040 020055 000 MDASH: .ASCIZ / - /
3584 030243 200 047125 047113 IDONTK: .ASCIZ <200>\UNKNOWN TYPE OF CHANNEL DETECTED - CHECK MNCAG FRONT PANEL SWITCHE
3585 030351 200 044103 041505 WOWAGS: .ASCIZ <200>\CHECK SYSTEM CONFIGURATION - TOO MANY MNCAG DETECTED\<200>
3586 030440 044600 020106 047115 RMPTXT: .ASCII <200>\IF MNCAG CHANNEL - SET MNCAG-TA SWITCH #1, 2, 3 AND 4 TO POSITION
3587 030544 020200 040440 042116 .ASCII <200>\ AND FRONT PANEL SWITCHES TO 'V' AND '100/10' POSITIONS\
3588 030635 200 043111 047040 .ASCIZ <200>\IF NOT, ENSURE SELECTED CHANNELS HAVE THE TEST RAMP CONNECTED\<200>
3589 030735 200 042514 020104 LEDON: .ASCIZ <200>\LED SHOULD BE 'ON'\
3590 030761 200 042514 020104 LEDOFF: .ASCIZ <200>\LED SHOULD BE 'OFF'\
3591 031006 050200 042514 051501 PUSHAG: .ASCII <200>\PLEASE DEPRESS MNCAG-TA SWITCH #\
3592 031047 065 000 AGTASW: .BYTE 65,0
3593 031051 111 020123 020101 SCLOCK: .ASCIZ \IS A MNCCKW (CLOCK) IN THE SYSTEM ? \
3594 031115 040 044523 043516 MSE: .ASCIZ / SINGLE ENDED/<15><12>
3595 031135 040 044504 043106 MDIF: .ASCIZ / DIFFERENTIAL/<15><12>
3596 031155 040 051120 040505 MPRMP: .ASCIZ / PREAMP/<15><12>
3597 031167 040 041524 040440 MTCMP: .ASCIZ / TC AMP/<15><12>
3598 031201 124 050131 020105 GCHAN: .ASCIZ \TYPE IN THE DESIRED 'GAIN OR TC TYPE' REGISTER VALUE (0-17) ? \
3599 031300 046600 041516 042101 MADR: .ASCIZ <200>\MNCAD (A/D) BASE ADDRESS <\
3600 031334 046600 041516 042101 MVCT: .ASCIZ <200>\MNCAD (A/D) VECTOR ADDRESS <\
3601 031372 020076 020077 000 ENCOM: .ASCIZ #> ? #
3602 031377 200 047115 040503 VTMSG: .ASCIZ <200>\MNCAD (A/D) UNIT #\
3603 031423 015 042412 050130 VTMSG3: .ASCIZ <15><12>/EXPECTED INTERRUPT AT /
3604 031454 051040 041505 044505 VTMSG1: .ASCIZ / RECEIVED INTERRUPT AT /
3605 031504 050200 042514 051501 VTMSG2: .ASCII <200>/PLEASE CHECK VECTOR SWITCHES/
3606 031541 015 004412 042522 .ASCIZ <15><12>/ RESTARTING LOGIC TEST/<15><12>
3607 031574 005015 042524 052123 TCHAN: .ASCIZ <15><12>/TEST CHANNELS /
3608 031615 124 050131 020105 YESNO: .ASCIZ /TYPE Y FOR YES, N FOR NO/<15><12>
3609 031650 005015 042524 052123 TSTAD: .ASCIZ <15><12>/TESTING MNCAD/<15><12>
3610 031672 005015 042524 052123 TSTADM: .ASCIZ <15><12>/TESTING MNCAM/<15><12>
3611 031714 052200 051505 044524 TSTAG: .ASCIZ <200>/TESTING MNCAG/<200>
3612 031734 042523 020124 047115 SADTST: .ASCIZ #SET MNCAD (A/D) FRONT PANEL SWITCHES TO 'TEST'#/<15><12>
3613 032015 123 052105 040440 SAGTST: .ASCIZ #SET ALL MNCAG (PREAMP) RANGE SWITCHES TO THE 'P' POSITION#/<200>
3614 032110 005015 042523 020124 SDSE: .ASCIZ <15><12>\SET MNCAD-TA SWITCH TO SINGLE ENDED\<15><12>
3615 032160 005015 042523 020124 SDDIF: .ASCIZ <15><12>\SET MNCAD-TA SWITCH TO DIFFERENTIAL\<15><12>
3616 032230 051600 052105 046440 SDMSE: .ASCIZ <200>\SET MNCAM-TA SWITCH TO SINGLE ENDED\<200>
3617 032276 051600 052105 046440 SDMDIF: .ASCIZ <200>\SET MNCAM-TA SWITCH TO DIFFERENTIAL\<200>
3618 032344 005015 051120 051505 EXTST: .ASCIZ <15><12>\PRESS EXTERNAL START ON MNCAD-TA (A/D) ON UNIT #\
3619 032427 015 025412 032461 TP15: .ASCIZ <15><12>/+15=/
3620 032436 005015 030455 036465 TM15: .ASCIZ <15><12>/-15=/
3621 032445 040 047526 052114 VOLTS: .ASCIZ / VOLTS/
3622 032454 042523 020124 047115 SCM: .ASCIZ /SET MNCAG (PREAMP) MODE SWITCH TO 'MA', /
3623 032525 123 052105 046440 SRM: .ASCIZ /SET MNCAG (PREAMP) MODE SWITCH TO 'K', /
3624 032575 123 052105 046440 SVM: .ASCIZ /SET MNCAG (PREAMP) MODE SWITCH TO 'V', /
3625 032645 200 047117 041440 CHAPOS: .ASCIZ <200>/ON CHANNEL 'A' - /
```

```
3626 032670 047600 020116 044103 CHBPOS: .ASCIZ <200>/ON CHANNEL 'B' - /
3627 032713 200 047117 041440 CHCPOS: .ASCIZ <200>/ON CHANNEL 'C' - /
3628 032736 047600 020116 044103 CHDPOS: .ASCIZ <200>/ON CHANNEL 'D' - /
3629 032761 200 042523 020124 TXTP2: .ASCIZ <200>/SET ALL (PREAMP) TEST MODULE CHANNEL SWITCHES TO POSITION 2/<200>
3630 033057 200 042523 020124 TXTP3: .ASCIZ <200>/SET ALL (PREAMP) TEST MODULE CHANNEL SWITCHES TO POSITION 3/<200>
3631 033155 107 044501 020116 GHLF: .ASCIZ \GAIN TO 100/10\<15><12>
3632 033176 042523 020124 040507 GAINS: .ASCIZ \SET GAIN TO 10/1\<15><12>
3633 033221 123 052105 043440 GAIN50: .ASCIZ \SET GAIN TO 1/.1\<15><12>
3634 033244 042523 020124 040507 GAIN5M: .ASCIZ \SET GAIN TO .1/.01\<15><12>
3635 033271 200 051525 047111 GANP5: .ASCIZ <200>/USING A GAIN OF .5/<200>
3636 033316 052600 044523 043516 GAN5P: .ASCIZ <200>/USING A GAIN OF 5.0/<200>
3637 033343 200 051525 047111 GAN5D: .ASCIZ <200>/USING A GAIN OF 50.0/<200>
3638 033371 200 051525 047111 GAN5T: .ASCIZ <200>/USING A GAIN OF 500.0/<200>
3639 033420 015 012 EXCNOI: .BYTE 15,12
3640 033422 054105 042503 051523 .ASCIZ \EXCESSIVE NOISE ON CHANNEL CAUSED AN\
3641 033467 015 012 ERDIV: .BYTE 15,12
3642 033471 101 044522 044124 .ASCIZ /ARITHMETIC ERROR IN DIVISION - PC= /
3643 033535 015 012 ERMUL: .BYTE 15,12
3644 033537 101 044522 044124 .ASCIZ /ARITHMETIC ERROR IN MULTIPLICATION - PC /
3645 033611 015 012 EROVF: .BYTE 15,12
3646 033613 101 044522 044124 .ASCIZ /ARITHMETIC OVERFLOW ERROR - PC= /
3647 033654 015 012 ERSQR: .BYTE 15,12
3648 033656 051101 052111 046510 .ASCIZ /ARITHMETIC ERROR IN SQUARE A 32 BIT NUMBER - PC /
3649 033740 052200 051505 020124 ENDTST: .ASCIZ <200>/TEST COMPLETED/<200>
3650 033761 040 051514 006502 LSBMSG: .ASCIZ / LSB/<15><12>
3651 033770 026455 000040 DASH: .ASCIZ /-- /
3652 033774 052123 052101 026505 STATE: .ASCIZ /STATE-- WIDTH/<15><12>
3653 034014 044103 000 CH: .ASCIZ /CH/
3654 034017 040 020040 000040 SPACE: .ASCIZ / /
3655 034024 046040 041123 047440 LSB: .ASCIZ / LSB ON CH/
3656 034037 040 042523 052124 SETCH: .ASCIZ / SETTLING FROM CH/
3657 034061 040 052101 000040 ATMSG: .ASCIZ / AT /
3658 034066 046522 020123 047040 RMSNOI: .ASCIZ /RMS NOISE /
3659 034102 042520 045501 047040 PKNOI: .ASCIZ /PEAK NOISE /
3660 034116 047440 020116 044103 CHAN: .ASCIZ / ON CHANNEL /
3661 034133 057 000 SLASH: .ASCIZ /#/
3662 034135 040 020040 047440 OKMSG: .ASCIZ / OK/<15><12>
3663 034146 005015 054524 042520 CCHAN: .ASCIZ <15><12>/TYPE IN OCTAL CHANNEL NUMBER AND DEPRESS 'RETURN': /
3664 034234 005015 054524 042520 SEL: .ASCIZ <15><12>/TYPE 'O' FOR OFFSET, 'G' FOR GAIN & DEPRESS 'RETURN': /
3665 034325 015 040412 045104 XADJ: .ASCII <15><12>/ADJUST R83/
3666 034341 040 047506 020122 MOLSB: .ASCII / FOR 0.00 LSB ERROR/
3667 034364 005015 042504 051120 .ASCIZ <15><12>/DEPRESS 'RETURN' WHEN ADJUSTED/<15><12>
3668 034427 015 044412 050116 IGND: .ASCII <15><12>/INPUT A GROUND ON THE CHANNEL/ ;MUST BE JUST BEFORE 'CRWR'
3669 034466 005015 042504 051120 CRWR: .ASCIZ <15><12>/DEPRESS 'RETURN' WHEN READY/<15><12>
3670 034526 005015 047111 052520 IVOLT: .ASCIZ <15><12>/INPUT +5.115 VOLTS ON THE CHANNEL/
3671 034572 005015 042101 052512 YADJ: .ASCIZ <15><12>/ADJUST R84/
3672 034607 053 000 POSITV: .ASCIZ /+ /
3673 034611 040 025052 051105 ERMSG: .ASCIZ / **ERROR**/<15><12>
3674 034626 051440 044513 050120 SKPMSG: .ASCIZ / SKIPPED STATE(S)/
3675 034650 041600 046517 047515 COMOD1: .ASCIZ <200>/COMMON MODE REJECTION TEST /
3676 034705 040 040516 051122 NARMSG: .ASCIZ # NARROW (< 1/2 LSB) STATE(S)#<15><12>
3677 034744 053440 042111 020105 WIDMSG: .ASCIZ # WIDE (> 1 1/2 LSB) STATE(S)#<15><12>
3678 035003 040 052123 052101 OUTMSG: .ASCIZ / STATE(S) WIDER THAN 2 LSB/
3679 035036 051440 040524 042524 HAFMSG: .ASCIZ # STATE-WIDTH(S) OUTSIDE + OR - 1/2 LSB#
3680 035105 200 051120 043517 FOUND1: .ASCIZ <200>/PROGRAM DETECTED /
3681 035130 046440 041516 042101 FOUND2: .ASCIZ \ MNCAD (A/D)'S \<15><12>
```

