

LPA11

LPA/AD11-K TEST  
CRLPKCO

AH-B050C-MC  
FICHE 1 OF 1

FEB 1981  
COPYRIGHT © 76-80  
MADE IN USA



The table contains a grid of approximately 10 columns and 15 rows of data. The text is extremely faint and illegible, appearing as a series of vertical lines and small clusters of characters. The data is organized in a structured, tabular format, likely representing test results or system parameters.



IDENTIFICATION

B 1

SEQ 0001

Product Code: AC-B049C-MC  
Diagnostic code: MAINDEC-11-CRLPK-C  
Product Name: CRLPKC0 LPA/AD11-K Test  
Revised: DEC. 1980  
Maintainer: Diagnostic Group

Copyright (C) 1976, 1977, 1979, 1980  
Digital Equipment corporation, Maynard, Mass.

This software is furnished under a license for use only on a single computer system and may be copied only with the inclusion of the above copyright notice. This software, or any other copies thereof, may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title and ownership of the software shall at all times remain in dec.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software in equipment which is not supplied by DEC.



## 1.0 ABSTRACT

-----

This diagnostic has two starting addresses: 200 for standard tolerances and 210 for tighter option test area tolerances.

This diagnostic tests the AD11K with or without a wraparound module (G5036).

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

- W: The entire Wraparound test (requires G5036 module)
  - a. Analog subtests
  - b. Noise test
  - c. Interchannel Settling test
  - d. Differential Linearity and Relative Accuracy test
- C: Calibration test only
- N: Noise test only
- S: Interchannel Settling only
- L: Logic Subtests only
- A: Auto test (requires G5036 module)
  - A. Logic subtests
  - B. Analog subtests
  - C. Noise Test
  - D. Interchannel Settling Test
  - E. Differential Linearity and Relative Accuracy Test

THIS PROGRAM IS A MODIFIED VERSION OF 'MD-11-DZADL-B' IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE AD 11K OPTION WHEN IT IS ON THE LPA11-KX I/O BUS. NO RECABLING IS NEEDED. SOME TEST DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION TEST, WERE DELETED AS THEY COULD NOT BE CHECKED. IF THIS DIAGNOSTIC DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO RUN 'MD-11-DZADL-B' YOU SHOULD RUN 'MD-11-CRLPA' BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 10.

## 2.0 REQUIREMENTS

-----

### 2.1 Equipment

PDP-11 family computer with 8K of memory  
Console terminal  
AD11K Module installed in an LPA-11  
Bit-map terminal <OPTIONAL>  
G5036 Wraparound Module



## 2.2 Storage

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

## 3.0 LOADING PROCEDURE

Procedure for loading normal binary tapes should be followed.

## 4.0 STARTING PROCEDURE

### 4.1 Control Switch Settings

Standard PDP-11 Format

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

200 is the starting address of the diagnostic for standard tolerances. 204 is the restart address. 210 is the starting address of the diagnostic for the option test area's tighter tolerances. Starting address of the USER LINK loop is at 214.

## 5.0 OPERATING PROCEDURE

Start the diagnostic at 200 or 210. The program heading and the list of tests available, will be printed out followed by a message "Type the letter and carriage return for the desired test:". Then type the letter you want, according to the table listed and hit carriage return.



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

For machines without a hardware switch register, location SWREG (176) is used as a software switch register. To modify the contents of SWREG, type ^G. The program responds with the current contents of SWREG and a slash. Type the desired new contents of SWREG followed by a carriage return.

If 'W' is typed, the program will type 'xx AD11K's FOUND'. Where xx is the number of AD11K's in octal. If the number is greater than 1, the test will be run successively on each AD11K. The program will run through the logic subtests, the Noise test on 8 edges, the Interchannel Settling test on 8 edges, and the Differential Linearity and Relative Accuracy test. A G5036 wraparound module is required. The program supports AD11K expansion beyond 16 channels. To run this test on a group of channels other than 0-17, load 20, 40, or 60 into location BASECH (1336) for channels 20-37, 40-57, 60-77.

If 'C' is typed, the program will run the calibration test and will loop on that test until the operator halts it. If a certain AD11K is to be tested, its status register address must be loaded into \$BASE (1250), and its vector address must be loaded into the low byte of \$VECT1 (1244) (the high byte containing the priority).

If 'N' is typed, the program will run the Noise test tagged 'BEGINN' and will loop on this test until the operator halts it. If a certain AD11K is to be tested its status register address must be loaded into \$BASE (1250), and its vector address must be loaded into the low byte of \$VECT1 (1244) (the high byte containing the priority).



If "S" is typed, the program will run the Interchannel Settling test tagged "BEGINS" and will loop on this test until the operator halts it. At the beginning of this test, the operator must respond to the statements asking for the "FROM" channel and the "TO" channel by typing in the channel value in octal and hitting carriage return. If a certain AD11K is to be tested its status register address must be loaded into \$BASE (1250), and its vector address must be loaded into \$VECT1 (1244) (the high byte containing the priority).

If "A" is typed, the program will execute the logic tests, analog tests, noise, settle and differential linearity. At the beginning of the test the program will type "XX AD11K's Found". Where XX IS THE NUMBER OF AD11K's in octal. If the number is greater than 1, the test will be run successively on each AD11K. The program supports AD11K expansion beyond 16 channels. To run this test on a group of channels other than 0-17, load 20,40, or 60 into location BASECH (1336) for channels 20-37, 40-57, 60-77.

If "L" is typed, the program will execute the logic tests, printing "END PASS" when it has completed an entire pass. At the beginning of the test the program will type "XX AD11K's Found". Where XX is the number of AD11K's in octal. If the number is greater than 1, the test will be run successively on each AD11K.

## 6.0 ERRORS

-----

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

ERRPC: Location at which an error was detected.  
STREG: Address of the status register.  
ADBUFF: Address of the buffer  
CHANL: Channel value  
NOMINAL: Expected correct data  
TOLERANCE: The acceptable deviation from the nominal  
ACTUAL: Actual data  
EXPECTED: Expected correct data

## 7.0 MISCELLANEOUS

-----

### 7.1 Execution Time

Execution time for each of the tests is:

Calibration: 8 conversions/5 seconds @ 110 baud  
Wraparound Test: 17 minutes first pass; 35 minutes  
for successive passes  
Settling Test: 1 minute  
Noise Test: 1 minute  
Logic Test: 1 minute  
Auto Test: 18 minutes first pass, 36 minutes  
for successive passes

### 7.2 Status Register and Vector Addresses and Priority

When testing more than one AD11K, the difference in addresses is presently 40 for bus address and vector address. These values are in VADR (bus address) (1326) and VVCT (vector address) (1330). The first AD11K's status register address must be in \$BASE (1250), its vector address must be in the low byte of \$VECT1 (1244), and the priority must be in the high byte of \$VECT1.

### 7.3 AD11K Priority

If AD11K is set for a priority other than 6, the high byte of \$VECT1 (1244) must be adjusted accordingly (the low byte containing the vector address). If more than one AD11K is being tested, all must be set at the same priority.



#### 7.4 Switch Register

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

#### 7.5 BIT-MAP Graphic Output

The screen display may be halted for examination by setting bit 12. And then just hit continue to complete the program's execution.

#### 7.6 USER LINK TO I/O DEVICE

A SPECIAL USER LINK HAS BEEN PROVIDED IN ORDER FOR THE OPERATOR TO EXAMINE OR MODIFY LOCATIONS ON THE LPA11-KX I/O BUS. (NOTE: THIS CANNOT BE DONE DIRECTLY.)

##### PROCEDURE:

- 1) START THE PROCESSOR AT LOCATION 214
- 2) THE DIALOG TO EXAMINE A LOCATION IS AS FOLLOWS:

```
E OR D      "E"  
DEVICE ADDRS= "OCTAL ADDRS"  
XXXXXX
```

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

- 3) THE DIALOG TO MODIFY A LOCATION IS AS FOLLOWS:

```
E OR D      "D"  
DATA=      "DATA TO BE DEPOSITED"
```

- 4) THE PROGRAM WILL STAY IN THIS LOOP UNTIL THE OPERATOR IS FINISHED. AT THIS TIME THE PROCESSOR SHOULD BE HALTED.

NOTE: THE OPERATORS RESPONSE IS ENCLOSED IN QUOTES.

## 8.0 RESTRICTIONS

---

- 8.1 A G5036 wraparound module must be present when running the auto test and the wraparound test.

Switch on G5036 must be in '0' position.

```
*****
The wraparound (G5036) module must be connected as follows:
  AD11K TO BC08R CONNECTION A-A, VV-VV
  BC08R TO G5036 CONNECTION 'UPSIDE-DOWN' A-VV, VV-A
*****
```

- 8.2 The program is chainable under XXDP. However, you may only execute it one pass ( R RLPKC0/1 ). The reason is because of the 'BUS RESET' in the end of pass handler.

## 9.0 PROGRAM DESCRIPTION

---

### 9.1 Logic Tests

These 8 logic subtests run sequentially without further operator intervention after he/she has typed in the number of AD11K's to be tested. Its purpose is to check that each of the mux bits can be loaded and properly read back; that initialize clears the external start enable bit, the done bit, the interrupt enable bit, the overflow bit, the error flag, and the A/D start bit. It also checks that the A/D done flag sets at end of conversion and clears when the converted value is read. It checks the interrupt logic and the correct setting of the error flag.

### 9.2 Calibration Test

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



### 9.3 Differential Linearity

This test is to determine if a change in the input voltage represents a similar change in the resulting converted binary value.

### 9.4 Settling Test

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

### 9.5 Noise Test

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

### 9.6 Analog Tests

These 11 subtests check the channels and their output.

## 10. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY

---

DIAGNOSTICS FOR THE LPA11 ARE WRITTEN AT THREE LEVELS: (1) TOTAL PDP-11 SYSTEM, (2) LPA11 SYSTEM; AND, (3) LPA11 OPTIONS.

LEVEL 1, IS DESIGNED TO ISOLATE A FAILURE TO THE LPA11 SYSTEM. ALL OPTIONS ON THE PDP-11 ARE EXERCISED.

LEVEL 2 DIAGNOSTICS ISOLATE A FAILURE TO THE INDIVIDUAL OPTION WITHIN THE LPA11. THE LEVEL 2 DIAGNOSTIC IS MD-11-CRLPA. WHEN THE USER RUNS "CRLPA" HE CAN GENERALLY TELL WHICH OPTION DIAGNOSTIC (LEVEL 3) TO RUN NEXT. M8254 AND M8200-YC ERRORS MAY "LOOK" ALIKE AND "CRLPA" MAY NOT BE ABLE TO DISTINGUISH BETWEEN THEM. ARBITRATION ERRORS WILL NOT BE DETECTED BY THIS DIAGNOSTIC.

LEVEL THREE DIAGNOSTICS AID IN DETERMINING IF THE ERROR WAS IN FACT ON THE OPTION THE "CRLPA" SPECIFIED. THE USER MAY "LOOP" ON THE ERROR. WITHIN LEVEL THREE, THERE ARE TWO GROUPS OF DIAGNOSTICS. THE FIRST GROUP REQUIRES NO "EXTRA" WORK BY THE USER IN ORDER TO RUN. GROUP "A" DIAGNOSTICS DO NOT CHECK ARBITRATION, AND REQUIRE EXTRA TIME FOR EXECUTION. THE SECOND GROUP (GROUP "B") REQUIRES THAT THE USER RECONFIGURE THE PDP-11 SYSTEM. THIS RECONFIGURATION INVOLVES CABLING THE UNIBUS TO THE LPA'S I/O BUS.

THE DIAGNOSTIC FOR THE M8254 FALLS INTO THE GROUP "B" CATEGORY.

THE LPA11-KX DIAGNOSTIC KIT WILL INCLUDE:

K 1

SEQ 0010

<u>OPTION</u>	<u>GROUP</u>	<u>DIAG. #</u>	<u>DIAG. TITLE</u>
LPA11-KX	LEVEL 2	MD-11-CRLPA	LPA11-K SYSTEM EXER.
M8254	'B'	MD-11-CRLPN	M8254 (IPBM) FIELD DIAG.
AA11-K	A	MD-11-CRLPB	LPA/AA11-K DIAG.
	B	MD-11-DZAAC	AA11-K DIAG.
AR11	A	MD-11-CRLPC	LPA/AR11 DIAG. #1
	A	MD-11-CRLPD	LPA/AR11 DIAG. #2
	A	MD-11-CRLPE	LPA/AR11 DIAG. #3
	B	MD-11-DZARA	AR11 DIAG. #1
	B	MD-11-DZARB	AR11 DIAG. #2
	B	MD-11-DZARC	AR11 DIAG. #3
	DR11-K	A	MD-11-CRLPF
B		MD-11-DZDRG	DR11-K DIAG.
KW11-K	A	MD-11-CRLPG	LPA/KW11-K DIAG.
	B	MD-11-DZKWK	KW11-K DIAG.
LPS11	A	MD-11-CRLPH	LPA/LPS11 DIAG. #1
	A	MD-11-CRLPI	LPA/LPS11 DIAG. #2
	A	MD-11-CRLPJ	LPA/LPS11 DIAG. #3
	B	MD-11-DZLPC	LPS11 DIAG. #1
	B	MD-11-DZLPD	LPS11 DIAG. #2
	B	MD-11-DZLPI	LPS11 DIAG. #3
AD11-K	A	MD-11-CRLPK	LPA/AD11-K DIAG.
	B	MD-11-DZADL	AD11-K DIAG.
M8200-YC	B	MD-11-CRLPL	LPA/DMC-11 DIAG. TST I
	B	MD-11-CRLPM	LPA/DMC-11 DIAG. TST II



PRODUCT CODE: MAINDEC-11-DZADL-B  
 PRODUCT NAME: AD11-K PERFORMANCE TEST  
 DATE: DECEMBER 1976  
 MAINTAINER: DIANOSTIC GROUP

\*\*\*\*\*

PRODUCT CODE: MAINDEC-11-DRLPK-A  
 PRODUCT NAME: LPA/AD11-K PERFORMANCE TEST  
 DATE: JANUARY 1978  
 MAINTAINER: DIAGNOSTIC GROUP

REASON FOR DEVELOPMENT:

- 1) TO ENABLE THE OPERATOR TO CHECK OUT THE AD11-K OPTION WHEN IT IS ON THE LPA11-KX I/O BUS.

CHANGES MADE:

- 1) TOOK OUT CERTAIN TESTS FROM ORIGINAL DIAGNOSTIC (I.E. INTERRUPTS, TIME DEPENDENT CODE).
- 2) REPLACED DIRECT LINKS TO DEVICE WITH MACRO CALLS TO THE KMC-11 MICRO CODE. KMC-11 MICRO CODE (FILE:DRLPX2) HANDLES DIRECT COMMUNICATIONS WITH THE DEVICE.

FILE: DRLPA.MAC  
 CONTAINS MACRO LINKS BETWEEN PDP-11 CODE AND KMC-11 MICRO CODE. FILE: DRLPX2 NEEDS TO BE ASSEMBLED WITH DRLPK (SEE .CTL FILE).

FILE: DRLPX2  
 MICRO CODE FILE THAT GETS LOADED INTO THE KMC-11 VIA ROUTINES IN DRLPA.MAC.

DRLPX2.P11 IS ASSEMBLED WITH MACY11 (ONLY) AS ANY OTHER .P11 FILE. THE RESULTS OF ITS ASSEMBLY IS A .OBJ MODULE AS WAS THE RESULT OF THE ASSEMBLY OF THE DIAGNOSTIC .P11 FILE. BOTH .OBJ FILES GET LINKED WITH LNKX11 (ONLY).

FILE: DRLPK.CTL  
 THIS FILE EXPLAINS SEQUENCE OF ASSEMBLES AND LINKS. IT IS IN TOPS-20 FORMAT.

\*\*\*\*\*

PRODUCT CODE: AC-B049B-MC

DIAGNOSTIC CODE: MD-11-CRLPK-B  
PRODUCE NAME: CRLPKB LPA/AD11-K TEST  
DATE REVISED: JULY 1979  
MAINTAINER: DIAGNOSTIC GROUP

M 1

SEQ 0012

\*\*\*\*\*

THE 'B' VERSION WAS GENERATED TO REPAIR THE FOLLOWING PROBLEMS:

1. PROGRAM LISTING DID NOT AGREE WITH THE BINARY FILE AFTER LOC. 12064. THIS WAS DUE TO THE RELEASE ENGINEERING GROUP REASSEMBLING TO GET THE LISTING AND USING THE BINARY FILE SUPPLIED BY AUTHOR. (DEVELOPED WITH C2 SYSMAC - RELEASED WITH C3 SYSMAC)
2. WHEN SUBTEST 'A' OR 'W' WAS SELECTED, A 'MICRO-CODE LOAD ERROR' OCCURRED AT LOCATION 17612 ON THE 'THIRD PASS'. (DUE TO THE AUTHOR FORGETTING ABOUT WHERE THE MICRO-CODE 'HIDES' AT.
3. 'TST11' COULD NOT BE LOOPED ON CORRECTLY. (ORIGINAL PROGRAM USED A ABSOLUTE TAG FOR AT THAT TEST <<TST17>>)
4. AFTER A POWER FAILURE, THE PROGRAM APPEARED TO RECOVERY PROPERLY. BUT AFTER THE OPERATOR ENTERED THE TEST NUMBER THE PROGRAM REPORTED 'LPA FAULT' AND THEN HALTS. (PROGRAM DID A RESTART - IT MUST BE STARTED)

- .....
1. REASSEMBLED THE FILE - <EASY AND FREE FIX WHEN WORKING ON PROBLEM 2-4
  2. PROTECT THE 'HIDDEN' SPACE THAT THE MICRO-CODE RESIDES AT.
  3. REMOVE INCORRECT TAG FROM 'TST11'
  4. BECAUSE THE KMC-11 IS A VOLIATLE DEVICE A COMPLETE PROGRAM START WAS NEEDED. JUST A ONE LOCATION PATCH IN THE POWER FAIL ROUTINE FIXES THE PROBLEM.

\*\*\*\*\*

VERSION 'C' WAS MADE BECAUSE OF A MICRO-CODE CHANGE IN THE LPA-11. EFFECTIVELY ONLY THE TITLE AND TITLE TYPEOUT WERE CHANGED IN THE SOURCE. THE MAJOR CHANGE OCCURRED IN THE MACRO FILE THAT THE PROGRAM IS ASSEMBLED WITH.



#CRLPKC.BIN/B:42000,CRLPKC.MAP=CRLPKC,CRLPX2/E

## LOAD MAP

IDENT: 4.01

TRANSFER ADDRESS: 000001

LOW LIMIT: 042000

HIGH LIMIT: 046000

\*\*\*\*\*

MODULE	LPA	SECTION ENTRY	ADDRESS	SIZE
<. ABS.>			000000	000000
<	DRLPX2	>	042000	
<		>	042000	000000

\*\*\*\*\*

MODULE	DRLPX2	SECTION ENTRY	ADDRESS	SIZE
<		>	042000	000000
<ABCODE>			042000	004000

RUN-TIME: 0 SECONDS

2K CORE USED

LPA-AD11K TEST MD-11-CRLPKC MACY11 30G(1063) 24-OCT-80 09:48

CRLPKC.P11 14-AUG-80 13:59

TABLE OF CONTENTS

SEQ 0014

1163	BASIC DEFINITIONS
1164	OPERATIONAL SWITCH SETTINGS
1215	TRAP CATCHER
(1)	STARTING ADDRESS(ES)
1220	ACT11 HOOKS
1222	APT PARAMETER BLOCK
1223	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
1263	MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
1313	CONTROL A AND C DECODERS
1343	INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE
1348	INITIALIZE THE COMMON TAGS
1354	DETERMINE IF VT55 TYPE TERMINAL IS PRESENT
1446	T1 FLOAT A ONE THRU MULTIPLEXER BITS
1455	T2 LOAD AND READ BACK INTERRUPT ENABLE BIT6
1461	T3 LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BITS
1466	T4 LOAD AND READ BACK EXTERNAL START ENABLE BIT4
1470	T5 LOAD AND READ BACK ERROR FLAG BIT15
1475	T6 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
1486	T7 TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
1496	T10 TEST ERROR FLAG SETS IF 2ND CONVERSION ENDS BEFORE READING BUFFER
1524	WRAPAROUND TEST SECTION
1526	T11 TEST CH14 GROUND
1537	T12 TEST CONVERSION FROM EXT. START
1553	T13 TEST CH0 GROUND
1561	T14 TEST CH1 GROUND
1569	T15 TEST CH2 +1 VOLT
1578	T16 TEST CH3 +2.5 VOLTS
1586	T17 TEST CH4 -2.5 VOLTS
1594	T20 TEST VERNIER OFFSET DAC ON CH12
1641	T21 TEST CH13 +2.5 VOLTS
1648	T22 TEST CH17 +4V

1655 T23 OFFSET ON CHO  
1682 T24 NOISE TEST ON 8 EDGES  
1691 T25 SETTLE TEST ON 8 EDGES  
1699 T26 DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST  
1710 CALIBRATION TEST  
1744 LOGIC TEST SECTION  
1753 AUTO TEST  
1770 WRAPAROUND TEST  
2508 END OF PASS ROUTINE  
2511 ASCII MESSAGES  
2600 TTY INPUT ROUTINE  
2602 READ AN OCTAL NUMBER FROM THE TTY  
2604 SCOPE HANDLER ROUTINE  
2605 ERROR HANDLER ROUTINE  
2606 ERROR MESSAGE TYPEOUT ROUTINE  
2608 TYPE ROUTINE  
2609 APT COMMUNICATIONS ROUTINE  
2611 BINARY TO OCTAL (ASCII) AND TYPE  
2612 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE  
2614 TRAP DECODER  
(3) TRAP TABLE  
2615 POWER DOWN AND UP ROUTINES



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
52  
53  
54  
140  
156  
169  
182  
183  
416  
417  
458  
510  
609  
651  
698  
747

.REM [

CRLPAB.MAC

WELCOME, THIS DIAGNOSTIC IS ONE IN A SERIES OF DIAGNOSTIC  
DESIGNED IN ORDER TO AID YOU IN TESTING THE LPA-11XX OPTION.  
I HOPE THAT YOU HAVE READ THE DOCUMENTATION SECTION OF THIS  
DIAGNOSTIC. IF YOU HAVE, YOU KNOW ABOUT ALL OF THE DIAGNOSTICS  
THAT ARE AVAILABLE FOR TESTING THE LPA SYSTEM.

GOOD LUCK !

[  
.GLOBL DRLPX2

MMAST.MAC  
CRLPAB.MAC

26-SEP-80 13:48

MACY11 30G(1063) 24-OCT-80 09:48 <sup>D 2</sup> PAGE 2

SEQ 0016

763  
764  
765  
766  
767  
768  
769  
770  
771  
905  
906  
907  
908  
909  
910  
911  
912  
913  
1047

.TITLE MMAST.MAC  
.IDENT /4.01/

:  
: LPA11-K MICRO CODE  
:  
: CHARLES A. SAMUELSON  
: NOVEMBER, 1977  
:

.TITLE DMAST.MAC  
.IDENT /4.01/

:  
: LPA11-K MICRO CODE  
:  
: CHARLES A. SAMUELSON  
: NOVEMBER, 1977  
:



```
1162 .TITLE LPA-AD11K TEST MD-11-CRLPKC
(1)  ;*COPYRIGHT (C) 1980
(1)  ;*DIGITAL EQUIPMENT CORP.
(1)  ;*MAYNARD, MASS. 01754
(1)  ;*
(1)  ;*PROGRAM BY MODIFIED BY R. SHOOP
(1)  ;*
(1)  ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)  ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)  ;*
1163 .SBTTL BASIC DEFINITIONS
(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)  ;*MISCELLANEOUS DEFINITIONS
(1)  000011 HT= 11      ;;CODE FOR HORIZONTAL TAB
(1)  000012 LF= 12      ;;CODE FOR LINE FEED
(1)  000015 CR= 15      ;;CODE FOR CARRIAGE RETURN
(1)  000200 CRLF= 200   ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)  177776 PS= 177776  ;;PROCESSOR STATUS WORD
(1)  .EQUIV PS,PSW
(1)  177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
(1)  177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)  177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1)  177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)  ;*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)  ;*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)  ;*'SWITCH REGISTER' SWITCH DEFINITIONS
(1)  100000 SW15= 100000
(1)  040000 SW14= 40000
(1)  020000 SW13= 20000
(1)  010000 SW12= 10000
```

BASIC DEFINITIONS

(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

;\*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

;\*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
1164 .SBTTL OPERATIONAL SWITCH SETTINGS
(1)
(1) * SWITCH USE
(1) * -----
(1) * 15 HALT ON ERROR
(1) * 14 LOOP ON TEST
(1) * 13 INHIBIT ERROR TYPEOUTS
(1) * 12 HALT FOR BIT-MAP DISPLAY
(1) * 11 INHIBIT ITERATIONS
(1) * 10 BELL ON ERROR
(1) * 9 LOOP ON ERROR
(1) * 8 LOOP ON TEST IN SWR<7:0>
1165 170400 ABASE= 170400
1166 140340 AVECT1= 140340
1167 000300 APRIOR= 300
1168
1173
1180
1185
1192
1197
1203
1209
1214
1215 .SBTTL TRAP CATCHER
(1)
(1) 000000 .=0
(1) ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1) ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1) ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1) .=174
(1) 000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
(1) .SBTTL STARTING ADDRESS(ES)
(1) 000200 000137 001712 JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
1216 000204 000137 002402 JMP @#BEG2 ;RESTART ADDRESS
1217 000210 000137 001720 JMP @#BEGIN2 ;START ADDRESS FOR OPTION TEST AREA
1218 000214 000137 020346 JMP @#SUTK ;STARTING ADDRESS FOR USER LINK
```

```
1220 .SBTTL ACT11 HOOKS
(1)
(2)
(1)
(1)
(1) 000220
(1) 000046
(1) 000046 012074
(1) 000052
(1) 000052 000000
(1) 000220
1221 001000
1222 .SBTTL APT PARAMETER BLOCK
(1)
(2)
(1)
(2)
(1) 001000
(1) 000024
(1) 000024 000200
(1) 000044
(1) 000044 001000
(1) 001000
(2)
(1)
(1)
(1)
(1) 001000
(1) 001000 000000
(1) 001002 001174
(1) 001004 002260
(1) 001006 000764
(1) 001010 003244
(1) 001012 000031

:*****
:HOOKS REQUIRED BY ACT11
$SVPC= . ;SAVE PC
.=46
$ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
.=52
.WORD 0 ;:2)SET LOC.52 TO ZERO
.=$SVPC ;: RESTORE PC
.=1000

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

$APTHD:
$HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
$STIM: .WORD 1200. ;:RUN TIM OF LONGEST TEST
$PASTM: .WORD 500. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 1700. ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
SETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
```



1223  
 (1)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1) 001100 001100  
 (1) 001100 000000  
 (1) 001102 000  
 (1) 001103 000  
 (1) 001104 000000  
 (1) 001106 000000  
 (1) 001110 000000  
 (1) 001112 000000  
 (1) 001114 000  
 (1) 001115 001  
 (1) 001116 000000  
 (1) 001120 000000  
 (1) 001122 000000  
 (1) 001124 000000  
 (1) 001126 000000  
 (1) 001130 000000  
 (1) 001132 000000  
 (1) 001134 000  
 (1) 001135 000  
 (1) 001136 000000  
 (1) 001140 177570  
 (1) 001142 177570  
 (1) 001144 177560  
 (1) 001146 177562  
 (1) 001150 177564  
 (1) 001152 177566  
 (1) 001154 000  
 (1) 001155 002  
 (1) 001156 012  
 (1) 001157 000  
 (1) 001160 000000  
 (1) 001162 000000  
 (1) 001164 177607 000377  
 (1) 001170 077  
 (1) 001171 015  
 (1) 001172 000012

.SBTTL COMMON TAGS

\*\*\*\*\*  
 \*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
 \*USED IN THE PROGRAM.

.=1100

SCMTAG: .WORD 0 ;:START OF COMMON TAGS  
 \$TSTNM: .BYTE 0 ;:CONTAINS THE TEST NUMBER  
 \$ERFLG: .BYTE 0 ;:CONTAINS ERROR FLAG  
 \$ICNT: .WORD 0 ;:CONTAINS SUBTEST ITERATION COUNT  
 \$LPADR: .WORD 0 ;:CONTAINS SCOPE LOOP ADDRESS  
 \$LPERR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS  
 \$ERTTL: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED  
 \$ITEMB: .BYTE 0 ;:CONTAINS ITEM CONTROL BYTE  
 \$ERMAX: .BYTE 1 ;:CONTAINS MAX. ERRORS PER TEST  
 \$ERRPC: .WORD 0 ;:CONTAINS PC OF LAST ERROR INSTRUCTION  
 \$GDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'GOOD' DATA  
 \$BDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'BAD' DATA  
 \$GDDAT: .WORD 0 ;:CONTAINS 'GOOD' DATA  
 \$BDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA  
 .WORD 0 ;:RESERVED--NOT TO BE USED  
 \$AUTOB: .BYTE 0 ;:AUTOMATIC MODE INDICATOR  
 \$INTAG: .BYTE 0 ;:INTERRUPT MODE INDICATOR  
 .WORD 0  
 \$SWR: .WORD DSWR ;:ADDRESS OF SWITCH REGISTER  
 \$DISPLAY: .WORD DDISP ;:ADDRESS OF DISPLAY REGISTER  
 \$TKS: 177560 ;:TTY KBD STATUS  
 \$TKB: 177562 ;:TTY KBD BUFFER  
 \$TPS: 177564 ;:TTY PRINTER STATUS REG. ADDRESS  
 \$TPB: 177566 ;:TTY PRINTER BUFFER REG. ADDRESS  
 \$NULL: .BYTE 0 ;:CONTAINS NULL CHARACTER FOR FILLS  
 \$FILLS: .BYTE 2 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED  
 \$FILLC: .BYTE 12 ;:INSERT FILL CHARS. AFTER A 'LINE FEED'  
 \$TPFLG: .BYTE 0 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)  
 \$TIMES: 0 ;:MAX. NUMBER OF ITERATIONS  
 \$ESCAPE: 0 ;:ESCAPE ON ERROR ADDRESS  
 \$BELL: .ASCIZ <207><377><377> ;:CODE FOR BELL  
 \$QUES: .ASCII /?/ ;:QUESTION MARK  
 \$CRLF: .ASCII <15> ;:CARRIAGE RETURN  
 \$LF: .ASCIZ <12> ;:LINE FEED

.SBTTL APT MAILBOX-ETABLE

\*\*\*\*\*  
 .EVEN  
 \$MAIL: .WORD ;:APT MAILBOX  
 \$MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE  
 \$FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER  
 \$TESTN: .WORD ATESTN ;:TEST NUMBER  
 \$PASS: .WORD APASS ;:PASS COUNT  
 \$DEVCT: .WORD ADEVCT ;:DEVICE COUNT  
 \$UNIT: .WORD AUNIT ;:I/O UNIT NUMBER  
 \$MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS

(2)	001212	000000	\$MSG LG: .WORD	AMSG LG	::MESSAGE LENGTH
(2)	001214		\$ETABLE:		::APT ENVIRONMENT TABLE
(2)	001214	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
(2)	001215	000	\$ENV M: .BYTE	AENV M	::ENVIRONMENT MODE BITS
(2)	001216	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
(2)	001220	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
(2)	001222	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
(2)			*		BITS 15-11=CPU TYPE
(2)			*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)			*		11/70=06,PDQ=07,Q=10
(2)			*		BIT 10=REAL TIME CLOCK
(2)			*		BIT 9=FLOATING POINT PROCESSOR
(2)			*		BIT 8=MEMORY MANAGEMENT
(2)	001224	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001225	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
(2)			*		MEM.TYPE BYTE -- (HIGH BYTE)
(2)			*		900 NSEC CORE=001
(2)			*		300 NSEC BIPOLAR=002
(2)			*		500 NSEC MOS=003
(2)	001226	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2)	001230	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001231	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
(2)	001232	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001234	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001235	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
(2)	001236	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001240	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001241	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
(2)	001242	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001244	140340	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001246	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001250	170400	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001252	000000	\$DEV M: .WORD	ADEV M	::DEVICE MAP
(2)	001254	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001256		\$ETEND:		
(2)			.MEXIT		