```

3682 035154 005012 025412 027461 MSG18: .ASCII <12><12><12>#+1/2 LSB<15><12><12><12><12><12><12><12><12><12><12><12><12><1
3683 035204 030455 031057 051514 .ASCIIZ \-1/2LSB\
3684 035214 005015 044504 043106 MSG20: .ASCIIZ <15><12>/DIFFERENTIAL LINEARITY ON UNIT # /
3685 035260 020040 020040 020040 MSG16: .ASCII / STATE-WIDTH DISTRIBUTION/<15><12><12><12>
3686 035340 020040 020043 043117 .ASCII / # OF STATES/<12><12><12><12><12><12><12><12><12><12><12><12><12><12><12><12><12><
3687 035400 020040 020040 020040 .ASCII / STATE WIDTH (LSB)/<15>
3688 035504 030040 020040 020040 .ASCIIZ # 0 1/2 1 1 1/2 2#
3689 035605 122 046105 052101 MSG21: .ASCIIZ /RELATIVE ACCURACY:/<15><12>
3690 035632 046040 041123 046440 LINEA: .ASCIIZ / LSB MAXIMUM AT /
3691 035653 040 052073 052117 ERRTOT: .ASCIIZ / ;TOTAL ERROR COUNT = /
3692 035702 035440 040502 020104 MESGD: .ASCIIZ / ;BAD UNITS /
3693 035717 033 015462 005110 C2: .ASCIIZ <33><62><33><110><12> ;EXIT GRAPH MODE, HOME AND LINE FEED
3694 035725 112 000 C3: .ASCIIZ <112> ;ASCII 'I'
3695 035727 015 047412 043106 MOFSET: .ASCIIZ <15><12>/OFFSET =/
3696 035742 046040 041123 000040 MLSB: .ASCIIZ / LSB /
3697 035750 046040 041123 040440 MLSBAT: .ASCIIZ / LSB AT /
3698 ;CODE TO SETUP BIT MAP VIEWING SCREEN
3699 035761 033 061 BUFF1: .BYTE 33,61 ;GRAPH ON
3700 035763 101 061 .BYTE 101,61 ;ENABLE HISTOGRAM 0
3701 035765 111 062 .BYTE 111,62 ;CLEAR DATA + ENABLE VERT LINES
3702 035767 114 041 060 .BYTE 114,41,60 ;LOAD VERT LINE CORD
3703 035772 045 063 .BYTE 45,63
3704 035774 051 066 .BYTE 51,66
3705 035776 055 071 .BYTE 55,71
3706 036000 061 074 .BYTE 61,74
3707 036002 110 041 040 .BYTE 110,41,40 ;LOAD STARTING CORD.
3708 036005 112 000 .BYTE 112,0 ;LOAD GRAPH 1 COMMAND <DATA TO FOLLOW>
3709 036007 033 061 BUFF2: .BYTE 33,61 ;GRAPH ON
3710 036011 101 047 .BYTE 101,47 ;ENABLE GRAPH 0 AND 1
3711 036013 111 061 .BYTE 111,61 ;ENABLE DISPLAY
3712 036015 104 050 065 .BYTE 104,50,65,44,62 ;LOAD HORIZ CORDINATES
3713 036022 110 040 040 .BYTE 110,40,40 ;LOAD STARTING GRAPH CORD.
3714 036025 102 000 .BYTE 102,0 ;LOAD GRAPH 0 <DATA TO FOLLOW>
3715 036027 033 061 INITVT: .BYTE 33,61 ;GRAPH ON
3716 036031 101 040 040 .BYTE 101,40,40 ;DISABLE SCREEN
3717 036034 111 060 040 .BYTE 111,60,40 ;SET RECTANGEL ASPECT RATIO
3718 036037 033 062 .BYTE 33,62 ;EXIT GRAPH MODE
3719 ; .BYTE 33,133,77,62,105 ;ENSURE 'ASCII' <CAUSES HOLD SCREEN ON VT55>
3720 036041 033 110 .BYTE 33,110 ;'HOME'
3721 036043 033 112 000 .BYTE 33,112,0 ;'ERASE SCREEN'
3722 036046 033 110 VTINIT: .BYTE 33,110 ;'HOME'
3723 036050 033 112 .BYTE 33,112 ;'ERASE SCREEN'
3724 036052 033 061 .BYTE 33,61 ;ENTER GRAPHIC MODE
3725 036054 101 040 .BYTE 101,40 ;CLEAR GRAPH DATA
3726 036056 033 062 000 .BYTE 33,62,0 ;EXIT GRAPHIC MODE
3727
3728
3729 .SBTTL ASCII TEXT MESSAGES
3730
3731 036061 200 020114 020075 PRIME1: .ASCII <200>/L = LOGIC TESTS/
3732 036101 200 020127 020075 .ASCII <200>/W = WRAPAROUND ANALOG TESTS/
3733 036135 200 020101 020075 .ASCII <200>/A = AUTO TESTS/
3734 036154 047200 036440 047040 .ASCII <200>/N = NOISE TESTS ON SELECTED CHANNELS/
3735 036221 200 020104 020075 .ASCII <200>/D = DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY/
3736 036302 046600 036440 041440 .ASCII <200>/M - COMMON MODE REJECTION TEST FOR MNCAG CHANNELS/
3737 036364 051600 036440 051440 .ASCII <200>/S - SETTLING TEST ON SELECTED CHANNELS/
  
```

```
3738 036433 200 020120 020075 .ASCII <200>/P = PRINT CONVERTED ANALOG VALUE LOOP/
3739 036501 200 020103 020075 .ASCII <200>/C = CALIBRATION LOOP FOR MNCAD/
3740 036540 043200 036440 043040 .ASCII <200>/F = FUNCTION TEST OF THE MNCAG FRONT PANEL/
3741 036613 200 020124 020075 .ASCII <200>/T = TEST MNCAG CHANNEL ANALOG INPUT/
3742 036657 200 020102 020075 .ASCII <200>/B = BASE AND VECTOR ADDRESS CHANGES/
3743 036723 200 020107 020075 .ASCII <200>/G = GET NEW SWITCH REGISTER VALUE/
3744 036765 200 020110 020075 .ASCII <200>/H = HELP THE OPERATOR AND RETYPE THIS LIST /
3745 037045 015 012 DOT: .BYTE 15,12
3746 037047 124 050131 020105 .ASCII /TYPE THE 'TEST CHARACTER' THEN DEPRESS 'RETURN KEY' /
3747 037135 115 041516 042101 EM1: .ASCII VMNCAD (A/D) STATUS REG. ERROR\
3748 037173 115 041516 042101 EM2: .ASCII VMNCAD (A/D) FAILED TO INTERRUPT\
3749 037233 115 041516 042101 EM3: .ASCII VMNCAD (A/D) UNEXPECTED INTERRUPT\
3750 037274 047115 040503 020104 EM4: .ASCII VMNCAD (A/D) ERROR ON A/D CHANNEL#
3751 037335 115 041516 042101 EM5: .ASCII VMNCAD (A/D) EXISTING MNCAD NOW FAIL'S TO RESPOND\
3752 037416 047115 040503 020104 EM6: .ASCII VMNCAD (A/D) DOES NOT EXIST <BUS ERROR> CHECK ADDRESS SWITCHES\
3753 037514 047111 047503 051122 EM7: .ASCII \INCORRECT I.D. VALUE\
3754 037541 111 041516 051117 EM10: .ASCII \INCORRECT 'MNCAG HOLD' SIGNAL LEVEL\
3755 037605 111 041516 051117 EM11: .ASCII \INCORRECT MNCAG FRONT PANEL SWITCH POSITION\
3756 037661 115 041516 043501 EM12: .ASCII VMNCAG (PREAMP) GAIN REGISTER IN ERROR\
3757 037727 125 044516 004524 DH1: .ASCII /UNIT ERRPC STREG EXPECTED ACTUAL/
3758 037773 125 044516 004524 DH2: .ASCII /UNIT ERRPC STREG CHANNEL NOMINAL TOL. ACTUAL/
3759 040057 125 044516 004524 DH3: .ASCII /UNIT ERRPC STREG ACTUAL/
3760 040113 125 044516 004524 DH5: .ASCII /UNIT ERRPC WERE ARE/
3761 040137 125 044516 004524 DH6: .ASCII /UNIT ERRPC STREG/
3762 040160 051105 050122 004503 DH7: .ASCII /ERRPC ACTUAL EXPECT OR OR/
3763 040216 047125 052111 042411 DH12: .ASCII /UNIT ERRPC STREG CHAN EXPECT ACTUAL/
3764 040262 000 THOUS: .BYTE 0
3765 040263 000 HUNS: .BYTE 0
3766 040264 056 DECPNT: .BYTE 56
3767 040265 000 TENS: .BYTE 0
3768 040266 000 000 ONES: .BYTE 0,0
3769 .EVEN
3770 .LIST BEX
3771
3772 040270 001564 001116 001422 DT1: UNITBD,$ERRPC, STREG, $GDDAT, $BDDAT,0
3773 040276 001124 001126 000000
3774 040304 001564 001116 001422 DT2: UNITBD,$ERRPC,STREG,CHANL,$GDDAT,SPREAD,$BDDAT,0
3775 040312 001520 001124 001530
3776 040320 001126 000000
3777 040324 001564 001116 001422 DT3: UNITBD,$ERRPC,STREG,$BDDAT,0
3778 040332 001126 000000
3779 040336 001564 001116 001202 DT5: UNITBD,$ERRPC,$UNIT,TEMP,0
3780 040344 001502 000000
3781 040350 001564 001116 001422 DT6: UNITBD,$ERRPC,STREG,0
3782 040356 000000
3783 040360 001116 001126 016544 DT7: $ERRPC,$BDDAT,K60,K20,K340,0
3784 040366 016546 016550 000000
3785 040374 001564 001116 001422 DT12: UNITBD,$ERRPC,STREG,CHANL,$GDDAT,$BDDAT,0
3786 040402 001520 001124 001126
3787 040410 000000
3788 040412 000 000 000 DF1: .BYTE 0,0,0,0,0,0,0,0
3789 040415 000 000 000
3790 040420 000 000 000
```

3781
(1)
(2)
(1)
(1) 040422 000000
(1) 040424 000000
(1) 040426 000000
(1) 040430 000040
(1) 040470
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 040470 005037 040422
(1) 040474 012737 040430 040424
(1) 040502 013737 040424 040426
(1) 040510 012737 040540 000060
(1) 040516 012737 000200 000062
(1) 040524 005777 140416
(1) 040530 012777 000100 140406
(1) 040536 000207
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 040540 117746 140402
(1) 040544 042716 177600
(1) 040550 021627 000003
(1) 040554 001007
(1) 040556 104401 041710
(1) 040562 004737 040470
(1) 040566 005726
(1) 040570 000137 001634
(1) 040574 021627 000007
(1) 040600 001004
(1) 040602 022737 000176 001140
(1) 040610 001500
(1)
(1) 040612
(1) 040612 022737 000040 040422
(1) 040620 001004
(1) 040622 104401 041704
(1) 040626 005726
(1) 040630 000451
(1) 040632 021627 000023
(1) 040636 001021
(1) 040640 005077 140300

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

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
:
:*CALL:
:*      JSR      PC,$TKINT
:*      RETURN
:
$TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
        MOV      #$TKQSR,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
        MOV      $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV      #$TKSRV,$TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
        MOV      #200,$TKVEC+2 ;;'SR' LEVEL 4
        TST      @TKKB      ;;CLEAR DONE FLAG
        MOV      #100,@TKS   ;;ENABLE TTY KEYBOARD INTERRUPT
        RTS      PC        ;;RETURN TO CALLER

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (BEG2)
:
$TKSRV: MOVB     @TKB,-(SP)    ;;PICKUP THE CHARACTER
        BIC      #^C(177),(SP) ;;STRIP THE JUNK
        CMP      (SP),#3      ;;IS IT A CONTROL C?
        BNE     1$          ;;BRANCH IF NO
        TYPE     ,SCNTLC      ;;TYPE A CONTROL-C (^C)
        JSR     PC,$TKINT    ;;INIT THE KEYBOARD
        TST     (SP)+        ;;CLEAN UP STACK
        JMP     BEG2        ;;CONTROL C RESTART
1$:     CMP      (SP),#7      ;;IS IT A CONTROL G?
        BNE     2$          ;;BRANCH IF NO
        CMP     #SWREG,SWR    ;;IS SOFT-SWR SELECTED?
        BEQ     6$          ;;GO TO SWR CHANGE
:
2$:     CMP      #32,$TKCNT   ;;IS THE QUEUE FULL?
        BNE     3$          ;;BRANCH IF NO
        TYPE     ,SBELL      ;;RING THE TTY BELL
        TST     (SP)+        ;;CLEAN CHARACTER OFF OF STACK
        BR      5$          ;;EXIT
3$:     CMP      (SP),#23    ;;IS IT A CONTROL-S?
        BNE     32$        ;;BRANCH IF NO
        CLR     @TKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
```

(1)	040644	005726			TST	(SP)+	::CLEAN CHAR OFF STACK
(1)	040646	105777	140272	31\$:	TSTB	@\$TKS	::WAIT FOR A CHAR
(1)	040652	100375			BPL	31\$::LOOP UNTIL ITS THERE
(1)	040654	117746	140266		MOVB	@\$TKB,-(SP)	::GET THE CHARACTER
(1)	040660	042716	177600		BIC	^C177,(SP)	::MAKE IT 7-BIT ASCII
(1)	040664	022627	000021		CMP	(SP)+,#21	::IS IT A CONTROL-Q?
(1)	040670	001366			BNE	31\$::BRANCH IF NO
(1)	040672	012777	000100	140244	MOV	#100,@\$TKS	::REENABLE TTY KEYBOARD INTERRUPTS
(1)	040700	000002			RTI		::RETURN
(1)	040702	005237	040422	32\$:	INC	\$TKCNT	::COUNT THIS CHARACTER
(1)	040706	021627	000140		CMP	(SP),#140	::IS IT UPPER CASE?
(1)	040712	002405			BLT	4\$::BRANCH IF YES
(1)	040714	021627	000175		CMP	(SP),#175	::IS IT A SPECIAL CHAR?
(1)	040720	003002			BGT	4\$::BRANCH IF YES
(1)	040722	042716	000040		BIC	#40,(SP)	::MAKE IT UPPER CASE
(1)	040726	112677	177472	4\$:	MOVB	(SP)+,@\$TKQIN	::AND PUT IT IN QUEUE
(1)	040732	005237	040424		INC	\$TKQIN	::UPDATE THE POINTER
(1)	040736	023727	040424	040470	CMP	\$TKQIN,\$\$TKQEND	::GO OFF THE END?
(1)	040744	007003			BNE	5\$::BRANCH IF NO
(1)	040746	012737	040430	040424	MOV	#\$TKQSRT,\$TKQIN	::RESET THE POINTER
(1)	040754	000002		5\$:	RTI		::RETURN

(2) ::*****
(1) ::*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ::*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ::*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
(1) ::*CALL WHEN OPERATING IN TTY INTERRUPT MODE.

(1)	040756	022737	000176	001140	\$CKSWR:	CMP	#\$SWREG,SWR	::IS THE SOFT-SWR SELECTED
(1)	040764	001124			BNE	15\$::EXIT IF NOT	
(1)	040766	105777	140152		TSTB	@\$TKS	::IS A CHAR WAITING?	
(1)	040772	100121			BPL	15\$::IF NOT, EXIT	
(1)	040774	117746	140146		MOVB	@\$TKB,-(SP)	::YES	
(1)	041000	042716	177600		BIC	^C177,(SP)	::MAKE IT 7-BIT ASCII	
(1)	041004	021627	000007		CMP	(SP),#7	::IS IT A CONTROL-G?	
(1)	041010	001300			BNE	2\$::IF NOT, PUT IT IN THE TTY QUEUE	
(1)							::AND EXIT	

(2) ::*****
(1) ::*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
(1) ::*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
(1) ::*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.

(1)	041012	123727	001134	000001	6\$:	CMPB	\$AUTOE,#1	::ARE WE RUNNING IN AUTO-MODE?
(1)	041020	001674			BEQ	2\$::BRANCH IF YES	
(1)	041022	005726			TST	(SP)+	::CLEAR CONTROL-G OFF STACK	
(1)	041024	004737	040470		JSR	PC,\$TKINT	::FLUSH THE TTY INPUT QUEUE	
(1)	041030	005077	140110		CLR	@\$TKS	::DISABLE TTY KEYBOARD INTERRUPTS	
(1)	041034	112737	000001	001135	MOVB	#1,\$INTAG	::SET INTERRUPT MODE INDICATOR	