```
(1) .SBTTL ERROR POINTER TABLE
(1)
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;* EM ;:POINTS TO THE ERROR MESSAGE
(1) ;* DH ;:POINTS TO THE DATA HEADER
(1) ;* DT ;:POINTS TO THE DATA
(1) ;* DF ;:POINTS TO THE DATA FORMAT
(1)
(1) $ERRTB:
(1) 001256
1225
1226
1227
1236 ;ITEM 1
1237 001256 014245 EM1 ;STATUS REG. ERROR
1238 001260 014405 DH1 ;ERRPC STREG EXPECTED ACTUAL
1239 001262 014570 DT1 ;$ERRPC, STREG, $GDDAT, $BDDAT
1240 001264 014630 DF1
1241
1242
1243 ;ITEM 2
1244 001266 014273 EM2 ;FAILED TO INTERRUPT
1245 001270 014526 DH3 ;ERRPC STREG ACTUAL
1246 001272 014620 DT3 ;$ERRPC, STREG, $BDDAT
1247 001274 014630 DF1
1248
1249 ;ITEM 3
1250 001276 014323 EM3 ;UNEXPECTED INTERRUPT
1251 001300 014526 DH3 ;ERRPC STREG
1252 001302 014620 DT3 ;$ERRPC, STREG
1253 001304 014630 DF1
1254
1255 ;ITEM 4
1256 001306 014354 EM4 ;ERROR ON A/D CHANNEL
1257 001310 014443 DH2 ;ERRPC STREG CHAN NOMINAL TOL ACTUAL
1258 001312 014602 DT2 ;$ERRPC, STREG, CHANL, $GDDAT, SPREAD, $BDDAT
1259 001314 014630 DF1
1260
1261
```

Line	Code	Value	Parameter	Description
1263			.SBTTL	MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS
1264	001316	170400	STREG:	ABASE ;ADDRESS OF STATUS REGISTER
1265	001320	170402	ADBUFF:	ABASE+2 ;ADDRESS OF A/D BUFFER
1266	001322	000300	BASEBR:	APRIOR ;INTERRUPT PRIORITY LEVEL
1267	001324	140342	VECTR1:	AVECT1+2
1268	001326	000040	VADR:	40 ;INCREMENT FOR BUS ADDRESS
1269	001330	000040	VVCT:	40 ;INCREMENT FOR VECTOR ADDRESS
1270	001332	000000	BASECH:	0 ;BASE CHANNEL
1271	001334	000060	KBVECT:	60
1272	001336	000000	WIDE:	0 ;NO. OF WIDE STATES
1273	001340	000000	NARROW:	0 ;NO. OF NARROW STATES
1274	001342	000000	FIRST:	0
1275	001344	000000	SKIPST:	0 ;NO. OF SKIPPED STATES
1276	001346	000000	TEMP:	0 ;WORK AREA
1277	001350	000000	CH1:	0 ;FIRST CHANNEL
1278	001352	000000	CH2:	0 ;SECOND CHANNEL
1279	001354	000000	NBEXT:	0 ;NO. OF AD11K'S TO BE TESTED
1280	001356	000000	NMBEXT:	0 ;NO. OF AD11K'S TO BE TESTED
1281	001360	000000	DUMMY:	0 ;DUMMY CHANNEL
1282	001362	000000	CHANL:	0 ;CHANNEL VALUE
1283	001364	000000	TADDR:	0 ;TEST ADDRESS
1284	001366	000000	RNA:	0 ;RANDOM
1285	001370	000000	RNB:	0 ;NUMBER
1286	001372	000000	RNC:	0 ;VALUES
1287	001374	000000	RMS:	0 ;RMS NOISE VALUE
1288	001376	000000	PEAK:	0 ;PEAK NOISE VALUE
1289	001400	000000	FLAG:	0 ;VT55 FLAG
1290	001402	000000	SPREAD:	0 ;DEVIATION FROM THE NOMINAL
1291	001404	000000	DAC:	0 ;SAR VALUE
1292	001406	000000	DELAY:	0 ;TIME DELAY COUNTER
1293	001410	000000	EDGE:	0 ;EDGE VALUE
1294	001412	000000	BITPNT:	0
1295	001414	000000	MIN:	0 ;MIN VALUE
1296	001416	000000	WFTST:	0 ;OPTION TEST AREA FLAG
1297	001420	000000	MAX:	0 ;MAX VALUE
1298	001422	000000	PERCNT:	0 ;PERCENT FOR SAR ROUTINE
1299	001424	000000	OUT:	0
1300	001426	000000	MYTEMP:	0
1301	001430	000000	\$TEMP1:	0
1302	001432	000000	\$TEMP2:	0



1304

1305

(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)

: ADDRESS OF KMC-11 OF LPA-11

THE ADDR FOR KMADO MAY BE  
CHANGED BY THE USER TO REFLECT  
A DIFFERENT KMC-11 ADDR. THE  
REST OF THE ADDRESSES WILL  
BE CHANGED BY THE PROGRAM.

(1)

001434

LPCI:

(1)

001434

170460

KMADO:

.WORD 170460

;BASE KMC ADDR. MAY BE PATCHED BY USER.

(1)

001436

LPMR:

(1)

001436

170461

KMAD1:

.WORD 170460+1

;>DO NOT <;KMC-CSR ADDR

(1)

001440

LPCO:

(1)

001440

170462

KMAD2:

.WORD 170460+2

;>PATCH <;

(1)

001442

LPSO:

(1)

001442

170463

KMAD3:

.WORD 170460+3

;>THIS AREA <

(1)

001444

LPADL:

(1)

001444

170464

KMAD4:

.WORD 170460+4

;

(1)

001446

LPADH:

(1)

001446

170465

KMAD5:

.WORD 170460+5

;>DO NOT <

(1)

001450

LPMS1:

(1)

001450

170466

KMAD6:

.WORD 170460+6

;>PATCH <

(1)

001452

LPMS2:

(1)

001452

170467

KMAD7:

.WORD 170460+7

;>THIS AREA <

(1)

001454

000340

VECTOR:

.WORD AVECT18777

;BASE VECTOR OF KMC

(1)

001456

000344

VECTPS:

.WORD 4+AVECT18777

;VECTR ADDR.+2

(1)

001460

000005

VERSN:

.WORD 5

;CURRENT VERSION NUMBER OF MICROCODE.

(1)

001462

000000

.DVLS:

.WORD 0

;/DEVICE LIST OF I/O ADDR. DEFINED

(1)

001464

000020

.BLKW

16.

;/BY INIT.

(1)

1306

1307

001524

UNEXP:

(1)

001524

012737

001540

001162

MOV #1\$, \$ESCAPE

;;ESCAPE TO 1\$ ON ERROR

1308

001532

005237

001103

INC \$ERFLG

1309

001536

104003

ERROR 3

1310

001540

005037

001162

1\$:

CLR \$ESCAPE

;RETURN ESCAPE TO NORMAL

1311

001544

000002

RTI

;UNEXPECTED INTERRUPT

				SBTTL	CONTROL A AND C DECODERS		
1313				ISERV:	MOV	RO,-(SP)	;SAVE RO
1314	001546	010046			MOV	@\$TKB,RO	;GET CHARACTER
1315	001550	017700	177372		BIC	#177600,RO	
1316	001554	042700	177600		CMPB	RO,#3	;IS IT ^C?
1317	001560	120027	000003		BNE	1\$	
1318	001564	001010			TYPE	,CMG	;ECHO CHARACTER
1319	001566	104401	012246		MOV	#STACK,SP	
1320	001572	012706	001100		JSR	PC,RST	;RESET & SET INTRPT. EN.
1321	001576	004737	011350		JMP	BEG2	
1322	001602	000137	002402		1\$:	CMPB	RO,#1
1323	001606	120027	000001		BNE	2\$	;IS IT ^A?
1324	001612	001010			TYPE	,AMG	;ECHO CHARACTER
1325	001614	104401	012241		MOV	#STACK,SP	
1326	001620	012706	001100		JSR	PC,RST	;RESET & SET INTRPT. EN.
1327	001624	004737	011350		JMP	@ADDR	;RETURN TO TEST
1328	001630	000177	177530		2\$:	CMPB	RO,#7
1329	001634	120027	000007		BNE	NONE	;IS IT ^G?
1330	001640	001021			CMP	SWR,#177570	;HARDWARE SWREG?
1331	001642	023727	001140 177570		BEQ	NONE	
1332	001650	001415			TYPE	,GMG	;ECHO CHARACTER
1333	001652	104401	012253		MOV	@SWR,-(SP)	::SAVE @SWR FOR TYPEOUT
1334	001656	017746	177256				::TYPE SWREG
(1)					TYPOS		::GO TYPE--OCTAL ASCII
(1)	001662	104403			.BYTE	6	::TYPE 6 DIGITS
(1)	001664	006			.BYTE	1	::TYPE LEADING ZEROS
(1)	001665	001			TYPE	,SLASH	
1335	001666	104401	012433		RDOCT		;READ NEW VALUE
1336	001672	104410			MOV	(SP)+,@SWR	;LOAD NEW SWREG VALUE
1337	001674	012677	177240		POPPO:	MOV	(SP)+,RO
1338	001700	012600			RETURN:	RTI	
1339	001702	000002			NONE:	TYPE	,QUEST
1340	001704	104401	012237		BR	POPPO	;TYPE "'?'"
1341	001710	000773					



```
1343 .SBTTL INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
1344 BEGIN: CLR WFTST
1345 001712 005037 001416 BR RBEG
1346 001716 000403 001416 BEGIN2: MOV #1,WFTST
1347 001720 012737 000001 RBEG: ;RESET
1348 001726 .SBTTL INITIALIZE THE COMMON TAGS
(1) ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001726 012706 001100 MOV #CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
(1) 001732 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
(1) 001734 022706 001140 CMP #SWR,R6 ;:DONE?
(1) 001740 001374 BNE -6 ;:LOOP BACK IF NO
(1) 001742 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
(1) ;:INITIALIZE A FEW VECTORS
(1) 001746 012737 015226 000020 MOV #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
(1) 001754 012737 000340 000022 MOV #340,@IOTVEC+2 ;:LEVEL 7
(1) 001762 012737 015504 000030 MOV #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
(1) 001770 012737 000340 000032 MOV #340,@EMTVEC+2 ;:LEVEL 7
(1) 001776 012737 021324 000034 MOV #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
(1) 002004 012737 000340 000036 MOV #340,@TRAPVEC+2 ;:LEVEL 7
(1) 002012 012737 021402 000024 MOV #PWRDN,@PWRVEC ;:POWER FAILURE VECTOR
(1) 002020 012737 000340 000026 MOV #340,@PWRVEC+2 ;:LEVEL 7
(1) 002026 013737 012042 012034 MOV SENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
(1) 002034 005037 001160 CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
(1) 002040 005037 001162 CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002044 112737 000001 001115 MOV #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
(1) 002052 012737 002052 001106 MOV #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002060 012737 002060 001110 MOV #,$SLPERR ;:SETUP THE ERROR LOOP ADDRESS
(2) ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002066 013746 000004 MOV @ERRVEC,-(SP) ;:SAVE ERROR VECTOR
(2) 002072 012737 002126 000004 MOV #64$,@ERRVEC ;:SET UP ERROR VECTOR
(2) 002100 012737 177570 001140 MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
(2) 002106 012737 177570 001142 MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
(2) 002114 022777 177777 177016 CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
(2) 002122 001012 BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;:AND THE HARDWARE SWR IS NOT = -1
(2) 002124 000403 BR 65$ ;:BRANCH IF NO TIMEOUT
(2) 002126 012716 002134 64$: MOV #65$,(SP) ;:SET UP FOR TRAP RETURN
(2) 002132 000002 RTI
(2) 002134 012737 000176 001140 65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
(2) 002142 012737 000174 001142 MOV #DISPREG,DISPLAY
(2) 002150 012637 000004 66$: MOV (SP)+,@ERRVEC ;:RESTORE ERROR VECTOR
(1)
(2) 002154 005037 001202 CLR $PASS ;:CLEAR PASS COUNT
(2) 002160 132737 000200 001215 BITB #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
(2) 002166 001403 BEQ 67$ ;:YES,USE NON-APT SWITCH
(2) 002170 012737 001216 001140 MOV #SSWREG,SWR ;:NO,USE APT SWITCH REGISTER
(2) 002176 67$:
```

```

1350      ; THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
(1)      ;
(1)      ;
(1)      ;
(1) 002176 010046      MOV      R0,-(SP)
(1) 002200 010146      MOV      R1,-(SP)
(1) 002202 013700 001434  MOV      KMADO,R0      ;GET KMC-11 ADDRESS.
(1) 002206 012701 001436  MOV      #KMAD1,R1     ;GET ADDR. OF ADDR. LIST.
(1)      ;
(1) 002212 005200      68$: INC      R0      ;UPDATE ADDR.
(1) 002214 010021      MOV      R0,(1)+      ;WRITE ADDR.
(1) 002216 020127 001454  CMP      R1,#KMAD7+2  ;DONE ALL ADDRESSES?
(1) 002222 001373      BNE      68$          ;NO - DO NEXT ADDR.
(1) 002224 005037 001462  CLR      .DVLS        ;CLR ADDR. LIST.
(1) 002230 012601      MOV      (SP)+,R1
(1) 002232 012600      MOV      (SP)+,R0
1351 002234 005037 001400  CLR      FLAG         ;CLEAR VT55 FLAG
1352 002240 005737 000042  TST      @#42         ;IS IT CHAINED?
1353 002244 001033      BNE      REST1
1354      ;
1355 002246 042777 000100 176670 .SBTTL  DETERMINE IF VT55 TYPE TERMINAL IS PRESENT
1356 002254 104401 013673      BIC      #100,@$TKS
1357 002260 004737 002664      TYPE    ,C0          ;TYPE ASCIZ STRING
1358 002264 020027 000033      JSR     PC,VTFLG     ;GET A CHARACTER
1359 002270 001017      CMP     R0,#33
1360 002272 004737 002664      BNE    NOVT55        ;NO VT55 PRESENT
1361 002276 020027 000057      JSR    PC,VTFLG     ;GET A CHARACTER
1362 002302 001012      CMP    R0,#57
1363 002304 004737 002664      BNE    NOVT55        ;NO VT55 PRESENT
1364 002310 020027 000103      JSR    PC,VTFLG     ;GET A CHARACTER
1365 002314 001403      CMP    R0,#103
1366 002316 020027 000105      BEQ    VT55          ;VT55 IS PRESENT
1367 002322 001002      CMP    R0,#105
1368 002324 005237 001400      BNE    NOVT55
VT55:   INC    FLAG
  
```



```
1370 ; DIALOGUE TO DETERMINE WHICH TEST TO RUN
1371 002330 104401 014036 NOVT55: TYPE ,HEAD1
1372 002334 004737 005364 REST1: JSR PC, FIXONE ; INITIALIZE ADDRESSES
1373 002340 013700 001334 MOV KBVECT, RO
1374 002344 012720 001546 MOV #ISERV, (RO)+
1375 002350 012710 000340 MOV #340, (RO)
1376 002354 012737 062341 001366 MOV #62341, RNA ; RANDOM NO, VARIABLES
1377 002362 012737 142315 001370 MOV #142315, RNB
1378 002370 012737 127623 001372 MOV #127623, RNC
1379 002376 004737 011636 JSR PC, WFADJ ; STANDARD OR OPTION TEST TOLERANCES?
1380 002402 012706 001100 BEG2: MOV #STACK, SP ; RESET STACK IN CASE RESTARTED
1381 002406 005737 000042 TST @#42 ; IS IT CHAINED?
1382 002412 001402 BEQ 1$
1383 002414 000137 005122 JMP BEGL ; GO TO LOGIC TESTS
1384 002420 104401 013501 1$: TYPE ,MSG71
1385 002424 104407 TRYAG: RDLIN
1386 002426 052777 000100 176510 BIS #100, @STKS
1387 002434 005037 177776 CLR PSW
1388 002440 012600 MOV (SP)+, RO ; READ ANSWER
1389 002442 142710 000040 BICB #40, (RO)
1390 002446 121027 000101 CMPB (RO), #'A
1391 002452 001002 BNE 1$ ; IS IT A?
1392 002454 000137 005160 JMP BEGINA ; NO, TRY C
1393 002460 121027 000103 1$: CMPB (RO), #'C ; GO TO AUTO TEST
1394 002464 001002 BNE 2$ ; IS IT C?
1395 002466 000137 004664 JMP BEGINC ; NO, TRY L
1396 002472 121027 000114 2$: CMPB (RO), #'L ; GO TO CALIBRATION TEST
1397 002476 001002 BNE 3$ ; IS IT L?
1398 002500 000137 005122 JMP BEGL ; NO, TRY N
1399 002504 121027 000116 3$: CMPB (RO), #'N ; GO TO LOGIC TESTS
1400 002510 001002 BNE 4$ ; IS IT N?
1401 002512 000137 005526 JMP BEGINN ; NO, TRY S
1402 002516 121027 000123 4$: CMPB (RO), #'S ; GO TO NOISE TEST
1403 002522 001002 BNE 5$ ; IS IT S?
1404 002524 000137 005576 JMP BEGINS ; NO, TRY W
1405 002530 121027 000127 5$: CMPB (RO), #'W ; GO TO SETTLE TEST
1406 002534 001002 BNE 6$ ; IS IT W?
1407 002536 000137 005246 JMP BEGINW ; NO, TRY AGAIN
1408 002542 104401 012237 6$: TYPE ,QUEST ; GO TO WRAPAROUND TEST
1409 002546 000726 BR TRYAG ; WAIT FOR CHARACTER
```

```
1411
1412 ;SIZE AND REPORT THE NUMBER OF AD11K DETECTED
1413
1414 002550 013737 001250 001126 TESTAD: MOV $BASE,$BDDAT ;SETUP TO TEST FOR AD11K'S
1415 002556 005037 001462 CLR .DVLS
1416 002562 005037 001464 CLR .DVLS+2
1417 002566 005037 001354 CLR NBEXT ;CLEAR AD11K COUNTER
1418 002572 1$: ;ADDRESS AD11K
1419
(1) ;
1420 002602 005737 017252 ; MOV $GDDAT,@$BDDAT ;/ PUT DATA FROM $GDDAT TO DEVICE REG $BDDAT
1421 002606 001006 ; TST $AERR ;DEVICE EXSIST? =0,YES
1422 ; BNE 2$ ;=1,NO.
1423 002610 005237 001354 INC NBEXT ;INCREMENT AD11K COUNTER
1424 002614 063737 001326 001126 ADD VADR,$BDDAT ;GET NEXT AD11K
1425 002622 000763 BR 1$ ;TRY NEXT AD11K
1426 002624 122737 000001 001134 2$: CMPB #1,$AUTOB ;TEST IF AUTO MODE
1427 002632 001406 BEQ 3$ ;BR IF YES
1428 002634 013746 001354 MOV NBEXT,-(SP) ;SAVE NBEXT FOR TYPEOUT
(1) ;TYPE NUMBER OF AD11K'S
(1) 002640 104403 TYPOS ;GO TYPE--OCTAL ASCII
(1) 002642 002 .BYTE 2 ;TYPE 2 DIGIT(S)
(1) 002643 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
1429 002644 104401 013041 TYPE ,MSG50
1430 002650 005337 001354 3$: DEC NBEXT ;ADJUST AD11K COUNT
1431 002654 013737 001354 001356 MOV NBEXT,NMBEXT ;KEEP COUNT OF NUMBER
1432 002662 000207 RTS PC
1433
1434 002664 005000 VTFLG: CLR R0 ;TEST FOR PRESENCE
1435 002666 105777 176252 1$: TSTB @$TKS ;OF VT55
1436 002672 100404 BMI 2$ ;VT55 RESPONDS WITH <33><57>[<103> OR <105>]
1437 002674 005300 DEC R0
1438 002676 001373 BNE 1$
1439 002700 005726 TST (SP)+ ;POP A WORD OFF STACK
1440 002702 000612 BR NOVT55 ;NO VT55 PRESENT
1441 002704 017700 176236 2$: MOV @$TKB,R0
1442 002710 042700 177600 BIC #177600,R0 ;TEST VT55 CODE
1443 002714 000207 RTS PC
```



```
1445 002716 BEGINL:
1446 (3) *****
(3) *TEST 1 FLOAT A ONE THRU MULTIPLEXER BITS
(2) 002716 012737 002716 001106 TST1: MOV #TST1,$LPADR
1447 002724 012737 002716 001110 MOV #TST1,$LPERR
1448 002732 012737 000400 001124 MOV #BIT8,$GDDAT ;LOAD FIRST BIT
1449 002740 004737 003406 2$: JSR PC,TESTIT
1450 002744 104001 ERROR 1 ;FAILED TO LOAD + READ BIT
1451 002746 006137 001124 1$: ROL $GDDAT ;GET NEXT BIT
1452 002752 023727 001124 040000 CMP $GDDAT,#BIT14 ;FINISHED?
1453 002760 001367 BNE 2$ ;;NO,GO TO NEXT TEST
1454
1455 *****
(3) *TEST 2 LOAD AND READ BACK INTERRUPT ENABLE BIT6
(3) *****
(2) 002762 000004 TST2: SCOPE
1456 002764 012777 001524 176462 MOV #UNEXP,@VECTOR ;SETUP FOR UNEXPECTED INTERUPT
1457 002772 012737 000100 001124 MOV #BIT6,$GDDAT ;LOAD EXPECTED DATA
1458 003000 004737 003406 JSR PC,TESTIT
1459 003004 104001 ERROR 1 ;FAILED TO LOAD + READ INTERRUPT ENABLE
1460
1461 *****
(3) *TEST 3 LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BITS
(3) *****
(2) 003006 000004 TST3: SCOPE
1462 003010 012737 000040 001124 MOV #BIT5,$GDDAT ;LOAD EXPECTED DATA
1463 003016 004737 003406 JSR PC,TESTIT
1464 003022 104001 ERROR 1 ;FAILED TO LOAD + READ CLOCK OVERFLOW START ENAB
1465
1466 *****
(3) *TEST 4 LOAD AND READ BACK EXTERNAL START ENABLE BIT4
(3) *****
(2) 003024 000004 TST4: SCOPE
1467 003026 012737 000020 001124 MOV #BIT4,$GDDAT ;LOAD EXPECTED DATA
1468 003034 004737 003406 JSR PC,TESTIT
1469 003040 104001 ERROR 1 ;FAILED TO LOAD + READ EXT. START ENABLE
1470
1471 *****
(3) *TEST 5 LOAD AND READ BACK ERROR FLAG BIT15
(3) *****
(2) 003042 000004 TST5: SCOPE
1471 003044 012737 100000 001124 MOV #BIT15,$GDDAT ;LOAD EXPECTED DATA
1472 003052 004737 003406 JSR PC,TESTIT
1473 003056 104001 ERROR 1 ;FAILED TO LOAD + READ ERROR FLAG
```

```
1475          ;:*****
(3)          ;*TEST 6      TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
(3)          ;:*****
(2) 003060 000004          TST6:  SCOPE
1476 003062 012700 001000      MOV      #BIT9,R0          ;STALL TIME COUNTER
1477
(2)
(2)          ;*      MOV      @STREG,MYTEMP      ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(1) 003076 005237 001426      INC      MYTEMP
(2)
(2)          ;*      MOV      MYTEMP,@STREG      ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1478 003112 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
1479 003120 005300          1$:      DEC      R0          ;STALL
1480 003122 001376          BNE      1$          ;TIME
1481
(2)
(2)          ;*      MOV      @STREG,MYTEMP      ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(1) 003134 042737 100000 001426      BIC      #BIT15,MYTEMP
(2)
(2)          ;*      MOV      MYTEMP,@STREG      ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1482 003152 004737 003416      JSR      PC,TEST
1483 003156 104001          ERROR    1          ;A/D DONE FLAG FAILED TO SET;BIT0 FAILED TO CLEAR
1484
(2)          ;*      MOV      @ADBUFF,MYTEMP      ;/READ DEVICE REG ADDBUFF,PUT DATA IN MYTEMP.
(1) 003170 013700 001426      MOV      MYTEMP,R0        ;/PUT CONVERTED VALUE IN R0.
1485
1486          ;:*****
(3)          ;*TEST 7      TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
(3)          ;:*****
(2) 003174 000004          TST7:  SCOPE
1487 003176 012737 000001 001426      MOV      #BIT0,MYTEMP
1488
(1)          ;*      MOV      MYTEMP,@STREG      ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1489 003214 005037 001124      CLR      $GDDAT
1490 003220          1$:
(2)
(2)          ;*      MOV      @STREG,MYTEMP      ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(1) 003230 105737 001426      TSTB    MYTEMP
1491 003234 100371          BPL      1$
1492
(2)          ;*      MOV      @ADBUFF,MYTEMP      ;/READ DEVICE REG ADDBUFF,PUT DATA IN MYTEMP.
(1) 003246 013700 001426      MOV      MYTEMP,R0        ;/PUT CONVERTED VALUE IN R0.
1493 003252 004737 003416      JSR      PC,TEST
1494 003256 104001          ERROR    1          ;DONE FLAG FAILED TO CLEAR
```



```
1496 (3) *****  
1497 (3) *TEST 10 TEST ERROR FLAG SETS IF 2ND CONVERSION ENDS BEFORE READING BUFFER  
1498 (2) 003260 000004 TST10: SCOPE  
1499 (1) 003262 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
1500 003270 012737 000001 001426 MOV #BIT0,MYTEMP  
1501 (1) * MOV MYTEMP,@STREG ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG  
1502 (2) 1$:  
1503 (2) * MOV @STREG,MYTEMP ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.  
1504 (1) 003316 105737 001426 TSTB MYTEMP  
1505 003322 100371 BPL 1$  
1506 003324 012737 100200 001124 2$: MOV #BIT15:BIT7,$GDDAT ;LOAD EXPECTED VALUE  
1507 003332 012737 000001 001426 MOV #BIT0,MYTEMP  
1508 (1) * MOV MYTEMP,@STREG ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG  
1509 003350 012700 001000 MOV #BIT9,R0 ;WAIT FOR 2ND  
1510 (2) 3$: DEC R0 ;CONVERSION TO END  
1511 (1) 003354 005300 BNE 3$  
1512 003356 001376 4$: JSR PC,TEST  
1513 003360 004737 003416 ERROR 1 ;ERROR FLAG NOT SET WHEN 2ND  
1514 003364 104001 ; CONVERT ENDS BEFORE READ BUFFER FROM FIRST  
1515 (2) * MOV @ADBUFF,MYTEMP ;/READ DEVICE REG ADBUFF,PUT DATA IN MYTEMP.  
1516 (1) 003376 013700 001426 MOV MYTEMP,R0 ;/PUT CONVERTED VALUE IN R0.  
1517 003402 000004 SCOPE  
1518 003404 000207 RTS PC ;RETURN TO TEST SECTION  
1519 (1) ;;SUBROUTINE FOR LOGIC TESIS;;  
1520 (1) TESTIT:  
1521 (1) * MOV $GDDAT,@STREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG  
1522 (1) TEST: MOV @STREG,$BDDAT ;/READ DEVICE REG STREG,PUT DATA IN $BDDAT.  
1523 003426 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULTS  
1524 003434 001002 BNE RETERR ;:ERROR RETURN  
1525 003436 062716 000002 ADD #2,(SP) ;BUMP RETURN ADDRESS TO GET AROUND ERROR  
1526 003442 000207 RETERR: RTS PC
```

```
1524 .SBTTL WRAPAROUND TEST SECTION
1525 003444 WRAP:
1526 :*****
(3) :*TEST 11 TEST CH14 GROUND
(3) :*****
(2) 003444 000240 TST11: NOP
(1) 003446 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
1527 003454 012737 000011 001102 MOV #$STN-1,$STNM
1528 003462 012737 003476 001110 MOV #1$,$LPERR
1529 003470 012737 003476 001106 MOV #1$,$LPADR
1530 003476 004537 011060 1$: JSR R5,CONVRT ;DO 8 CONVERSIONS
1531 003502 000014 14 ;
1532 003504 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS
1533 003510 004000 4000 ;NOMINAL
1534 003512 011714 V50 ;TOLERANCE
1535 003514 104004 ERROR 4 ;ERROR-CH14 NOT GROUND-AD11K MUST BE IN SINGLE-ENDED
1536 :CONFIGURATION,G5036 WRAPAROUND MODULE MUST BE PRESENT,CHECK CONNECTION A-VV,VV-A
1537 :*****
(3) :*TEST 12 TEST CONVERSION FROM EXT. START
(3) :*****
(2) 003516 000004 TST12: SCOPE
(1) 003520 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
1538 003526 005737 001332 TST BASECH ;TESTING AN AM?
1539 003532 001044 BNE TST13 ;;YES, GOTO NEXT TEST
1540 003534 012737 000020 001426 MOV #BIT4,MYTEMP
1541 (1) ;* MOV MYTEMP,@STREG ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1542 003552 012700 001000 MOV #BIT9,R0 ;TIME DELAY COUNTER
1543 003556 012737 000220 001124 MOV #BIT7!BIT4,$GDDAT ;LOAD EXPECTED
1544 003564 012737 000200 001426 MOV #200,MYTEMP
1545 (1) ;* MOV MYTEMP,@ADBUFF ;/ PUT DATA FROM MYTEMP TO DEVICE REG ADBUFF
1546 ; WRAPAROUND MODULE PRESENT
1547 003602 005300 1$: DEC R0
1548 003604 001376 BNE 1$
1549 003606 004737 003416 JSR PC,TEST
1550 003612 104001 ERROR 1 ;FAILED TO DO CONVERSION FROM EXT. START
1551 (2) ;* MOV @ADBUFF,MYTEMP ;/READ DEVICE REG ADBUFF,PUT DATA IN MYTEMP.
(1) 003624 013700 001426 MOV MYTEMP,R0 ;/PUT CONVERTED VALUE IN R0.
1552 003630 005037 001426 CLR MYTEMP
(2) ;* MOV MYTEMP,@STREG ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1553 :*****
(3) :*TEST 13 TEST CH0 GROUND
(3) :*****
(2) 003644 000004 TST13: SCOPE
(1) 003646 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
1554 003654 004537 011060 JSR R5,CONVRT ;CONVERT 8 TIMES
1555 003660 000000 0 ;
1556 003662 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS
1557 003666 004000 4000 ;NOMINAL
1558 003670 011706 V1 ;TOLERANCE
1559 003672 104004 ERROR 4 ;ERROR ON A/D CHANNEL
```