(1)	041042	104401	041722		TYPE	,\$CNTLG	::ECHO THE CONTROL-G (^G)
(1)	041046	104401	041727	\$GTSWR:	TYPE	,\$MSWR	::TYPE CURRENT CONTENTS
(2)	041052	013746	000176		MOV	SWREG,-(SP)	::SAVE SWREG FOR TYPEOUT
(2)	041056	104402			TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGIT)
(1)	041060	104401	041740		TYPE	,\$MNEW	::PROMPT FOR NEW SWR
(1)	041064	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
(1)	041066	005046			CLR	-(SP)	::THE NEW SWR
(1)	041070	105777	140050	7\$:	TSTB	@\$TKS	::CHAR THERE?

```

(1) 041074 100375      BPL      7$          ::IF NOT TRY AGAIN
(1)
(1) 041076 117746 140044  MOVB     @STKB, -(SP)  ::PICK UP CHAR
(1) 041102 042716 177600  BIC      #^C177, (SP)  ::MAKE IT 7-BIT ASCII
(1)
(1) 041106 021627 000003  CMP      (SP), #3      ::IS IT A CONTROL-C?
(1) 041112 001015      BNE      9$          ::BRANCH IF NOT
(1) 041114 104401 041710  TYPE     , $CNTLC      ::YES, ECHO CONTROL-C (^C)
(1) 041120 062706 000006  ADD      #6, SP        ::CLEAN UP STACK
(1) 041124 123727 001135 000001  CMPB     $INTAG, #1    ::REENABLE TTY KEYBOARD INTERRUPTS?
(1) 041132 001003      BNE      8$          ::BRANCH IF NO
(1) 041134 012777 000100 140002  MOV      #100, @STKS   ::ALLOW TTY KEYBOARD INTERRUPTS
(1) 041142 000137 001634 8$:     JMP      BEG2        ::CONTROL-C RESTART
(*)
(1)
(1) 041146 021627 000025 9$:     CMP      (SP), #25   ::IS IT A CONTROL-U?
(1) 041152 001005      BNE     10$         ::BRANCH IF NOT
(1) 041154 104401 041715  TYPE     , $CNTLU      ::YES, ECHO CONTROL-U (^U)
(1) 041160 062706 000006 20$:    ADD      #6, SP        ::IGNORE PREVIOUS INPUT
(1) 041164 000737      BR       19$         ::LET'S TRY IT AGAIN
(1)
(1)
(1) 041166 021627 000015 10$:    CMP      (SP), #15    ::IS IT A <CR>?
(1) 041172 001022      BNE     16$         ::BRANCH IF NO
(1) 041174 005766 000004  TST      4(SP)        ::YES, IS IT THE FIRST CHAR?
(1) 041200 001403      BEQ     11$         ::BRANCH IF YES
(1) 041202 016677 000002 137730  MOV      2(SP), @SWR   ::SAVE NEW SWR
(1) 041210 062706 000006 11$:    ADD      #6, SP        ::CLEAN UP STACK
(1) 041214 104401 001165 14$:    TYPE     , $CRLF      ::ECHO <CR> AND <LF>
(1) 041220 123727 001135 000001  CMPB     $INTAG, #1    ::RE-ENABLE TTY KBD INTERRUPTS?
(1) 041226 001003      BNE     15$         ::BRANCH IF NOT
(*) 041230 012777 000100 137706  MOV      #100, @STKS   ::RE-ENABLE TTY KBD INTERRUPTS
(1) 041236 000002      RTI                          ::RETURN
(1) 041240 004737 043372 16$:    JSR      PC, $TYPEC    ::ECHO CHAR
(1) 041244 021627 000060  CMP      (SP), #60    ::CHAR < 0?
(1) 041250 002420      BLT     18$         ::BRANCH IF YES
(1) 041252 021627 000067  CMP      (SP), #67    ::CHAR > 7?
(1) 041256 003015      BGT     18$         ::BRANCH IF YES
(1) 041260 042726 000060  BIC      #60, (SP)+   ::STRIP-OFF ASCII
(1) 041264 005766 000002  TST      2(SP)        ::IS THIS THE FIRST CHAR
(1) 041270 001403      BEQ     17$         ::BRANCH IF YES
(1) 041272 006316      ASL     (SP)         ::NO, SHIFT PRESENT
(1) 041274 006316      ASL     (SP)         ::CHAR OVER TO MAKE
(1) 041276 006316      ASL     (SP)         ::ROOM FOR NEW ONE.
(1) 041300 005266 000002 17$:    INC      2(SP)        ::KEEP COUNT OF CHAR
(1) 041304 056616 177776  BIS      -2(SP), (SP)  ::SET IN NEW CHAR
(1) 041310 000667      BR      7$          ::GET THE NEXT ONE
(1) 041312 104401 001164 18$:    TYPE     , $QUES     ::TYPE ?<CR><LF>
(1) 041316 000720      BR      20$         ::SIMULATE CONTROL-U
(1)
(1)
(1)
(2)
(1)
(1)
(1)

```

```

*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
*      RDCHR          ::GET A CHARACTER FROM THE QUEUE

```



```

(1)          ;*      RETURN HERE          ;; CHARACTER IS ON THE STACK
(1)          ;*          ;; WITH PARITY BIT STRIPPED OFF
(1)          ;*          ;;
(1)          ;*          ;;
(1) 041320 011646 $RDCHR: MOV      (SP),-(SP)  ;; PUSH DOWN THE PC AND
(1) 041322 016666      MOV      4(SP),2(SP)  ;; THE PS
(1) 041330 005066 000004 000002      CLR      4(SP)  ;; GET READY FOR A CHARACTER
(2) 041334 005046      CLR      -(SP)  ;; PUT NEW PS ON STACK
(2) 041336 012746 041344      MOV      #64$,-(SP)  ;; PUT NEW PC ON STACK
(2) 041342 000002      RTI          ;; POP NEW PC AND PS
(2) 041344      64$:
(1) 041344 005737 040422 1$:      TST      $TKCNT  ;; WAIT ON A CHARACTER
(1) 041350 001775      BEQ      1$
(1) 041352 005337 040422      DEC      $TKCNT  ;; DECREMENT THE COUNTER
(1) 041356 117766 177044 000004      MOVB   @$TKQOUT,4(SP)  ;; GET ONE CHARACTER
(1) 041364 005237 040426      INC      $TKQOUT  ;; UPDATE THE POINTER
(1) 041370 023727 040426 040470      CMP      $TKQOUT,#$TKQEND  ;; DID IT GO OFF OF THE END?
(1) 041376 001003      BNE      2$  ;; BRANCH IF NO
(1) 041400 012737 040430 040426      MOV      #$TKQSRRT,$TKQOUT  ;; RESET THE POINTER
(1) 041406 000002      RTI          ;; RETURN
(2)          ;*****
(1)          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)          ;*CALL:
(1)          ;*      RDLIN          ;; INPUT A STRING FROM THE TTY
(1)          ;*      RETURN HERE  ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)          ;*          ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)          ;*          ;;
(1) 041410 010346 $RDLIN: MOV      R3, -(SP)  ;; SAVE R3
(1) 041412 005046      CLR      -(SP)  ;; CLEAR THE RUBOUT KEY
(1) 041414 012703 041644 1$:      MOV      #$TTYIN,R3  ;; GET ADDRESS
(1) 041420 022703 041704 2$:      CMP      #$TTYIN+52.,R3  ;; BUFFER FULL?
(1) 041424 101456      BLOS     4$  ;; BR IF YES
(1) 041426 104411      RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
(1) 041430 112613      MOVB   (SP)+,(R3)  ;; GET CHARACTER
(1) 041432 122713 000177 10$:      CMPB   #177,(R3)  ;; IS IT A RUBOUT
(1) 041436 001022      BNE      5$  ;; BR IF NO
(1) 041440 005716      TST      (SP)  ;; IS THIS THE FIRST RUBOUT?
(1) 041442 001007      BNE      6$  ;; BR IF NO
(1) 041444 112737 000134 041642      MOVB   #' \,9$  ;; TYPE A BACK SLASH
(1) 041452 104401 041642      TYPE   ,9$
(1) 041456 012716 177777      MOV      #-1,(SP)  ;; SET THE RUBOUT KEY
(1) 041462 005303 6$:      DEC      R3  ;; BACKUP BY ONE
(1) 041464 020327 041644      CMP      R3,$$TTYIN  ;; STACK EMPTY?
(1) 041470 103434      BLO     4$  ;; BR IF YES
(1) 041472 111337 041642      MOVB   (R3),9$  ;; SETUP TO TYPEOUT THE DELETED CHAR.
(1) 041476 104401 041642      TYPE   ,9$  ;; GO TYPE
(1) 041502 000746      BR      2$  ;; GO READ ANOTHER CHAR.
(1) 041504 005716 5$:      TST      (SP)  ;; RUBOUT KEY SET?
(1) 041506 001406      BEQ      7$  ;; BR IF NO
(1) 041510 112737 000134 041642      MOVB   #' \,9$  ;; TYPE A BACK SLASH
(1) 041516 104401 041642      TYPE   ,9$
(1) 041522 005016      CLR      (SP)  ;; CLEAR THE RUBOUT KEY
(1) 041524 122713 000025 7$:      CMPB   #25,(R3)  ;; IS CHARACTER A CTRL U?
(1) 041530 001003      BNE      8$  ;; BR IF NO
(1) 041532 104401 041715      TYPE   ,8CNTLL  ;; TYPE A CONTROL 'U'
(1) 041536 000726      BR      1$  ;; GO START OVER
  