```
1561 (3) *****  
1562 (3) *TEST 14 TEST CH1 GROUND *****  
1563 (2) 003674 000004 TST14: SCOPE  
1564 (1) 003676 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
1565 003704 004537 011060 JSR R5,CONVRT ;CONVERT 8 TIMES  
1566 003710 000001 1 JSR R5,CONVRT ;CHANNEL 1  
1567 003712 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS  
1568 003716 004000 4000 ;NOMINAL  
1569 003720 011712 V10 ;TOLERANCE  
1570 003722 104004 ERROR 4 ;ERROR ON A/D CHANNEL  
1571 *****  
1572 (3) *TEST 15 TEST CH2 +1 VOLT *****  
1573 (3) *****  
1574 (2) 003724 000004 TST15: SCOPE  
1575 (1) 003726 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
1576 003734 004537 011060 JSR R5,CONVRT ;CONVERT 8 TIMES  
1577 003740 000002 2 JSR R5,CONVRT ;CHANNEL 2  
1578 003742 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS  
1579 003746 004632 4632 ;NOMINAL  
1580 003750 011714 V50 ;TOLERANCE  
1581 003752 104004 ERROR 4 ;ERROR ON A/D CHANNEL  
1582 ;AD11K MUST BE SET UP FOR +OR- 5V OR +OR- 5.12V  
1583 *****  
1584 (3) *TEST 16 TEST CH3 +2.5 VOLTS *****  
1585 (3) *****  
1586 (2) 003754 000004 TST16: SCOPE  
1587 (1) 003756 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
1588 003764 004537 011060 JSR R5,CONVRT ;CONVERT 8 TIMES  
1589 003770 000003 3 JSR R5,CONVRT ;CHANNEL 3  
1590 003772 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS  
1591 003776 006000 6000 ;NOMINAL  
1592 004000 011722 V240 ;TOLERANCE  
1593 004002 104004 ERROR 4 ;ERROR ON A/D CHANNEL  
1594 *****  
1595 (3) *TEST 17 TEST CH4 -2.5 VOLTS *****  
1596 (3) *****  
1597 (2) 004004 000004 TST17: SCOPE  
1598 (1) 004006 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
1599 004014 004537 011060 JSR R5,CONVRT ;CONVERT 8 TIMES  
1600 004020 000004 4 JSR R5,CONVRT ;CHANNEL 4  
1601 004022 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS  
1602 004026 002000 2000 ;NOMINAL  
1603 004030 011722 V240 ;TOLERANCE  
1604 004032 104004 ERROR 4
```

```
1594 (3) *****  
(3) *TEST 20 TEST VERNIER OFFSET DAC ON CH12  
(2) 004034 000004 TST20: SCOPE  
(1) 004036 012737 000001 001160 MOV #1,$TIMES ;;DO 1 ITERATION  
1595 004044 005037 001426 CLR MYTEMP  
1596 (1) ;* MOV MYTEMP,@ADBUFF ;/ PUT DATA FROM MYTEMP TO DEVICE REG ADBUFF  
1597 004060 004737 004654 JSR PC,DAWAIT ;DELAY FOR DAC SETTling  
1598 004064 004537 011060 JSR R5,CONVRT ;CONV. CH12, DIRECT VERNIER DAC  
1599 004070 000012 12  
1600 004072 013704 001346 MOV TEMP,R4 ;SAVE VALUE IN R4  
1601 004076 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS  
1602 004102 002376 2376 ;WITH -1.875 VOLTS  
1603 004104 011720 V115 ;TOLERANCE OF 10%  
1604 004106 104004 ERROR 4  
1605 004110 005037 001420 CLR MAX  
1606 004114 012702 000001 MOV #1,R2  
1607 004120 010237 001426 1$: MOV R2,MYTEMP ;SET UP NEXT VERNIER DAC VALUE  
1608 (1) ;* MOV MYTEMP,@ADBUFF ;/ PUT DATA FROM MYTEMP TO DEVICE REG ADBUFF  
1609 004134 004737 004654 JSR PC,DAWAIT ;DELAY FOR DAC SETTling  
1610 004140 004537 011060 JSR R5,CONVRT ;CONVERT IT  
1611 004144 000012 12  
1612 004146 005737 001420 TST MAX  
1613 004152 001010 BNE 2$  
1614 004154 023727 001346 004000 CMP TEMP,#4000  
1615 004162 002404 BLT 2$  
1616 004164 005237 001420 INC MAX  
1617 004170 010237 001414 MOV R2,MIN  
1618 004174 020227 000200 2$: CMP R2,#200  
1619 004200 001003 BNE 3$  
1620 004202 013737 001346 004274 MOV TEMP,4$  
1621 004210 013703 001346 3$: MOV TEMP,R3 ;SAVE VALUE  
1622 004214 160437 001346 SUB R4,TEMP ;TEMP=DIFF. BETWEEN VALUE&PREVIOUS  
1623 004220 010304 MOV R3,R4 ;SET UP PREVIOUS VALUE FOR NEXT TIME THRU  
1624 004222 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS  
1625 004226 000006 6 ;WITH 15 MILLIVOLTS(1 DAC LSB)  
1626 004230 011724 V5  
1627 004232 104004 ERROR 4  
1628 004234 005202 INC R2  
1629 004236 020227 000400 CMP R2,#400 ;DONE?  
1630 004242 001326 BNE 1$ ;NO-DO NEXT VERNIER DAC VALUE  
1631 004244 004737 020224 JSR PC,$RESET  
1632 004250 052777 000100 174666 BIS #100,@$TKS  
1633 004256 004737 004654 JSR PC,DAWAIT ;LET DAC SETTLE  
1634 004262 004537 011060 JSR R5,CONVRT ;CONVERT IT  
1635 004266 000012 12  
1636 004270 004537 011302 JSR R5,COMPAR ;COMPARE RESULTS  
1637 004274 000000 0 4$: 0  
1638 004276 011710 V2  
1639 004300 104004 ERROR 4
```



```

1641
(3)
(3)
(2) 004302 000004
(1) 004304 012737 000010 001160
1642 004312 004537 011060
1643 004316 000013
1644 004320 004537 011302
1645 004324 006000
1646 004326 011716
1647 004330 104004
1648
(3)
(3)
(2) 004332 000004
(1) 004334 012737 000010 001160
1649 004342 004537 011060
1650 004346 000017
1651 004350 004537 011302
1652 004354 007146
1653 004356 011722
1654 004360 104004
1655
(3)
(3)
(2) 004362 000004
(1) 004364 012737 000001 001160
1656 004372 013737 001332 001362
1657 004400 013737 001332 001360
1658 004406 012737 004001 001410
1659 004414 004537 006440
1660 004420 000062
1661 004422 013737 001404 001346
1662 004430 004537 006440
1663 004434 000062
1664 004436 063737 001404 001346
1665 004444 162737 000062 001346
1666 004452 013700 001414
1667 004456 006300
1668 004460 160037 001346
1669 004464 104401 013705
1670 004470 013702 001346
1671 004474 004737 011472
1672 004500 104401 013720
1673 004504 004537 011302
1674 004510 000000
1675 004512 011726
1676 004514 000401
1677 004516 000403
1678 004520 104401 012507
1679 004524 000402
1680 004526 104401 012476
    
```

```

*****
:*TEST 21 TEST CH13 +2.5 VOLTS
*****
TST21: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
JSR R5,CONVRT ;CONVERT 8 TIMES
13
JSR R5,COMPAR ;COMPARE RESULTS
6000 ;NOMINAL
V144 ;TOLERANCE
ERROR 4

*****
:*TEST 22 TEST CH17 +4V
*****
TST22: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
JSR R5,CONVRT ;CONVERT 8 TIMES
17 ;CHANNEL 17
JSR R5,COMPAR ;COMPARE RESULTS
7146 ;NOMINAL
V240 ;TOLERANCE
ERROR 4 ;ERROR ON A/D CHANNEL

*****
:*TEST 23 OFFSET ON CHO
*****
TST23: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV BASECH,CHANL ;LOAD CHANNEL
MOV BASECH,DUMMY ;LOAD DUMMY
MOV #4001,EDGE
JSR R5,SARSUB
50.
MOV DAC,TEMP
JSR R5,SARSUB
50.
ADD DAC,TEMP
SUB #62,TEMP
MOV MIN,R0
ASL R0
SUB R0,TEMP
TYPE ,MOFSET ;TYPE ASCIZ STRING
MOV TEMP,R2
JSR PC,DECTYP
TYPE ,MLSB ;TYPE ASCIZ STRING
JSR R5,COMPAR ;IS RESULT WITHIN LIMITS?
0
V50D
BR OFFERR ;NO-ERROR
BR OFFOK ;YES-OK
OFFERR: TYPE ,ERMSG
BR TST24 ;;GO TO NEXT TEST
OFFOK: TYPE ,OKMSG
    
```

1682  
(3)  
(3)  
(2) 004532 000004  
(1) 004534 012737 000001 001160  
1683 004542 012737 000116 001346  
1684 004550 004537 010652  
1685 004554 000015  
1686 004556 004537 010652  
1687 004562 000007  
1688 004564 004537 010652  
1689 004570 000016  
1690  
1691  
(3)  
(3)  
(2) 004572 000004  
(1) 004574 012737 000001 001160  
1692 004602 004537 006110  
1693 004606 000015  
1694 004610 000016  
1695 004612 012737 000116 001346  
1696 004620 004537 006110  
1697 004624 000016  
1698 004626 000015  
1699  
(3)  
(3)  
(2) 004630 000004  
(1) 004632 012737 000001 001160  
1700 004640 005737 001202  
1701 004644 001402  
1702 004646 004737 006736  
1703 004652 000207  
1704  
1705 004654 005000  
1706 004656 105300  
1707 004660 001376  
1708 004662 000207

```
*****  
:*TEST 24 NOISE TEST ON 8 EDGES  
*****  
TST24: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
MOV #116,TEMP ;DAC VALUE  
JSR R5,NOI8 ;NOISE AT -FULL SCALE  
15  
JSR R5,NOI8 ;NOISE AT MID-RANGE  
7  
JSR R5,NOI8 ;NOISE AT +FULL SCALE  
16  
  
*****  
:*TEST 25 SETTLE TEST ON 8 EDGES  
*****  
TST25: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
JSR R5,SET8 ;SETTLE-POSITIVE DIRECTION  
15  
16  
MOV #116,TEMP  
JSR R5,SET8 ;SETTLE-NEGATIVE DIRECTION  
16  
15  
  
*****  
:*TEST 26 DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST  
*****  
TST26: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
TST $PASS ;FIRST TIME-SKIP DIPLIN  
BEQ LEND  
JSR PC,DIPLIN  
LEND: RTS PC ;RETURN TO TEST SECTION  
  
DAWAIT: CLR R0  
1$: DECB R0  
BNE 1$  
RTS PC
```



```

1710 .SBTTL CALIBRATION TEST
1711 004664 012737 004664 001364 BEGINC: MOV #BEGINC,TADDR ;TEST ADDRESS IN TADDR
1712 004672 005037 001426 CLR MYTEMP
(2)
(2) ;* MOV MYTEMP,@STREG ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1713 004706 104401 013615 TYPE ,HEADS ;TYPE OUT HEADING
1714 004712 005037 177776 CLR PSW
1715 004716 017700 174216 1$: MOV @SWR,R0 ;READ CHANNEL FROM SWITCH REG.
1716 004722 042700 177700 BIC #177700,R0 ;ISOLATE MUX BITS
1717 004726 032777 020000 174204 BIT #BIT13,@SWR ;IS BIT 13 SET?
1718 004734 001005 BNE 2$ ;;YES,SKIP TYPEOUT
1719 004736 104401 012321 TYPE ,CH
1720 004742 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
(1) ;;TYPE CHANNEL
(1) 004744 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 004746 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
(1) 004747 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1721 004750 2$: SWAB R0 ;SWITCH BYTES
1722 004750 000300 MOV R0,MYTEMP
1723 004752 010037 001426
(2)
(2) ;* MOV MYTEMP,@STREG ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1724 004766 012702 000010 MOV #10,R2 ;TYPEOUT COUNTER
1725 004772 3$:
(1)
(2)
(2) ;* MOV @STREG,MYTEMP ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(1) 005002 005237 001426 INC MYTEMP
(2)
(2) ;* MOV MYTEMP,@STREG ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1726 005016 30$:
1727
(2) ;* MOV @STREG,MYTEMP ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(1) 005026 105737 001426 TSTB MYTEMP
1728 005032 100371 BPL 30$
1729
(2) ;* MOV @ADBUFF,MYTEMP ;/READ DEVICE REG ADBUFF,PUT DATA IN MYTEMP.
(1) 005044 013700 001426 MOV MYTEMP,R0 ;/PUT CONVERTED VALUE IN R0.
1730 005050 032777 020000 174062 BIT #BIT13,@SWR ;IS BIT 13 SET?
1731 005056 009403 BEQ 4$ ;NOT SET, TYPE OUT LIST
1732 005060 010077 174056 MOV R0,@DISPLAY ;PUT VALUE IN DISPLAY FOR DISPLAY CONTRO
1733 005064 000714 BR 1$ ;REPEAT CONVERSION
1734 005066 104401 012324 4$: TYPE ,SPACE
1735 005072 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
(1) ;;PRINT OCTAL CONVERTED VALUE
(1) 005074 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 005076 004 .BYTE 4 ;;TYPE 4 DIGIT(S)
(1) 005077 001 .BYTE 1 ;;TYPE LEADING ZEROS
1736 005100 012701 010000 5$: MOV #10000,R1
1737 005104 005301 DEC R1
1738 005106 001376 BNE 5$ ;DECREMENT THE COUNTER
1739 005110 005302 DEC R2 ;NO CARRIAGE RETURN
1740 005112 001327 BNE 3$ ;CARRIAGE RETURN
1741 005114 104401 001171 TYPE ,$CRLF
1742 005120 000676 BR 1$ ;REPEAT CONVERSION

```

Line	Address	Code	Label	Op	Opnd	Comment	
1744			.SBTTL			LOGIC TEST SECTION	
1745	005122	012737	005122	001364	BEGL: MOV	#BEGL,TADDR ;TEST ADDRESS	
1746	005130	004737	002550		JSR	PC,TESTAD ;NO OF ADDITIONAL AD'S	
1747	005134	004737	002716		1\$: JSR	PC,BEGINL ;LOGIC TESTS	
1748	005140	004737	005310		JSR	PC,BUMPAD ;MORE TO TEST?	
1749	005144	000773			BR	1\$ ;TEST NEXT A/D	
1750	005146	012737	005134	012004	MOV	#1\$,AGTST ;ADDRESS FOR EOP	
1751	005154	000137	012006		JMP	\$EOP ;TYPE END OF PASS	
1752							
1753			.SBTTL			AUTO TEST	
1754	005160	012737	005160	001364	BEGINA: MOV	#BEGINA,TADDR ;TEST ADDRESS	
1755	005166	005037	001202		CLR	\$PASS ;CLEAR PASS COUNTER	
1756	005172	004737	002550		JSR	PC,TESTAD ;NO. OF AD'S TO BE TESTED	
1757	005176	004737	002716		1\$: JSR	PC,BEGINL ;LOGIC TESTS	
1758	005202	104401	012777		TYPE	,MEND ;TYPE END OF LOGIC TEST	
1759	005206	013746	001316		MOV	STREG,-(SP) ;SAVE STREG FOR TYPEOUT	
1760	005212	104403			TYPOS		;TYPE OCTAL NUMBER
1761	005214	006			.BYTE	6 ;TYPE 6 DIGITS	
1762	005215	001			.BYTE	1 ;TYPE LEADING ZEROS	
1763	005216	104401	001171		TYPE	,\$CRLF ;TYPE A CR,LF	
1764	005222	004737	003444		JSR	PC,WRAP ;TEST NEXT A/D	
1765	005226	004737	005310		JSR	PC,BUMPAD ;TEST NEXT AD	
1766	005232	000761			BR	1\$ ;ADDRESS FOR EOP	
1767	005234	012737	005176	012004	MOV	#1\$,AGTST ;TYPE END OF PASS	
1768	005242	000137	012006		JMP	\$EOP	
1769							
1770			.SBTTL			WRAPAROUND TEST	
1771	005246	012737	005246	001364	BEGINW: MOV	#BEGINW,TADDR ;TEST ADDRESS	
1772	005254	005037	001202		CLR	\$PASS ;CLEAR PASS COUNT	
1773	005260	004737	002550		JSR	PC,TESTAD ;NO. OF AD'S TO BE TESTED	
1774	005264	004737	003444		1\$: JSR	PC,WRAP ;WRAPAROUND TESTS	
1775	005270	004737	005310		JSR	PC,BUMPAD ;MORE A/D'S TO BE TESTED?	
1776	005274	000773			BR	1\$ ;YES-GO TEST NEXT AD11K	
1777	005276	012737	005264	012004	MOV	#1\$,AGTST	
1778	005304	000137	012006		JMP	\$EOP ;INCREMENTS \$PASS	



```

1780          ; DETERMINE IF MORE AD11K'S TO BE TESTED
1781 005310 005737 001354      ;BUMPAD: TST      NBEXT      ;ADDITIONAL AD'S?
1782 005314 001421              BEQ      FIXADR      ;NO-INITIALIZE ADDRESSES
1783 005316 063737 001326 001316 ADD      VADR,STREG    ;SET UP NEW ST. REG.
1784 005324 063737 001326 001320 ADD      VADR,ADBUFF   ;SET UP NEW BUFFER ADDRESS
1785 005332 063737 001330 001454 ADD      VVCT,VECTOR   ;SET UP NEW VECTOR
1786 005340 063737 001330 001324 ADD      VVCT,VECTR1
1787 005346 005077 173752      CLR      @VECTR1
1788 005352 005337 001354      DEC      NBEXT      ;ONE LESS AD11K
1789 005356 000441              BR       BYPASS
1790 005360 062716 000002      FIXADR: ADD     #2,(SP)
1791 005364 013737 001250 001316 FIXONE: MOV     $BASE,STREG ;RELOAD INITIAL ADDRESSES
1792 005372 013737 001250 001320 MOV     $BASE,ADBUFF
1793 005400 062737 000002 001320 ADD     #2,ADBUFF
1794 005406 013737 001244 001454 MOV     $VECT1,VECTOR
1795 005414 042737 170000 001454 BIC     #170000,VECTOR
1796 005422 113737 001245 001322 MOV     $VECT1+1,BASEBR
1797 005430 105037 001323      CLR     BASEBR+1 ;CLEAR HIGH BYTE
1798 005434 013737 001454 001324 MOV     VECTOR,VECTR1
1799 005442 062737 000002 001324 ADD     #2,VECTR1
1800 005450 005077 173650      CLR     @VECTR1
1801 005454 013737 001356 001354 MOV     NMBEXT,NBEXT ;RESET COUNTER
1802          ;:LOAD .+2 AND HALT TRAP CATCH;;
1803 005462 012700 000216      BYPASS: MOV     #216,R0 ;FILL .+2
1804 005466 012701 000214      MOV     #214,R1 ;LOAD HALT
1805 005472 020137 001334      1$:  CMP     R1,KBVECT
1806 005476 001410              BEQ     2$
1807 005500 010021              MOV     R0,(R1)+
1808 005502 005021              CLR     (R1)+
1809 005504 010100              MOV     R1,R0
1810 005506 005720              TST     (R0)+
1811 005510 020027 001002      CMP     R0,#1002
1812 005514 001366              BNE     1$
1813 005516 000207              RTS     PC ;TEST NEXT A/D
1814 005520 022021      2$:  CMP     (R0)+,(R1)+
1815 005522 022021      CMP     (R0)+,(R1)+
1816 005524 000762      BR      1$
1817
1818
1819          ; NOISE TEST, 1 EDGE
1820 005526 012737 005526 001364 ;BEGINN: MOV     #BEGINN,TADDR ;TEST ADDRESS IN TADDR
1821 005534 104401 012130      TYPE     ,NOIMSG ;ASK FOR CHANNEL
1822 005540 104401 013634      TYPE     ,ASKCH
1823 005544 017737 173370 001350 1$:  MOV     @SWR,CH1 ;LOAD CHANNEL
1824 005552 042737 177700 001350 BIC     #177700,CH1
1825 005560 012737 000200 001346 MOV     #200,TEMP ;LOAD DAC VALUE
1826 005566 004537 010366      JSR     R5,NOITST ;GO TO NOISE SUBROUTINE
1827 005572 001350      CH1
1828 005574 000763      BR      1$

```

```

1830 ; INTERCHANNEL SETTLING TEST, 1 EDGE
1831 005576 012737 005576 001364 ; BEGINS: MOV #BEGINS,TADDR ; TEST ADDRESS IN TADDR
1832 005604 104401 012150 ; TYPE ,SETMSG ; ASK FOR CHANNELS
1833 005610 104410 RDOCT
1834 005612 012637 001350 MOV (SP)+,CH1
1835 005616 104401 012435 ; TYPE ,TOMSG
1836 005622 104410 RDOCT
1837 005624 012637 001352 MOV (SP)+,CH2
1838 005630 012737 000200 001346 BK3: MOV #200,TEMP ; LOAD DAC
1839 005636 013737 001352 001362 MOV CH2,CHANL
1840 005644 004737 006214 JSR PC,GETEDG ; GET EDGE VALUES
1841 005650 005002 CLR R2
1842 005652 004737 006046 JSR PC,SET1A ; SCALING = .02 LSB
1843 005656 004737 006046 JSR PC,SET1A ; MAKE IT .01 LSB
1844 005662 100001 BPL POSR2
1845 005664 005402 NEG R2
1846 005666 010204 POSR2: MOV R2,R4
1847 005670 012737 000001 006436 MOV #1,EDGFLG
1848 005676 004737 005704 JSR PC,TYPSET
1849 005702 000752 BR BK3
1850 005704 004737 011472 TYPSET: JSR PC,DECTYP
1851 005710 104401 012331 ; TYPE ,LSB
1852 005714 013746 001352 MOV CH2,-(SP) ;::SAVE CH2 FOR TYPEOUT
(1) ;::TYPE CH
(1) 005720 104403 ;::GO TYPE--OCTAL ASCII
(1) 005722 002 ;::TYPE 2 DIGIT(S)
(1) 005723 000 ;::SUPPRESS LEADING ZEROS
1853 005724 104401 013726 ;::TYPE ASCII STRING
1854 005730 004737 006374 JSR PC,TYPEDG
1855 005734 104401 012344 ; TYPE ,SETCH
1856 005740 013746 001350 MOV CH1,-(SP) ;::SAVE CH1 FOR TYPEOUT
(1) ;::TYPE CH
(1) 005744 104403 ;::GO TYPE--OCTAL ASCII
(1) 005746 002 ;::TYPE 2 DIGIT(S)
(1) 005747 000 ;::SUPPRESS LEADING ZEROS
1857 005750 104401 012366 ; TYPE ,ATMSG
1858 005754 013737 001350 006012 MOV CH1,1$
1859 005762 163737 001332 006012 SUB BASECH,1$
1860 005770 012737 000200 001426 MOV #200,MYTEMP
1861 (1) ;* MOV MYTEMP,@ADBUFF ;/ PUT DATA FROM MYTEMP TO DEVICE REG ADBUFF
1862 006006 004537 011060 ;* JSR R5,CONVRT
1863 006012 000000 1$: 0
1864 006014 013746 001346 MOV TEMP,-(SP) ;::SAVE TEMP FOR TYPEOUT
(1) ;::TYPE VALUE
(1) 006020 104403 ;::GO TYPE--OCTAL ASCII
(1) 006022 004 ;::TYPE 4 DIGIT(S)
(1) 006023 001 ;::TYPE LEADING ZEROS
1865 006024 020437 011734 CMP R4,VSET
1866 006030 003003 BGT ERR
1867 006032 104401 012476 ; TYPE ,OKMSG
1868 006036 000207 RTS PC

```



```

1870 006040 104401 012507      ERR:  TYPE  ,ERMSG
1871 006044 000207              RTS    PC
1872
1873
1874
1875      ;; SUBROUTINE FOR SETTLING TESTS;;
1876 006046 013737 001352 001360 SET1A: MOV    CH2,DUMMY      ;LOAD DUMMY
1877 006054 004537 006440          JSR    R5,SARSUB      ;DO SAR ROUTINE AT 50%
1878 006060 000062                    50.
1879 006062 063702 001404          ADD    DAC,R2        ;ADD RESULT TO R2
1880 006066 013737 001350 001360 MOV    CH1,DUMMY      ;CHANGE DUMMY VALUE
1881 006074 004537 006440          JSR    R5,SARSUB      ;DO SAR ROUTINE AT 50%
1882 006100 000062                    50.
1883 006102 163702 001404          SUB    DAC,R2        ;SUBTRACT RESULT FROM R2
1884 006106 000207              RTS    PC            ;RETURN
1885
1886 006110 012537 001350          SET8:  MOV    (R5)+,CH1      ;GET FIRST CHANNEL
1887 006114 012537 001352          MOV    (R5)+,CH2      ;GET SECOND CHANNEL
1888 006120 063737 001332 001350 ADD    BASECH,CH1
1889 006126 063737 001332 001352 ADD    BASECH,CH2
1890 006134 004737 006214          JSR    PC,GETEDG      ;GET EDGE VALUES
1891 006140 005002                    CLR    R2
1892 006142 012703 000010          MOV    #10,R3        ;SET UP COUNTER
1893 006146 004737 006046          SETAA: JSR    PC,SET1A   ;GET SETTLE VALUES
1894 006152 005237 001410          INC    EDGE
1895 006156 005303                    DEC    R3
1896 006160 001372                    BNE    SETAA          ;REPEAT 8 TIMES
1897 006162 162737 000010 001410 SUB    #10,EDGE
1898 006170 005702                    TST    R2
1899 006172 100001                    BPL    R2POS
1900 006174 005402                    NEG    R2
1901 006176 010204                    R2POS: MOV    R2,R4
1902 006200 012737 000010 006436 MOV    #8.,EDGFLG
1903 006206 004737 005704          JSR    PC,TYPSET     ;TYPE OUT RESULTS
1904 006212 000205              RTS    R5            ;RETURN
1905
1906
1907      ; SUBROUTINE TO GET EDGE VALUE
1908      ; CALL=JSR PC,GETEDG
1909      ; CONVERSIONS ON A/D CHANNEL 'CHANL'
1910      ; RESULT IN EDGE, USES R0
1911 006214      GETEDG:
1912      (1)
1913      (1)      ; * MOV    TEMP,@ADBUFF    ; / PUT DATA FROM TEMP TO DEVICE REG ADBUFF
1914      006224 113700 001362      MOVB   CHANL,R0      ; GET CHANNEL
1915      006230 000300          SWAB   R0            ; SET UP A.D STATUS REG.
1916      006232 010037 001426      MOV    R0,MYTEMP
1917      (2)
1918      (2)      ; * MOV    MYTEMP,@STREG    ; / PUT DATA FROM MYTEMP TO DEVICE REG STREG
1919      006246 012700 000100      MOV    #100,R0      ; DAC SETTLING DELAY
1920      006252 005300          1$:   DEC    R0
1921      006254 001376          BNE    1$
1922      006256 005037 001410          CLR    EDGE
1923      006262 012700 000010          MOV    #10,R0
1924      006266          CONV:

```

```

(2)
(2)
(1) 006276 005237 001426      ;*   MOV   @STREG,MYTEMP  ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(2)                               INC   MYTEMP
(2)
(2)                               ;*   MOV   MYTEMP,@STREG  ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
1921 006312      30$:
(2)
(2)                               ;*   MOV   @STREG,MYTEMP  ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(1) 006322 105737 001426      TSTB  MYTEMP
1922 006326 100371      BPL   30$
1923
(2)
(2)                               ;*   MOV   @ADBUFF,MYTEMP ;/READ DEVICE REG ADBUFF,PUT DATA IN MYTEMP.
(1) 006340 063737 001426 001410 ADD   MYTEMP,EDGE
1924 006346 005300      DEC   R0
1925 006350 001346      BNE   CONV
1926 006352 006237 001410      ASR   EDGE
1927 006356 006237 001410      ASR   EDGE
1928 006362 006237 001410      ASR   EDGE
1929 006366 005537 001410      ADC   EDGE
1930 006372 000207      RTS   PC
1931      ;;SUBROUTINE TO TYPE EDGE VALUES;;
1932 006374 013703 001410      TYPEDG: MOV  EDGE,R3
1933 006400 010346      MOV  R3,-(SP)      ;;SAVE R3 FOR TYPEOUT
(1)                               ;;TYPE OCTAL VALUE OF EDGE
(1) 006402 104403      TYPOS      ;;GO TYPE--OCTAL ASCII
(1) 006404      004      .BYTE 4      ;;TYPE 4 DIGIT(S)
(1) 006405      001      .BYTE 1      ;;TYPE LEADING ZEROS
1934 006406 023727 006436 000001  CMP   EDGFLG,#1
1935 006414 001407      BEQ   RET
1936 006416 062703 000007      ADD   #7,R3
1937 006422 104401 013676      TYPE  ,C1      ;TYPE ASCII STRING
1938 006426 010346      MOV  R3,-(SP)      ;;SAVE R3 FOR TYPEOUT
(1)                               ;;TYPE EDGE VALUE
(1) 006430 104403      TYPOS      ;;GO TYPE--OCTAL ASCII
(1) 006432      004      .BYTE 4      ;;TYPE 4 DIGIT(S)
(1) 006433      001      .BYTE 1      ;;TYPE LEADING ZEROS
1939 006434 000207      RET:  RTS   PC
1940 006436 000000      EDGFLG: 0
  
```