```

(1)	041540	122713	000022	8S:	CMPB	#22,(R3)	:: IS CHARACTER A '^R'?
(1)	041544	001011			BNE	3S	:: BRANCH IF NO
(1)	041546	105013			CLRB	(R3)	:: CLEAR THE CHARACTER
(1)	041550	104401	001165		TYPE	,SCLF	:: TYPE A 'CR' & 'LF'
(1)	041554	104401	041644		TYPE	,STTYIN	:: TYPE THE INPUT STRING
(1)	041560	000717			BR	2S	:: GO PICKUP ANOTHER CHACTER
(1)	041562	104401	001164	4S:	TYPE	,SQUES	:: TYPE A '?'
(1)	041566	000712			BR	1S	:: CLEAR THE BUFFER AND LOOP
(1)	041570	111337	041642	3S:	MOVB	(R3),9S	:: ECHO THE CHARACTER
(1)	041574	104401	041642		TYPE	,9S	
(1)	041600	122723	000015		CMPB	#15,(R3)+	:: CHECK FOR RETURN
(1)	041604	001305			BNE	2S	:: LOOP IF NOT RETURN
(1)	041606	105063	177777		CLRB	-1(R3)	:: CLEAR RETURN (THE 15)
(1)	041612	104401	001166		TYPE	,SLF	:: TYPE A LINE FEED
(1)	041616	005726			TST	(SP)+	:: CLEAN RUBOUT KEY FROM THE STACK
(1)	041620	012603			MOV	(SP)+,R3	:: RESTORE R3
(1)	041622	011646			MOV	(SP),-(SP)	:: ADJUST THE STACK AND PUT ADDRESS OF THE
(1)	041624	016666	000004 000002		MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
(1)	041632	012766	041644 000004		MOV	#STTYIN,4(SP)	
(1)	041640	000002			RTI		:: RETURN
(1)	041642	000		9S:	.BYTE	0	:: STORAGE FOR ASCII CHAR. TO TYPE
(1)	041643	000			.BYTE	0	:: TERMINATOR
(1)	041644	000040		\$TTYIN:	.BLKB	32.	:: RESERVE 32. BYTES FOR TTY INPUT
(1)	041704	177607	000377	\$BELL:	.ASCIZ	<207><377><377>	:: CODE FOR BELL
(1)	041710	041536	005015 000	\$CNTLC:	.ASCIZ	/^C/<15><12>	:: CONTROL 'C'
(1)	041715	136	006525 000012	\$CNTLU:	.ASCIZ	/^U/<15><12>	:: CONTROL 'U'
(1)	041722	043536	005015 000	\$CNTLG:	.ASCIZ	/^G/<15><12>	:: CONTROL 'G'
(1)	041727	015	051412 051127	\$MSWR:	.ASCIZ	<15><12>/SWR /	
(1)	041734	036440	000040				
(1)	041740	020040	042516 020127	\$MNEW:	.ASCIZ	/ NEW /	
(1)	041746	020075	000				
(1)	041752				.EVEN		


```

3785  .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)  ;:*****
(1)  ;:THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)  ;:AND LOAD THE TEST NUMBER($STSTM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)  ;:AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)  ;:THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)  ;:*SW14=1  LOOP ON TEST
(1)  ;:*SW11=1  INHIBIT ITERATIONS
(1)  ;:*SW09=1  LOOP ON ERROR
(1)  ;:*SW08=1  LOOP ON TEST IN SWR<7:0>
(1)  ;:*CALL
(1)  ;:*      SCOPE           ;;SCOPE=IOT
(1)
(1)  $SCOPE:
(1)  042054          CKSWR
(1)  042054  104410  040000  137054  1$: BIT #BIT14,@SWR ;;TEST FOR CHANGE IN SOFT-SWR
(1)  042056  032777  BNE $OVER ;;LOOP ON PRESENT TEST?
(1)  042064  001114  :#####START OF CODE FOR THE XOR TESTER##### ;;YES IF SW14=1
(1)  042066  000416  $XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)  042070  013746  MOV @ERRVEC,-(SP) ;;THIS INSTRUCTION TO A 'NOP' (NOP-240)
(1)  042074  012737  MOV #$$,@ERRVEC ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  042102  005737  TST @177060 ;;SET FOR TIMEOUT
(1)  042106  012637  MOV (SP)+,@ERRVEC ;;TIME OUT ON XOR?
(1)  042112  000463  BR $SVLAD ;;RESTORE THE ERROR VECTOR
(1)  042114  022626  5$: CMP (SP)+,(SP)+ ;;GO TO THE NEXT TEST
(1)  042116  012637  MOV (SP)+,@ERRVEC ;;CLEAR THE STACK AFTER A TIME OUT
(1)  042122  000423  BR 7$ ;;RESTORE THE ERROR VECTOR
(1)  042124  6$:;#####END OF CODE FOR THE XOR TESTER##### ;;LOOP ON THE PRESENT TEST
(1)  042124  032777  BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
(1)  042132  001404  BEQ 2$ ;;BR IF NO
(1)  042134  127737  CMPB @SWR,$STSTM ;;ON THE RIGHT TEST? SWR<7.0>
(1)  042142  001465  BEQ $OVER ;;BR IF YES
(1)  042144  105737  2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
(1)  042150  001421  BEQ 3$ ;;BR IF NO
(1)  042152  123737  CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  042160  101015  BHI 3$ ;;BR IF NO
(1)  042162  032777  BIT #BIT09,@SWR ;;LOOP ON ERROR?
(1)  042170  001404  BEQ 4$ ;;BR IF NO
(1)  042172  013737  7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  042200  000446  BR $OVER
(1)  042202  105037  4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
(1)  042206  005037  CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  042212  000415  BR 1$ ;;ESCAPE TO THE NEXT TEST
(1)  042214  032777  3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1)  042222  001011  BNE 1$ ;;BR IF YES
(1)  042224  005737  TST $PASS ;;IF FIRST PASS OF PROGRAM
(1)  042230  001406  BEQ 1$ ;; INHIBIT ITERATIONS
(1)  042232  005237  INC $ICNT ;;INCREMENT ITERATION COUNT
(1)  042236  023737  CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)  042244  002024  BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1)  042246  012737  1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1)  042254  013737  MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1)  042262  105237  $SVLAD: INCB $STSTM ;;COUNT TEST NUMBERS
(1)  042266  113737  MOVB $STSTM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX

```

```
(1) 042274 011637 001106      MOV      (SP), $LPADR      ;; SAVE SCOPE LOOP ADDRESS
(1) 042300 011637 001110      MOV      (SP), $LPERR     ;; SAVE ERROR LOOP ADDRESS
(1) 042304 005037 001162      CLR      $ESCAPE         ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 042310 112737 000001 001115  MOVB     #1, $ERMAX       ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 042316 013777 001102 136616  SOVER:  MOV      $TSTNM, @DISPLAY ;; DISPLAY TEST NUMBER
(1) 042324 013716 001106      MOV      $LPADR, (SP)    ;; FUDGE RETURN ADDRESS
(1) 042330 000002      RTI                     ;; FIXES PS
(1) 042332 003720      SMXCNT: 2000.           ;; MAX. NUMBER OF ITERATIONS
3786 042334 053737 001562 001560  WHICHV: BIS      MASKNM, BADUNT ;; SET CURRENT UNIT INTO BAD FIELD
3787 042342 013737 001562 042372  WHICHU: MOV      MASKNM, 11$   ;; GET CURRENT UNIT
3788 042350 012737 000000 001564  MOV      #0, UNITBD      ;; PRIME THE VALUE
3789 042356 006237 042372 10$:    ASR      11$           ;; CONVERT
3790 042362 001404      BEQ      12$           ;; BR WHEN DONE
3791 042364 005237 001564      INC      UNITBD        ;; BUMP POINTER
3792 042370 000772      BR       10$
3793 042372 000000 11$:    0
3794 042374 000207 12$:    RTS      PC              ;EXIT
3798 .SBTTL  ERROR HANDLER ROUTINE

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO $ERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW09=1      LOOP ON ERROR
*CALL
*          ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
(1) 042376 104410      CKSWR           ;; TEST FOR CHANGE IN SOFT-SWR
(1) 042376 104410      JSR      PC, WHICHV      ;; INDICATE BAD UNIT
(3) 042400 004737 042334 7$:    INCB     $ERFLG        ;; SET THE ERROR FLAG
(1) 042404 105237 001103      BEQ      7$           ;; DON'T LET THE FLAG GO TO ZERO
(1) 042410 001775      MOV      $TSTNM, @DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
(1) 042412 013777 001102 136522  INC      $ERTTL        ;; INC THE ERROR COUNT
(1) 042420 005237 001112      MOV      (SP), $ERRPC    ;; GET ADDRESS OF ERROR INSTRUCTION
(1) 042424 011637 001116      SUB      #2, $ERRPC
(1) 042430 162737 000002 001116  MOVB     @ $ERRPC, $ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
(1) 042436 117737 136454 001114  BIT      #BIT13, @SWR    ;; SKIP TYPEOUT IF SET
(1) 042444 032777 020000 136466  BNE     20$           ;; SKIP TYPEOUTS
(1) 042452 001004      JSR      PC, $ERRTYP     ;; GO TO USER ERROR ROUTINE
(1) 042454 004737 042566      TYPE     , $CRLF
(1) 042460 104401 001165 20$:   CMPB     #APTENV, $ENV   ;; RUNNING IN APT MODE
(1) 042464 122737 000001 001210  BNE     2$           ;; NO, SKIP APT ERROR REPORT
(1) 042472 001007      MOVB     $ITEMB, 21$    ;; SET ITEM NUMBER AS ERROR NUMBER
(1) 042474 113737 001114 042506  JSR      PC, $ATY4      ;; REPORT FATAL ERROR TO APT
(1) 042502 004737 043704 21$:   .BYTE   0
(1) 042506 000      .BYTE   0
(1) 042507 000      BR       22$           ;; APT ERROR LOOP
(1) 042510 000777 136422 22$:   TST     @SWR          ;; HALT ON ERROR
(1) 042512 005777 2$:    BPL     3$           ;; SKIP IF CONTINUE
(1) 042516 100002      HALT
(1) 042520 000000      HALT
(1) 042522 104410      CKSWR           ;; TEST FOR CHANGE IN SOFT-SWR
```

```

(1) 042524 032777 001000 136406 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(1) 042532 001402 BEQ 4$ ;;BR IF NO
(1) 042534 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 042540 005737 001162 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(1) 042544 001402 BEQ 5$ ;;BR IF NONE
(1) 042546 013716 001162 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 042552 5$:
(1) 042552 022737 027164 000042 CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 042560 001001 BNE 6$ ;;BRANCH IF NO
(1) 042562 000000 HALT ;;YES
(1) 042564 6$:
(1) 042564 000002 RTI ;;RETURN
3799 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)
(1)
(1)
(1)
(1)
(1) *****
(1) *THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
(1) *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
(1) *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1) $ERRTYP:
(1) 042566 104401 001165 TYPE , $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 042572 010046 MOV R0,-(SP) ;;SAVE R0
(1) 042574 005000 CLR R0 ;;PICKUP THE ITEM INDEX
(1) 042576 153700 001114 BISB @#$ITEMB,R0
(1) 042602 001004 BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
(2) 042604 013746 001116 MOV $ERRPC,-(SP) ;;TYPE THE PC OF THE ERROR
(2) ;;SAVE $ERRPC FOR TYPEOUT
(2) 042610 104402 TYPOC ;;ERROR ADDRESS
(1) 042612 000445 BR 10$ ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 042614 005300 1$: DEC R0 ;;GET OUT
(1) 042616 006300 ASL R0 ;;ADJUST THE INDEX SO THAT IT WILL
(1) 042620 006300 ASL R0 ;; WORK FOR THE ERROR TABLE
(1) 042622 006300 ASL R0
(1) 042624 062700 001252 ADD # $ERRTB,R0 ;;FORM TABLE POINTER
(1) 042630 012037 042640 MOV (R0)+,2$ ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 042634 001404 BEQ 3$ ;;SKIP TYPEOUT IF NO POINTER
(1) 042636 104401 TYPE ;;TYPE THE 'ERROR MESSAGE'
(1) 042640 000000 2$: .WORD 0 ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 042642 104401 001165 TYPE , $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 042646 012037 042656 3$: MOV (R0)+,4$ ;;PICKUP 'DATA HEADER' POINTER
(1) 042652 001404 BEQ 5$ ;;SKIP TYPEOUT IF 0
(1) 042654 104401 TYPE ;;TYPE THE 'DATA HEADER'
(1) 042656 000000 4$: .WORD 0 ;;'DATA HEADER' POINTER GOES HERE
(1) 042660 104401 001165 TYPE , $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 042664 010146 5$: MOV R1,-(SP) ;;SAVE R1
(1) 042666 012001 MOV (R0)+,R1 ;;PICKUP 'DATA TABLE' POINTER
(1) 042670 001415 BEQ 9$ ;;BR IF NO DATA TO BE TYPED
(1) 042672 012000 MOV (R0)+,R0 ;;PICKUP 'DATA FORMAT' POINTER
(1) 042674 105720 6$: TSTB (R0)+ ;;'OCTAL' OR 'DECIMAL'
(1) 042676 001003 BNE 7$ ;;BR IF DECIMAL
(2) 042700 013146 MOV @ (R1)+,-(SP) ;;SAVE @ (R1)+ FOR TYPEOUT
(2) 042702 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 042704 000402 BR 8$
(1) 042706 7$:
(2) 042706 013146 MOV @ (R1)+,-(SP) ;;SAVE @ (R1)+ FOR TYPEOUT

```

```
(2) 042710 104405  
(1) 042712 005711  
(1) 042714 001403  
(1) 042716 104401 042736  
(1) 042722 000764  
(1)  
(1) 042724 012601  
(1) 042726 012600  
(1) 042730 104401 001165  
(1) 042734 000207  
(1) 042736 020040 000  
(1) 042742 047742  
3800  
(1)  
(2)  
(1)  
(1) 042742 012737 043106 000024  
(1) 042750 012737 000340 000026  
(3) 042756 010046  
(3) 042760 010146  
(3) 042762 010246  
(3) 042764 010346  
(3) 042766 010446  
(3) 042770 010546  
(3) 042772 017746 136142  
(1) 042776 010637 043112  
(1) 043002 012737 043014 000024  
(1) 043010 000000  
(1) 043012 000776  
(1)  
(2)  
(1)  
(1) 043014 012737 043106 000024  
(1) 043022 013706 043112  
(1) 043026 005037 043112  
(1) 043032 005237 043112  
(1) 043036 001375  
(3) 043040 012677 136074  
(3) 043044 012605  
(3) 043046 012604 1  
(3) 043050 012603  
(3) 043052 012602  
(3) 043054 012601  
(3) 043056 012600  
(1) 043060 012737 042742 000024  
(1) 043066 012737 000340 000026  
(1) 043074 104401  
(1) 043076 043114  
(1) 043100 012716  
(1) 043102 001626  
(1) 043104 000002  
(1) 043106 000000  
(1) 043110 000776  
(1) 043112 000000  
3801 043114 051200 051505 040524  
043122 052122 047111 020107
```

```
      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN  
8$:  TST      (R1)      ;;IS THERE ANOTHER NUMBER?  
      BEQ      9$      ;;BR IF NO  
      TYPE     ,11$     ;;TYPE TWO(2) SPACES  
      BR       6$      ;;LOOP  
  
9$:  MOV      (SP)+,R1   ;;RESTORE R1  
10$: MOV      (SP)+,R0  ;;RESTORE R0  
      TYPE     ,$CRLF   ;;'CARRIAGE RETURN' & 'LINE FEED'  
      RTS      PC      ;;RETURN  
11$: .ASCIZ   / /      ;;TWO(2) SPACES  
      .EVEN  
      .SBTTL  POWER DOWN AND UP ROUTINES  
  
*****  
:POWER DOWN ROUTINE  
$PWRDN: MOV     #SILLUP,@PWRVEC ;;SET FOR FAST UP  
        MOV     #340,@PWRVEC+2 ;;PRIO:7  
        MOV     R0,-(SP)      ;;PUSH R0 ON STACK  
        MOV     R1,-(SP)      ;;PUSH R1 ON STACK  
        MOV     R2,-(SP)      ;;PUSH R2 ON STACK  
        MOV     R3,-(SP)      ;;PUSH R3 ON STACK  
        MOV     R4,-(SP)      ;;PUSH R4 ON STACK  
        MOV     R5,-(SP)      ;;PUSH R5 ON STACK  
        MOV     @SWR,-(SP)    ;;PUSH @SWR ON STACK  
        MOV     SP,$SAVR6    ;;SAVE SP  
        MOV     #SPWRUP,@PWRVEC ;;SET UP VECTOR  
        HALT  
        BR      -2          ;;HANG UP  
  
*****  
:POWER UP ROUTINE  
$PWRUP: MOV     #SILLUP,@PWRVEC ;;SET FOR FAST DOWN  
        MOV     $SAVR6,SP    ;;GET SP  
        CLR     $SAVR6      ;;WAIT LOOP FOR THE TTY  
1$:  INC     $SAVR6        ;;WAIT FOR THE INC  
        BNE    1$         ;;OF WORD  
        MOV     (SP)+,@SWR   ;;POP STACK INTO @SWR  
        MOV     (SP)+,R5    ;;POP STACK INTO R5  
        MOV     (SP)+,R4    ;;POP STACK INTO R4  
        MOV     (SP)+,R3    ;;POP STACK INTO R3  
        MOV     (SP)+,R2    ;;POP STACK INTO R2  
        MOV     (SP)+,R1    ;;POP STACK INTO R1  
        MOV     (SP)+,R0    ;;POP STACK INTO R0  
        MOV     #PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR  
        MOV     #340,@PWRVEC+2 ;;PRIO:7  
        TYPE    PWRMSG      ;;REPORT THE POWER FAILURE  
$PWRMG: .WORD   PWRMSG     ;;POWER FAIL MESSAGE POINTER  
        MOV     (PC)+,(SP)  ;;RESTART AT BEGIN  
$PWRAD: .WORD   BEGIN      ;;RESTART ADDRESS  
        RTI  
$ILLUP: HALT              ;;THE POWER UP SEQUENCE WAS STARTED  
        BR      -2          ;;BEFORE THE POWER DOWN WAS COMPLETE  
$SAVR6: 0                ;;PUT THE SP HERE  
PWRMSG: .ASCIZ  <200>/RESTARTING AFTER A POWER FAILURE /
```



```
(1) (1) 043326 105366 000001 7$: DECB 1(SP) ::AND THE NULL CHAR.  
(1) 043332 002770 BLT 6$ ::DOES A NULL NEED TO BE TYPED?  
(1) 043334 004737 043372 JSR PC,$TYPEC ::BR IF NO--GO POP THE NULL OFF OF STACK  
(1) 043340 005337 043436 DECB $CHARCNT ::GO TYPE A NULL  
(1) 043344 000770 BR 7$ ::DO NOT COUNT AS A COUNT  
(1) ::LOOP  
(1) :HORIZONTAL TAB PROCESSOR  
(1) 043346 112716 000040 8$: MOVB #' (SP) ::REPLACE TAB WITH SPACE  
(1) 043352 004737 043372 9$: JSR PC,$TYPEC ::TYPE A SPACE  
(1) 043356 132737 000007 043436 BITB #7,$CHARCNT ::BRANCH IF NOT AT  
(1) 043364 001372 BNE 9$ ::TAB STOP  
(1) 043366 005726 TST (SP)+ ::POP SPACE OFF STACK  
(1) 043370 000724 BR 2$ ::GET NEXT CHARACTER  
(1) 043372 105777 135552 $TYPEC: TSTB @STPS ::WAIT UNTIL PRINTER IS READY  
(1) 043376 100375 BPL $TYPEC  
(1) 043400 116677 000002 135544 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.  
(1) 043406 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?  
(1) 043414 001003 BNE 1$ ::BRANCH IF NO  
(1) 043416 105037 043436 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT  
(1) 043422 000406 BR $TYPEX ::EXIT  
(1) 043424 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?  
(1) 043432 001402 BEQ $TYPEX ::BRANCH IF YES  
(1) 043434 105227 INCB (PC)+ ::COUNT THE CHARACTER  
(1) 043436 000000 $CHARCNT: WORD 0 ::CHARACTER COUNT STORAGE  
(1) 043440 000207 $TYPEX: RTS PC
```

3804 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
(1) (1) ::*****  
(2) ::*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT  
(1) ::*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE  
(1) ::*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED  
(1) ::*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE  
(1) ::*REPLACED WITH SPACES.  
(1) ::*CALL:  
(1) ::* MOV NUM,-(SP) ::PUT THE BINARY NUMBER ON THE STACK  
(1) ::* TYPDS ::GO TO THE ROUTINE  
(1) $TYPDS:  
(1) 043442 MOV R0,-(SP) ::PUSH R0 ON STACK  
(3) 043444 010046 MOV R1,-(SP) ::PUSH R1 ON STACK  
(3) 043446 010246 MOV R2,-(SP) ::PUSH R2 ON STACK  
(3) 043450 010346 MOV R3,-(SP) ::PUSH R3 ON STACK  
(3) 043452 010546 MOV R5,-(SP) ::PUSH R5 ON STACK  
(1) 043454 012746 020200 MOV #20200,-(SP) ::SET BLANK SWITCH AND SIGN  
(1) 043460 016605 000020 MOV 20(SP),R5 ::GET THE INPUT NUMBER  
(1) 043464 100004 BPL 1$ ::BR IF INPUT IS POS.  
(1) 043466 005405 NEG R5 ::MAKE THE BINARY NUMBER POS.  
(1) 043470 112766 000055 000001 MOVB #'-,1(SP) ::MAKE THE ASCII NUMBER NEG.  
(1) 043476 005000 1$: CLR R0 ::ZERO THE CONSTANTS INDEX  
(1) 043500 012703 043656 MOV #SDBLK,R3 ::SETUP THE OUTPUT POINTER  
(1) 043504 112723 000040 MOVB #' ,(R3)+ ::SET THE FIRST CHARACTER TO A BLANK  
(1) 043510 005002 2$: CLR R2 ::CLEAR THE BCD NUMBER  
(1) 043512 016001 043446 MOV $DTBL(R0),R1 ::GET THE CONSTANT
```