```

1942 ;SUBROUTINE TO DO SUCCESSIVE APPROXIMATION ROUTINE
1943 ;CALL=JSR R5,SARSUB
1944 ; XXX;XXX=PERCENT
1945 ;RESULT RETURNED IN 'DAC',USES R0,R1,R4
1946 006440 012537 001422 SARSUB: MOV (R5)+,PERCNT ;GET PERCENT
1947 006444 006337 001422 ASL PERCNT
1948 006450 006337 001422 ASL PERCNT
1949 006454 012737 000620 006734 MOV #400,CNNO ;NO OF SAMPLES FOR SHORT PASS.
1950 006462 032777 004000 172450 BIT #BIT11,@SWR ;USER WANT SHORT PASS?
1951 006470 001010 BNE SAR1
1952 006472 000407 BR SAR1 ;ALWAYS USE SHORT SAMPLE COUNT.
1953 006474 012737 003100 006734 MOV #1600.,CNNO
1954 006502 006337 001422 ASL PERCNT ;RESCALE PERCENT FOR 1600.
1955 006506 006337 001422 ASL PERCNT ;POINTS PER BURST
1956 006512 012737 000200 001412 SAR1: MOV #200,BITPNT ;INITIALIZE BIT POINTER AT MSB
1957 006520 005037 001404 CLR DAC ;INITIALIZE DAC VALUE
1958 006524 004537 020542 ! JSR R5,$PUTS
1959 006530 001316 .WORD STREG
1960 006532 005000 TRY: CLR R0
1961 006534 063737 001412 001404 ADD BITPNT,DAC ;TRY BIT
1962 (1)
1963 006552 012737 000100 001406 ;* MOV DAC,@ADBUFF ;/ PUT DATA FROM DAC TO DEVICE REG ADBUFF
1964 006560 005337 001406 TS: MOV #100,DELAY
1965 006564 001375 BNE DELAY ;STALL TIME
1966 006566 013701 006734 MOV CNNO,R1 ;SET UP FOR 1600. OR 400. CONVERSIONS
1967 006572 113737 001362 001433 MOVB CHANL,$TEMP2+1
1968 006600 052737 000001 001432 BIS #1,$TEMP2
1969 006606 113737 001360 001431 MOVB DUMMY,$TEMP1+1
1970 006614 052737 000001 001430 BIS #1,$TEMP1
1971 006622
1972 006622 013777 001430 172614 NXTCVT: MOV $TEMP1,@KMAD4
1973 006630 112777 000006 172602 MOVB #6,@KMAD2
1974 006636 122777 000377 172574 10$: CMPB #377,@KMAD2
1975 006644 001374 BNE 10$
1976 006646 013777 001432 172570 MOV $TEMP2,@KMAD4
1977 006654 112777 000006 172556 MOVB #6,@KMAD2
1978 006662 122777 000377 172550 20$: CMPB #377,@KMAD2
1979 006670 001374 BNE 20$
1980 006672 027737 172546 001410 CMP @KMAD4,EDGE
1981 006700 002001 BGE 2$
1982 006702 005200 INC R0 ;COUNT RESULTS .LT. EDGE
1983 006704 005301 2$: DEC R1
1984 006706 001345 BNE NXTCVT
1985 006710 020037 001422 CMP R0,PERCNT
1986 006714 003003 BGT SHIFT
1987 006716 163737 001412 001404 SUB BITPNT,DAC ;TAKE THE BIT OUT
1988 006724 006237 001412 SHIFT: ASR BITPNT
1989 006730 001300 BNE TRY
1990 006732 000205 RTS R5
1991
1992 006734 000000 CNNO: .WORD 0

```

```

1994      ;:DIFFERENTIAL LINEARITY SUBROUTINE::
1995 006736 104401 013122  DIFLIN: TYPE      ,MSG20
1996 006742 005037 001424      CLR      OUT
1997 006746 012700 042300      MOV      #BUFFER,R0
1998 006752 012701 010000      MOV      #4096.,R1      ;4096 WORDS FOR HISTOGRAM
1999 006756 005020      CLEAR1: CLR      (R0)+      ;CLEAR BUFFER AREA
2000 006760 005301      DEC      R1
2001 006762 001375      BNE      CLEAR1
2002 006764 012700 021622      MOV      #DIST,R0      ;DISTRIBUTION BUFFER POINTER
2003 006770 012701 000310      MOV      #200.,R1      ;200. WORDS FOR DISTRIBUTION
2004 006774 005003      CLR      R3
2005 006776 005037 001424      CLR      OUT
2006 007002 005037 001336      CLR      WIDE
2007 007006 005037 001340      CLR      NARROW
2008 007012 005037 001342      CLR      FIRST
2009 007016 005037 001344      CLR      SKIPST
2010 007022 005020      CLEAR2: CLR      (R0)+      ;CLEAR DISTRIBUTION BUFFER AREA
2011 007024 005301      DEC      R1
2012 007026 001375      BNE      CLEAR2
2013 007030 012700 000011      CHANNL: MOV      #11,R0      ;CHANNEL 11
2014 007034 063700 001332      ADD      BASECH,R0
2015 007040 000300      SWAB     R0      ;LOAD MUX BITS
2016 007042 004537 020542      JSR      R5,$PUTS
2017 007046 001316      .WORD   STREG
2018 007050 010037 001426      MOV      R0,MYTEMP
(2)
(2)
2019 007064 010037 001430      ;*      MOV      MYTEMP,@STREG      ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
2020 007070 052737 000001 001430      MOV      R0,$TEMP1
2021 007076 012700 001440      BIS      #1,$TEMP1
2022 007102 012777 001702 172344      MOV      #800.,R0      ;NOMINAL STATE WIDTH - 1 LSB
2023 007110 012701 007776      MOV      #RETURN,@VECTOR
2024 007114 004737 010776      AGAIN:  MOV      #4094.,R1
2025 007120 013702 001366      NEXT:   JSR      PC,RANDY      ;GET RANDOM NUMBER
2026 007124 042702 177760      MOV      RNA,R2
2027 007130 001402      BIC      #177760,R2      ;MASK IT TO 4 BITS ONLY
2028 007132 005302      BEQ     CONVR
2029 007134 001376      DELAY3: DEC      R2      ;STALL
2030 007136      BNE     DELAY3      ;TIME
2031 007136 013777 001430 172300      CONVR:  MOV      $TEMP1,@KMAD4
2032 007144 112777 000006 172266      $TBF4:  MOV      #6,@KMAD2
2033 007152 122777 000377 172260      31$:   CMPB     #377,@KMAD2
2034 007160 001374      BNE     31$
2035 007162 017702 172256      MOV      @KMAD4,R2
2036 007166 001413      BEQ     DELAY1      ;IGNORE IF =0
2037 007170 020227 007777      CMP      R2,#7777      ;IGNORE IF =7777
2038 007174 001413      BEQ     DELAY2
2039 007176 006302      ASL     R2
2040 007200 005262 042300      INC     BUFFER(R2)      ;MAKE HISTOGRAM
2041 007204 100013      BPL     OKAY
2042 007206 012762 077777 042300      MOV      #077777,BUFFER(R2)      ;PREVENT OVERFLOW
2043 007214 000407      BR      OKAY
2044 007216 020227 007777      DELAY1: CMP      R2,#7777      ;EQUALIZE LOOP TIME
2045 007222 001400      BEQ     DELAY2      ;WITH DUMMY INSTR.
2046 007224 005201      DELAY2: INC      R1
2047 007226 005263 001346      INC     TEMP(R3)

```



2048	007232	100403							
2049	007234	005301		OKAY:	DEC	R1			
2050	007236	001326			BNE	NEXT			
2051	007240	000403			BR	AROUND			
2052	007242	005037	001346	NOTOK:	CLR	TEMP			
2053	007246	000772			BR	OKAY			
2054	007250	005300		AROUND:	DEC	R0			
2055	007252	001316			BNE	AGAIN			
2056					;DATA COLLECTION HAS NOW BEEN COMPLETED - WORK ON THE DATA COLLECTED				
2057	007254	012700	007776		MOV	#4094.,R0			
2058	007260	012701	042302		MOV	#BUFFER+2,R1			
2059	007264	012102		READ:	MOV	(R1)+,R2		;GET STATE WIDTH	
2060	007266	006202			ASR	R2		;1 LSB = 800.	
2061	007270	006202			ASR	R2			
2062	007272	006202			ASR	R2			
2063	007274	005502			ADC	R2		;1 LSB = 100.	
2064	007276	020227	000310		CMP	R2,#200.		;OUT OF RANGE?	
2065	007302	002403			BLT	INRNGE			
2066	007304	005237	001424		INC	OUT		;YES - INCREMENT COUNTER	
2067	007310	000423			BR	TYPBAD			
2068	007312	006302		INRNGE:	ASL	R2			
2069	007314	005262	021622		INC	DIST(R2)		;MAKE STATE WIDTH DISTRIBUTION	
2070	007320	006202			ASR	R2			
2071	007322	020227	000062		CMP	R2,#50.		;IS IT 1/2 LSB?	
2072	007326	002007			BGE	NOTNAR			
2073	007330	005237	001340		INC	NARROW			
2074	007334	005702			TST	R2		;IS IT A SKIPPED STATE?	
2075	007336	001002			BNE	31\$			
2076	007340	005237	001344		INC	SKIPST			
2077	007344	000405		31\$:	BR	TYPBAD			
2078	007346	020227	000226	NOTNAR:	CMP	R2,#150.		;IS IT 1.5 LSB?	
2079	007352	003426			BLE	LAST			
2080	007354	005237	001336		INC	WIDE			
2081	007360	005737	001342	TYPBAD:	TST	FIRST			
2082	007364	001004			BNE	60\$			
2083	007366	005237	001342		INC	FIRST			
2084	007372	104401	012301		TYPE	,STATE			
2085	007376	010103		60\$:	MOV	R1,R3			
2086	007400	162703	042302		SUB	#BUFFER+2,R3			
2087	007404	006203			ASR	R3			
2088	007406	010346			MOV	R3,-(SP)		::SAVE R3 FOR TYPEOUT	
(1)								::TYPE STATE	
(1)	007410	104403		TYPOS				::GO TYPE--OCTAL ASCII	
(1)	007412	004		.BYTE	4			::TYPE 4 DIGIT(S)	
(1)	007413	001		.BYTE	1			::TYPE LEADING ZEROS	
2089	007414	104401	012275	TYPE	,DASH				
2090	007420	004737	011472	JSR	PC,DECTYP				
2091	007424	104401	012266	TYPE	,LSBMSG				
2092	007430	005300		LAST:	DEC	R0			
2093	007432	001314			BNE	READ			
2094	007434	112737	000177	014564	MOVB	#177,DECPNT			
2095	007442	013702	001344		MOV	SKIPST,R2		;GET NO. OF SKIPPED STATES	
2096	007446	004737	011472		JSR	PC,DECTYP		;TYPE IT	
2097	007452	104401	012524		TYPE	,SKPMSG		;TYPE MESSAGE	
2098	007456	005737	001344		TST	SKIPST			
2099	007462	001403			BEQ	1\$			

2100	007464	104401	012507		TYPE	,ERMSG	;TYPE 'ERROR'
2101	007470	000402			BR	NAR	
2102	007472	104401	012476	1\$:	TYPE	,OKMSG	;TYPE #OK#
2103	007476	013702	001340	NAR:	MOV	NARROW,R2	;GET NO. OF NARROW STATES
2104	007502	004737	011472		JSR	PC,DECTYP	;TYPE IT
2105	007506	104401	012546		TYPE	,NARMSG	;TYPE MESSAGE
2106	007512	013702	001336		MOV	WIDE,R2	
2107	007516	063702	001424		ADD	OUT,R2	
2108	007522	004737	011472		JSR	PC,DECTYP	;TYPE NO. OF WIDE STATES
2109	007526	104401	012605		TYPE	,WIDMSG	;TYPE MESSAGE
2110	007532	013702	001424		MOV	OUT,R2	
2111	007536	004737	011472		JSR	PC,DECTYP	;TYPE NO. OF STATES OUTSIDE 2 LSB
2112	007542	104401	012644		TYPE	,OUTMSG	;TYPE MESSAGE
2113	007546	005737	001424		TST	OUT	
2114	007552	001403			BEQ	11\$	
2115	007554	104401	012507		TYPE	,ERMSG	;TYPE 'ERROR'
2116	007560	000402			BR	HALF	
2117	007562	104401	012476	11\$:	TYPE	,OKMSG	;TYPE 'OK'
2118	007566	013702	001340	HALF:	MOV	NARROW,R2	
2119	007572	063702	001336		ADD	WIDE,R2	
2120	007576	063702	001424		ADD	OUT,R2	
2121	007602	010200			MOV	R2,R0	
2122	007604	004737	011472		JSR	PC,DECTYP	;TYPE NO. OF STATES OUTSIDE LIMITS
2123	007610	112737	000056	014564	MOVB	#56,DECPNT	
2124	007616	104401	012677		TYPE	,HAFMSG	
2125	007622	020027	000051		CMP	R0,#41.	;COMPARE IT TO NOMINAL
2126	007626	003403			BLE	21\$	
2127	007630	104401	012507		TYPE	,ERMSG	;TYPE 'ERROR'
2128	007634	000402			BR	SWDIST	
2129	007636	104401	012476	21\$:	TYPE	,OKMSG	;TYPE 'OK'
2130	007642	005737	001400	SWDIST:	TST	FLAG	;VT55?
2131	007646	001426			BEQ	RELACC	
2132	007650	004737	010330		JSR	PC,DELCLR	;WAIT AWHILE, THEN CLEAR VT55
2133	007654	104401	013154		TYPE	,MSG16	
2134	007660	104401	013755		TYPE	,BUFF1	;TYPE BUFF1-PRINT GRID
2135	007664	012700	021622		MOV	#DIST,R0	;POINTER TO STATE WIDTH DISTRIBUTION
2136	007670	012701	000310		MOV	#200.,R1	;GO 200. TIMES UP TO 2 LSB
2137	007674	012002		NXTY1:	MOV	(R0)+,R2	
2138	007676	004737	011370		JSR	PC,LOADY	
2139	007702	005002			CLR	R2	
2140	007704	004737	011370		JSR	PC,LOADY	
2141	007710	005301			DEC	R1	
2142	007712	001370			BNE	NXTY1	
2143	007714	104401	013700		TYPE	,C2	;TYPE ASCIZ STRING
2144	007720	004737	010330		JSR	PC,DELCLR	
2145							