```
(1) 043516 160105 3$: SUB R1,R5 ;:FORM THIS BCD DIGIT
(1) 043520 002402 BLT 4$ ;:BR IF DONE
(1) 043522 005202 INC R2 ;:INCREASE THE BCD DIGIT BY 1
(1) 043524 000774 BR 3$
(1) 043526 060105 4$: ADD R1,R5 ;:ADD BACK THE CONSTANT
(1) 043530 005702 TST R2 ;:CHECK IF BCD DIGIT=0
(1) 043532 001002 BNE 5$ ;:FALL THROUGH IF 0
(1) 043534 105716 TSTB (SP) ;:STILL DOING LEADING 0'S?
(1) 043536 100407 BMI 7$ ;:BR IF YES
(1) 043540 106316 5$: ASLB (SP) ;:MSD?
(1) 043542 103003 BCC 6$ ;:BR IF NO
(1) 043544 116663 000001 177777 MOVB 1(SP),-1(R3) ;:YES--SET THE SIGN
(1) 043552 052702 000060 6$: BIS #'0,R2 ;:MAKE THE BCD DIGIT ASCII
(1) 043556 052702 000040 7$: BIS #' ,R2 ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 043562 110223 MOVB R2,(R3)+ ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 043564 005720 TST (R0)+ ;:JUST INCREMENTING
(1) 043566 020027 000010 CMP R0,#10 ;:CHECK THE TABLE INDEX
(1) 043572 002746 BLT 2$ ;:GO DO THE NEXT DIGIT
(1) 043574 003002 BGT 8$ ;:GO TO EXIT
(1) 043576 010502 MOV R5,R2 ;:GET THE LSD
(1) 043600 000764 BR 6$ ;:GO CHANGE TO ASCII
(1) 043602 105726 8$: TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
(1) 043604 100003 BPL 9$ ;:BR IF NO
(1) 043606 116663 177777 177776 9$: MOVB -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
(1) 043614 105013 CLRB (R3) ;:SET THE TERMINATOR
(3) 043616 012605 MOV (SP)+,R5 ;:POP STACK INTO R5
(3) 043620 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
(3) 043622 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
(3) 043624 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
(3) 043626 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
(1) 043630 104401 043656 TYPE $DBLK ;:NOW TYPE THE NUMBER
(1) 043634 016666 000002 000004 MOV 2(SP),4(SP) ;:ADJUST THE STACK
(1) 043642 012616 MOV (SP)+,(SP)
(1) 043644 000002 RTI ;:RETURN TO USER
(1) 043646 023420 $DTBL: 10000.
(1) 043650 001750 1000.
(1) 043652 000144 100.
(1) 043654 000012 10.
(1) 043656 000004 $DBLK: .BLKW 4
3805 .SBTTL APT COMMUNICATIONS ROUTINE
(1)
(2)
(1) 043666 112737 000001 044132 ;:*****
(1) 043674 112737 000001 044130 $ATY1: MOVB #1,$FFLG ;:TO REPORT FATAL ERROR
(1) 043702 000403 $ATY3: MOVB #1,$MFLG ;:TO TYPE A MESSAGE
(1) 043704 112737 000001 044132 $ATY4: BR $ATYC ;:
(1) 043712 $ATYC: MOVB #1,$FFLG ;:TO ONLY REPORT FATAL ERROR
(3) 043712 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
(3) 043714 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
(1) 043716 105737 044130 TSTB $MFLG ;:SHOULD TYPE A MESSAGE?
(1) 043722 001450 BEQ 5$ ;:IF NOT: BR
(1) 043724 122737 000001 001210 CMPB #APTENV,$ENV ;:OPERATING UNDER APT?
(1) 043732 001031 BNE 3$ ;:IF NOT: BR
(1) 043734 132737 000100 001211 BITB #APTPOOL,$ENVM ;:SHOULD SPOOL MESSAGES?
(1) 043742 001425 BEQ 3$ ;:IF NOT: BR
(1) 043744 017600 000004 MOV @4(SP),R0 ;:GET MESSAGE ADDR.
```

```
(1) 043750 062766 000002 000004      ADD      #2,4(SP)          ;;BUMP RETURN ADDR.
(1) 043750 005737 001170      TST      $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1) 043762 001375      BNE      1$              ;;IF NOT: WAIT
(1) 043764 010037 001204      MOV      R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1) 043770 105720      TSTB    (R0)+           ;;FIND END OF MESSAGE
(1) 043772 001376      BNE      2$              ;;
(1) 043774 163700 001204      SUB      $MSGAD,R0       ;;SUB START OF MESSAGE
(1) 044000 006200      ASR      R0              ;;GET MESSAGE LNTH IN WORDS
(1) 044002 010037 001206      MOV      R0,$MSGGLT      ;;PUT LENGTH IN MAILBOX
(1) 044006 012737 000004 001170      MOV      #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1) 044014 000413      BR       5$              ;;
(1) 044016 017637 000004 044042 3$:      MOV      @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1) 044024 062766 000002 000004      ADD      #2,4(SP)          ;;BUMP RETURN ADDRESS
(3) 044032 013746 177776      MOV      177776,-(SP)    ;;PUSH 177776 ON STACK
(1) 044036 004737 043160      JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1) 044042 000000      .WORD   0                ;;
(1) 044044      5$:                      ;;
(1) 044044 105737 044132      10$:      TSTB    $FFLG          ;;SHOULD REPORT FATAL ERROR?
(1) 044050 001416      BEQ     12$              ;;IF NOT: BR
(1) 044052 005737 001210      TST     $ENV             ;;RUNNING UNDER APT?
(1) 044056 001413      BEQ     12$              ;;IF NOT: BR
(1) 044060 005737 001170      11$:      TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1) 044064 001375      BNE     11$              ;;IF NOT: WAIT
(1) 044066 017637 000004 001172      MOV      @4(SP),$FATAL   ;;GET ERROR #
(1) 044074 062766 000002 000004      ADD      #2,4(SP)          ;;BUMP RETURN ADDR.
(1) 044102 005237 001170      INC     $MSGTYPE         ;;TELL APT TO TAKE ERROR
(1) 044106 105037 044132      12$:      CLRB   $FFLG            ;;CLEAR FATAL FLAG
(1) 044112 105037 044131      CLRB   $LFLG            ;;CLEAR LOG FLAG
(1) 044116 105037 044130      CLRB   $MFLG            ;;CLEAR MESSAGE FLAG
(3) 044122 012601      MOV     (SP)+,R1         ;;POP STACK INTO R1
(3) 044124 012600      MOV     (SP)+,R0         ;;POP STACK INTO R0
(1) 044126 000207      RTS     PC               ;;RETURN
(1) 044130      000      $MFLG: .BYTE 0          ;;MESSG. FLAG
(1) 044131      000      $LFLG: .BYTE 0          ;;LOG FLAG
(1) 044132      000      $FFLG: .BYTE 0          ;;FATAL FLAG
(1)      044134      .EVEN
(1)      000200      APTSIZE=200
(1)      000001      APTENV=001
(1)      000100      APTSPool=100
(1)      000040      APTCSUP=040
```



```

(1) 044276 005204          4$: INC R4          ;;DON'T SUPPRESS ANYMORE 0'S
(1) 044300 052703 000060  BIS #'0,R3        ;;MAKE THIS DIGIT ASCII
(1) 044304 052703 000040  5$: BIS #' ,R3        ;;MAKE ASCII IF NOT ALREADY
(1) 044310 110337 044354  MOVB R3,8$        ;;SAVE FOR TYPING
(1) 044314 104401 044354  TYPE ,8$          ;;GO TYPE THIS DIGIT
(1) 044320 105337 044356  7$: DECB $OCNT      ;;COUNT BY 1
(1) 044324 003347          BGT 2$            ;;BR IF MORE TO DO
(1) 044326 002402          BLT 6$            ;;BR IF DONE
(1) 044330 005204          INC R4            ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 044332 000744          BR 2$            ;;GO DO THE LAST DIGIT
(1) 044334 012605 6$: MOV (SP)+,R5        ;;RESTORE R5
(1) 044336 012604          MOV (SP)+,R4        ;;RESTORE R4
(1) 044340 012603          MOV (SP)+,R3        ;;RESTORE R3
(1) 044342 016666 000002 000004  MOV 2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
(1) 044350 012616          MOV (SP)+,(SP)
(1) 044352 000002          RTI                ;;RETURN
(1) 044354 000          8$: .BYTE 0          ;;STORAGE FOR ASCII DIGIT
(1) 044355 000          .BYTE 0          ;;TERMINATOR FOR TYPE ROUTINE
(1) 044356 000          $OCNT: .BYTE 0      ;;OCTAL DIGIT COUNTER
(1) 044357 000          $OFILL: .BYTE 0    ;;ZERO FILL SWITCH
(1) 044360 000000          $OMODE: .WORD 0    ;;NUMBER OF DIGITS TO TYPE
3808 .SBTTL BINARY TO ASCII AND TYPE ROUTINE

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 044362 010146          $TYPBN: MOV R1,-(SP)  ;;SAVE R1 ON THE STACK
(1) 044364 016601 000006  MOV 6(SP),R1        ;;GET THE INPUT NUMBER
(1) 044370 000261          SEC                ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1) 044372 112737 000060 044434  1$: MOVB #'0,$BIN    ;;SET CHARACTER TO AN ASCII '0'.
(1) 044400 006101          ROL R1             ;;GET THIS BIT
(1) 044402 001406          BEQ 2$            ;;DONE?
(1) 044404 105537 044434  ADCB $BIN           ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1) 044410 104401 044434  TYPE , $BIN        ;;GO TYPE THIS BIT
(1) 044414 000241          CLC                ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
(1) 044416 000765          BR 1$             ;;GO DO THE NEXT BIT
(1) 044420 012601          2$: MOV (SP)+,R1    ;;POP THE STACK INTO R1
(1) 044422 016666 000002 000004  MOV 2(SP),4(SP)    ;;ADJUST THE STACK
(1) 044430 012616          MOV (SP)+,(SP)
(1) 044432 000002          RTI                ;;RETURN TO USER
(1) 044434 000 000          $BIN: .BYTE 0,0    ;;STORAGE FOR ASCII CHAR. AND TRMFINATOR
3809 .SBTTL TRAP DECODER

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 044436 010046          $TRAP: MOV R0,-(SP)  ;;SAVE R0
(1) 044440 016600 000002  MOV 2(SP),R0        ;;GET TRAP ADDRESS
(1) 044444 005740          TST -(R0)         ;;BACKUP BY 2

```

```

(1) 044446 111000          MOVB   (R0),R0          ;;GET RIGHT BYTE OF TRAP
(1) 044450 006300          ASL    R0              ;;POSITION FOR INDEXING
(1) 044452 016000 044472   MOV    $TRPAD(R0),R0   ;;INDEX TO TABLE
(1) 044456 000200          RTS    R0              ;;GO TO ROUTINE
(1)
(1)
(1) ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
(1)
(1) 044460 011646          $TRAP2: MOV   (SP),-(SP) ;;MOVE THE PC DOWN
(1) 044462 016666 000004 000002 MOV    4(SP),2(SP)    ;;MOVE THE PSW DOWN
(1) 044470 000002          RTI                    ;;RESTORE THE PSW
(1)
(3) .SBT^L TRAP TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3) ;*BY THE "TRAP" INSTRUCTION.
(3)
(3) : ROUTINE
(3) : -----
(3) $TRPAD: .WORD $TRAP2
(3)          $TYPE  ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
(3)          $TYPOC ;;CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)          $TYPOS ;;CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)          $TYPON ;;CALL=TYPON   TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)          $TYPDS ;;CALL=TYPDS   TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(3)          $TYPBN ;;CALL=TYPBN   TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
(1)
(3) 044510 041046          $GTSWR ;;CALL=GTSWR      TRAP+7(104407) GET SOFT-SWR SETTING
(1)
(3) 044512 040756          $CKSWR ;;CALL=CKSWR      TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
(3) 044514 041320          $RDCHR ;;CALL=RDCHR     TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
(3) 044516 041410          $RDLIN ;;CALL=RDLIN     TRAP+12(104412) TTY TYPEIN STRING ROUTINE
(3) 044520 041752          $RDOCT ;;CALL=RDOCT     TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
3810 044522 011546          TEST   ;;CALL=CHECK    TRAP+14(104414)
3811 044524 011540          TESTIT ;;CALL=CHKIT     TRAP+15(104415)
3812 044526 026416          DECTYP ;;CALL=TYPDC    TRAP+16(104416)
3813 044530 013272          TPRMP  ;;CALL=TESTID   TRAP+17(104417)
3814 044532 023732          DELAY4 ;;CALL=DELY      TRAP+20(104420)
3818
3819 ;BYTE TABLE CONTAINING A TYPE CODE FOR EACH CHANNEL
3820 ;0=NON EXISTANT CHANNEL, 1=SINGLE ENDED, 2=DIFFERENTIAL, 3=MNCAG
3821 ;0XX-DONT TEST ANALOG VALUES, 2XX=TEST ANALOG VALUES
3822 044534 000144          CHTABL: .BLKW 100.      ;CHANNEL TYPE BUFFER
3823
3824 045044 000310          DIST:  .BLKW 200.      ;STATE-WIDTH DISTRIBUTION
3825 045664 010000          BUFFER: .BLKW 4096.    ;BUFFER AREA
3826 065664 000000          BUFEND: 0              ;LAST LOCATION USED BY PROGRAM
3827 000001          .END
  
```


LO2	025560	3131	3175#								
LSB	034024	2759	3655#								
LSBMSG	033761	3023	3650#								
MADR	031300	2690	3599#								
MASKNM	001562	157#	2715*	2717	2731*	3786	3787				
MAX	001550	152#	3151*	3156	3158*	3162					
MAXTST	025460	3154	3156#								
MDASH	030237	2044	2651	3583#							
MDIF	031135	2062	3595#								
MESGD	035702	3469	3692#								
MIN	001542	149#	3150*	3153	3155*	3161					
MINUS	030231	2801	3277	3332	3581#						
MLSB	035742	1354	1903	2299	2340	3696#					
MLSBAT	035750	3201	3697#								
MNCADO	001402	97#	2722	2724							
MOFSET	035727	963	3695#								
MPRMP	031155	2058	3596#								
MSE	031115	2051	2072	3594#							
MSG16	035260	3078	3685#								
MSG18	035154	3166	3682#								
MSG20	035214	2865	3684#								
MSG21	035605	3095	3689#								
MTCMP	031167	3597#									
MTEST	002514	213	222#	305							
MTESTO	002724	219	251	272#							
MTEST1	002730	211	273#	309	333	2706					
MULTI	021366	2190	2465#	2516	2521	2528	2565	2575	2588	2603	
MVCT	031334	2697	3600#								
MOLSB	034341	1417	3666#								
NAR	024734	3038	3040#								
NARMSG	034705	3042	3676#								
NARROW	001474	130#	2888*	3005*	3040	3057					
NBEXT	001512	137#	2713	2727*	2736*						
NEXT1	024134	2901#	2984								
NMBEXT	001514	138#	379*	380*	1854*	1918*	1942*	2736	3467		
NOIJMP	011124	1007	1112#								
NOIMSG	027640	1010	3574#								
NOITST	010406	1006	1010#	1865							
NOITS1	010460	1017	1020	1023#	1107	1869					
NOTNAR	024600	3004	3010#								
NOTNEW	025244	3105	3110#								
NOTOK1	024220	2920#	2926								
NOTOK2	024334	2948#	2954								
NOTOK3	024450	2976#	2982								
NXTCMP	025444	3152#	3160								
NXTCVT	023620	2835#	2849								
NXTSTA	025206	3097#	3111								
NXTY1	025142	3082#	3087								
NXT8	025426	3148#	3165								
OFFERR	010212	968	970#								
OFFOK	010230	969	974#								
OFFSET	012230	962	1301#	1403							
OFSET	027723	955	3576#								
OKAYD	026442	3335	3537#								
OKAY1	024244	2917	2919	2921	2927#						
OKAY2	024360	2945	2947	2949	2955#						

TSETUP	013610	1511	1524	1537	1550	1566#		
TSRT1	014072	1607	1608	1626#				
TSTAD	031650	1979	3609#					
TSTADM	031672	1984	3610#					
TSTAG	031714	1989	3611#					
TSTDAC	012304	1272	1276	1316#	2785	2790		
TSTHLD	014216	1501	1503	1505	1507	1654#		
TSTSDF	012332	663	667	675	679	699	707	1326#
TST1	003536	389#	391					
TST10	004156	465#						
TST11	004172	470#						
TST12	004206	475#						
TST13	004222	479#						
TST14	004260	487#						
TST15	004312	493#						
TST16	004356	501#						
TST17	004420	511#						
TST2	003702	412	417#					
TST20	004450	520#						
TST21	004514	528	530#					
TST22	004572	539	541#					
TST23	004752	568#						
TST24	005126	593#						
TST25	005202	606#						
TST26	005300	621#						
TST27	005364	634#						
TST3	003774	393	395	397	429	435#		
TST30	005444	647#						
TST31	005524	649	660#					
TST32	005566	662	672#					
TST33	005630	674	684#					
TST34	005754	686	688	692	694	713#		
TST35	006160	715	719	725	789#			
TST36	006356	790	792#					
TST37	006554	793	795#					
TST4	004062	447#						
TST40	006752	796	798#					
TST41	007150	799	802#					
TST42	007220	819#						
TST43	007232	824#	825					
TST44	007302	832#						
TST45	007332	839#						
TST46	007362	846#						
TST47	007440	862#						
TST5	004076	451#						
TST50	007470	870#						
TST51	007520	879#						
TST52	007670	911#						
TST53	010034	941#						
TST54	010122	954#						
TST55	010234	973	976#					
TST56	010362	1004#						
TST57	011124	1113#						
TST6	004126	457#						
TST60	011372	1128	1161#					
TST61	011430	1163	1165	1169#				

SSNEW	21#	389	417	435	447	451	457	461	465	470	475	479	487	493	501
	511	520	530	541	568	593	606	621	634	647	660	672	684	713	789
	792	795	798	802	819	824	832	839	846	862	870	879	911	941	954
	976	1004	1113	1161	1169										
SSSET	3809#	3810	3811	3812	3813	3814									
SSSETM	179#														
SSSKIP	21#	412	429	528	539	649	662	674	686	688	692	694	715	719	725
	790	793	796	799	973	1128	1163	1165							
.EQUAT	7#	21													
.HEADE	7#	20													
.SETUP	9#	60													
.SWRHI	9#	22													
.SWRLO	22#														
.SACT1	10#	56													
.SAPT8	10#	59#													
.SAPTH	10#	58													
.SAPTY	10#	3805													
.SCATC	7#														
.SCMTA	7#	59													
.SEOP	7#	3460													
.SERRO	7#	3798													
.SERRT	9#	3799													
.SPARM	8#														
.SPOWE	8#	3800													
.SRAND	10#														
.SRDDE	7#														
.SRDOC	10#	3783													
.SREAD	8#	3781													
.SSAVE	8#														
.SSCOP	8#	3785													
.SSPAC	9#														
.SSWDO	9#														
.STRAP	9#	3809													
.STYPB	8#	3808													
.STYPD	10#	3804													
.STYPE	9#	3803													
.STYPC	8#	3807													

. ABS. 065666 000 OVR RO REL LCL D

ERRORS DETECTED: 0
 CVMNAB,CVMNAB/CRF=CVMNAB
 RUN-TIME: 32 21 2 SECONDS
 RUN-TIME RATIO: 196/57-3.4
 CORE USED: 28K (55 PAGES)