```

2147 ;CHANGE HISTOGRAM ERROR TO RELATIVE ACCURACY ERROR
2148
2149 007724 005001 RELACC: CLR R1 ;RUNNING ERROR = 0
2150 007726 005003 CLR R3 ;MAXIMUM ERROR = 0
2151 007730 104401 013547 TYPE ,MSG21
2152 007734 012700 042302 MOV #BUFFER+2,R0
2153 007740 011002 NXTSTA: MOV (R0),R2 ;STATE WIDTH = R2
2154 007742 162702 001440 SUB #800.,R2 ;STATE WIDTH ERROR IN R2
2155 007746 060201 ADD R2,R1 ;UPDATE RUNNING ERROR
2156 007750 010120 MOV R1,(R0)+ ;SAVE IN BUFFER
2157 007752 010104 MOV R1,R4 ;SAVE IN R4 ALSO
2158 007754 100001 BPL PLUS ;IS IT POSITIVE?
2159 007756 005404 NEG R4 ;NO - MAKE IT POSITIVE
2160 007760 020403 PLUS: CMP R4,R3 ;CHECK AGAINST PREVIOUS MAX. ERROR
2161 007762 003405 BLE NOTNEW ;NOT A NEW MAXIMUM
2162 007764 010403 MOV R4,R3 ;UPDATE MAXIMUM IN R3
2163 007766 010005 MOV R0,R5
2164 007770 162705 042302 SUB #BUFFER+2,R5
2165 007774 006205 ASR R5 ;R5=EDGE VALUE AT MAX. RELACC
2166 007776 020027 062276 NOTNEW: CMP R0,#BUFFER+8190. ;DONE?
2167 010002 001356 BNE NXTSTA ;NO - REPEAT
2168 010004 006203 ASR R3 ;RESCALE FROM 1 LSB = 800. SCALING
2169 010006 006203 ASR R3 ;TO 1 LSB = 100. SCALING
2170 010010 006203 ASR R3
2171 010012 005503 ADC R3
2172 010014 010302 MOV R3,R2
2173 010016 004737 011472 JSR PC,DECTYP
2174 010022 104401 013574 TYPE ,LINEA
2175 010026 010546 MOV R5,-(SP) ;;SAVE R5 FOR TYPEOUT
(1) ;;TYPE VALUE
(1) 010030 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 010032 004 .BYTE 4 ;;TYPE 4 DIGIT(S)
(1) 010033 001 .BYTE 1 ;;TYPE LEADING ZEROS
2176 010034 104401 012433 TYPE ,SLASH ;PRINT '/'
2177 010040 005205 INC R5
2178 010042 010546 MOV R5,-(SP) ;;SAVE R5 FOR TYPEOUT
(1) ;;TYPE VALUE
(1) 010044 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 010046 004 .BYTE 4 ;;TYPE 4 DIGIT(S)
(1) 010047 001 .BYTE 1 ;;TYPE LEADING ZEROS
2179 010050 020337 011736 CMP R3,VLIN
2180 010054 003403 BLE 41$
2181 010056 104401 012507 TYPE ,ERMSG
2182 010062 000402 BR 42$
2183 010064 104401 012476 41$: TYPE ,OKMSG
2184 010070 005737 001400 42$: TST FLAG ;VT55?
2185 010074 001503 BEQ LO2
2186 010076 012700 042300 MOV #BUFFER,R0
2187 010102 012701 010000 MOV #4096.,R1

```

2189	010106	011002		GETDAT:	MOV	(R0),R2		:GET RELATIVE ACCURACY ERROR SCALED 1LSB = 800.
2190	010110	006202			ASR	R2		:RESCALE IT TO 1 LSB = 100.
2191	010112	006202			ASR	R2		
2192	010114	006202			ASR	R2		
2193	010116	005502			ADC	R2		
2194	010120	062702	000166		ADD	#118.,R2		:AND MOVE IT TO MID-SCREEN
2195	010124	010220			MOV	R2,(R0)+		:PUT IT BACK INTO BUFFER
2196	010126	005301			DEC	R1		
2197	010130	001366			BNE	GETDAT		
2198	010132	012700	042300		MOV	#BUFFER,R0		
2199	010136	012704	042300		MOV	#BUFFER,R4		
2200	010142	012705	042302		MOV	#BUFFER+2,R5		
2201	010146	012701	001000		MOV	#512.,R1		
2202	010152	012702	000007	NXT8:	MOV	#7.,R2		
2203	010156	012003			MOV	(R0)+,R3		
2204	010160	010337	001414		MOV	R3,MIN		:MINIMUM
2205	010164	010337	001420		MOV	R3,MAX		:MAXIMUM
2206	010170	012003		NXTCMP:	MOV	(R0)+,R3		
2207	010172	020337	001414		CMP	R3,MIN		
2208	010176	002002			BGE	MAXTST		
2209	010200	010337	001414		MOV	R3,MIN		:NEW MINIMUM
2210	010204	020337	001420	MAXTST:	CMP	R3,MAX		
2211	010210	003402			BLE	TST8		
2212	010212	010337	001420		MOV	R3,MAX		:NEW MAXIMUM
2213	010216	005302		TST8:	DEC	R2		
2214	010220	001363			BNE	NXTCMP		
2215	010222	013724	001414		MOV	MIN,(R4)+		
2216	010226	013725	001420		MOV	MAX,(R5)+		
2217	010232	022425			CMP	(R4)+,(R5)+		:BUMP EACH ONCE MORE
2218	010234	005301			DEC	R1		
2219	010236	001345			BNE	NXT8		
2220	010240	104401	013062		TYPE	,MSG18		
2221	010244	104401	014003		TYPE	,BUFF2		:TYPE BUFF2
2222	010250	012700	042300		MOV	#BUFFER,R0		
2223	010254	004737	010306		JSR	PC,LOAD		
2224	010260	104401	013703		TYPE	,C3		:TYPE ASCIZ STRING
2225	010264	012700	042302		MOV	#BUFFER+2,R0		
2226	010270	004737	010306		JSR	PC,LOAD		
2227	010274	104401	013700		TYPE	,C2		:TYPE ASCIZ STRING
2228	010300	004737	010330		JSR	PC,DELCLR		
2229	010304	000207		LO2:	RTS	PC		
2230	010306	012701	001000	LOAD:	MOV	#512.,R1		
2231	010312	012002		LOAD0:	MOV	(R0)+,R2		
2232	010314	005720			TST	(R0)+		
2233	010316	004737	011370		JSR	PC,LOADY		
2234	010322	005301			DEC	R1		
2235	010324	001372			BNE	LOAD0		
2236	010326	000207			RTS	PC		



```

2238 010330 005000          DELCLR: CLR      R0
2239 010332 012701 000020    MOV      #20,R1          ;DELAY BEFORE CLEANING SCREEN
2240 010336 005300          1$:    DEC      R0
2241 010340 001376          BNE      1$
2242 010342 005301          DEC      R1
2243 010344 001374          BNE      1$
2244 010346 032777 010000 170564 BIT      #BIT12,@SWR      ;TEST FOR HALT FOR DISPLAY
2245 010354 001401          BEQ      2$              ;:DON'T HALT FOR DISPLAY
2246 010356 000000          HALT
2247 010360 104401 014023    2$:    TYPE     ,VTINIT
2248 010364 000207          RTS      PC
2249          ;;NOISE SUBROUTINE;;
2250 010366 013537 001362    NOITST: MOV      @(R5)+,CHANL      ;LOAD CHANNEL
2251 010372 013737 001362 001360 MOV      CHANL,DUMMY        ;LOAD DUMMY CHANNEL
2252 010400 004737 006214          JSR      PC,GETEDG        ;GET EDGE VALUE
2253 010404 004737 010560          JSR      PC,NOIA         ;GET RMS AND PEAK VALUES
2254 010410 012737 000001 006436 MOV      #1,EDGFLG
2255 010416 004737 010424          JSR      PC,TYPRP        ;TYPE RMS AND PEAK VALUES
2256 010422 000205          RTS      R5
2257
2258
2259
2260
2261
2262          ;;TYPE RMS AND PEAK VALUES;;
2263 010424 104401 012373    TYPRP: TYPE     ,NOI
2264 010430 005737 001374          TST      RMS
2265 010434 100002          BPL      POSRMS
2266 010436 005037 001374          CLR      RMS              ;RMS<0,SET RMS=0
2267 010442 005737 001376    POSRMS: TST      PEAK
2268 010446 100002          BPL      POSPEA
2269 010450 005037 001376          CLR      PEAK            ;PEAK<0,SET PEAK=0
2270 010454 013702 001374    POSPEA: MOV      RMS,R2
2271 010460 004737 011472          JSR      PC,DECTYP
2272 010464 104401 012746          TYPE     ,MESR
2273 010470 013702 001376          MOV      PEAK,R2
2274 010474 004737 011472          JSR      PC,DECTYP
2275 010500 104401 012761          TYPE     ,MESP
2276 010504 004737 006374          JSR      PC,TYPEDG
2277 010510 104401 012403          TYPE     ,CHAN
2278 010514 013746 001362          MOV      CHANL,-(SP)      ;;SAVE CHANL FOR TYPEOUT
(1)          ;;TYPE CHANL
(1) 010520 104403          ;;GO TYPE--OCTAL ASCII
(1) 010522 002          ;;TYPE 2 DIGIT(S)
(1) 010523 000          ;;SUPPRESS LEADING ZEROS
2279 010524 023737 001374 011730 CMP      RMS,VNR          ;WITHIN LIMITS?
2280 010532 003007          BGT      ER
2281 010534 023737 001376 011732 CMP      PEAK,VNP
2282 010542 003003          BGT      ER
2283 010544 104401 012476          TYPE     ,OKMSG
2284 010550 000207          RTS      PC
2285 010552 104401 012507    ER:    TYPE     ,ERMSG
2286 010556 000207          RTS      PC

```

```

2288      ;:SUBROUTINES FOR NOISE TEST;:
2289 010560 005037 001374      NOIA: CLR RMS ;CLEAR RMS VLAUE
2290 010564 005037 001376      CLR PEAK ;CLEAR PEAK VALUE
2291 010570 004537 006440      NGI1: JSR R5,SAR SUB ;DO SAR ROUTINE AT 16%
2292 010574 000020      16.
2293 010576 063737 001404 001374      ADD DAC,RMS ;ADD RESULT TO RMS
2294 010604 004537 006440      JSR R5,SAR SUB ;DO SAR ROUTINE AT 84%
2295 010610 000124      84.
2296 010612 163737 001404 001374      SUB DAC,RMS ;SUBTRACT RESULT FROM RMS
2297 010620 004537 006440      JSR R5,SAR SUB ;DO SAR ROUTINE AT 1%
2298 010624 000001      1
2299 010626 063737 001404 001376      ADD DAC,PEAK ;ADD RESULT TO PEAK
2300 010634 004537 006440      JSR R5,SAR SUB ;DO SAR ROUTINE AT 99%
2301 010640 000143      99.
2302 010642 163737 001404 001376      SUB DAC,PEAK ;SUBTRACT RESULT FROM PEAK
2303 010650 000207      RTS PC ;RETURN
2304
2305 010652 012537 001362      NOI8: MOV (R5)+,CHANL ;GET CHANNEL VALUE
2306 010656 063737 001332 001362      ADD BASECH,CHANL
2307 010664 013737 001362 001360      MOV CHANL,DUMMY ;LOAD DUMMY CHANNEL
2308 010672 004737 006214      JSR PC,GETEDG ;GET EDGE VALUES
2309 010676 005037 001374      CLR RMS ;CLEAR RMS VALUE
2310 010702 005037 001376      CLR PEAK ;CLEAR PEAK VALUE
2311 010706 012737 000010 010774      MOV #10,10$ ;SET UP COUNTER
2312 010714 004737 010570      1$: JSR PC,NOI1 ;GET NOISE VALUES
2313 010720 005237 001410      INC EDGE
2314 010724 005337 010774      DEC 10$
2315 010730 001371      BNE 1$ ;REPEAT 8 TIMES
2316 010732 162737 000010 001410      SUB #10,EDGE ;SCALE IT TO 1 LSB=100.
2317 010740 006237 001374      ASR RMS
2318 010744 005537 001374      ADC RMS
2319 010750 006237 001376      ASR PEAK
2320 010754 005537 001376      ADC PEAK
2321 010760 012737 000010 006436      MOV #8.,EDGFLG
2322 010766 004737 010424      JSR PC,TYPRP ;TYPE RESULTS
2323 010772 000205      RTS R5 ;RETURN
2324 010774 000000      10$: 0 ;COUNTER
2325
2326
2327      ;:RANDOM NUMBER GENERATOR;:
2328 010776 063737 001370 001366      RANDY: ADD RNB,RNA
2329 011004 063737 001372 001366      ADD RNC,RNA
2330 011012 005537 001366      ADC RNA
2331 011016 063737 001366 001370      ADD RNA,RNB
2332 011024 063737 001372 001370      ADD RNC,RNB
2333 011032 005537 001370      ADC RNB
2334 011036 063737 001366 001372      ADD RNA,RNC
2335 011044 063737 001370 001372      ADD RNB,RNC
2336 011052 005537 001372      ADC RNC
2337 011056 000207      RTS PC

```



```

2339          ;;ROUTINE TO AVERAGE 8 CONVERSIONS;;
2340 011060 012500          CONVRT: MOV      (R5)+,RO          ;GET CHANNEL VALUE
2341 011062 063700 001332      ADD      BASECH,RO
2342 011066 010037 001362      MOV      RO,CHANL
2343 011072 000300          SWAB     RO
2344 011074 005037 001346      CLR      TEMP
2345
(1)          ;*      MOV      @ADBUFF,MYTEMP  ;/READ DEVICE REG ADBUFF,PUT DATA IN MYTEMP.
2346 011110 010037 001426      MOV      RO,MYTEMP
(2)
(2)          ;*      MOV      MYTEMP,@STREG  ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
2347 011124 012700 010000      MOV      #10000,RO
2348 011130 005300          2$:     DEC      RO
2349 011132 001376          BNE     2$
2350 011134 012777 001702 170312  MOV     #RETURN,@VECTOR  ;LOAD VECTOR
2351 011142 012700 000010      MOV     #10,RO          ;SET UP COUNTER
2352 011146          1$:
(1)
(1)          ;*      MOV      @STREG,MYTEMP  ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
2353 011156 052737 000001 001426  BIS     #1,MYTEMP
2354
(1)          ;*      MOV      MYTEMP,@STREG  ;/ PUT DATA FROM MYTEMP TO DEVICE REG STREG
2355 011174 005001          CLR     R1
2356 011176 105201          10$:   INCB   R1
2357 011200 001007          BNE    11$
2358 011202 012737 000200 001124  MOV     #BIT7,$GDDAT  ;EXPECT DONE TO SET BY NOW
2359 011210 013737 001426 001126  MOV     MYTEMP,$BDDAT
2360
2361 011216 104001          ERROR  1          ;DONE FAILED TO SET ON A/D
2362
2363 011220          11$:
2364
(2)          ;*      MOV      @STREG,MYTEMP  ;/READ DEVICE REG STREG,PUT DATA IN MYTEMP.
(1) 011230 105737 001426      TSTB  MYTEMP
2365 011234 100360          BPL   10$
2366
(1)          ;*      MOV      @ADBUFF,MYTEMP  ;/READ DEVICE REG ADBUFF,PUT DATA IN MYTEMP.
2367 011246 063737 001426 001346  ADD     MYTEMP,TEMP
2368          ;WAIT FOR CONVERSION
2369          ;READ BUFFER
2370          DEC     RO
2371 011256 001333          BNE    1$          ;DO 8 TIMES
2372 011260 006237 001346      ASR   TEMP        ;AVERAGE VALUE
2373 011264 006237 001346      ASR   TEMP
2374 011270 006237 001346      ASR   TEMP
2375 011274 005537 001346      ADC   TEMP
2376 011300 000205          RTS    R5          ;RETURN

```

```
2378 ;COMPARE $GDDAT AND $BDDAT;;
2379 011302 012537 001124 COMPAR: MOV (R5)+,$GDDAT ;GET GOOD DATA
2380 011306 013537 001402 MOV @ (R5)+,SPREAD ;GET SPREAD
2381 011312 013737 001346 001126 MOV TEMP,$BDDAT ;GET BAD(ACTUAL) DATA
2382 011320 013701 001126 MOV $BDDAT,R1
2383 011324 013700 001124 MOV $GDDAT,R0
2384 011330 160100 SUB R1,R0 ;GET DIFFERENCE
2385 011332 100001 BPL 7$
2386 011334 005400 NEG R0
2387 011336 020037 001402 7$: CMP R0,SPREAD ;COMPARE IT TO SPREAD
2388 011342 003001 BGT 10$ ;GO TO ERROR PRINTOUT
2389 011344 005725 TST (R5)+ ;BUMP RETURN POINTER AROUND ERROR CALL
2390 011346 000205 10$: RTS R5
2391 ;SUBROUTINE TO RESET & SET INTRPT. EN.;
2392 RST: JSR PC,$RESET
2393 011350 004737 020224 167562 BIS #100,@$TKS
2394 011354 052777 000100 CLR PSW
2395 011362 005037 177776 RTS PC
2396 011366 000207
2397
2398
2399 ;SUBROUTINE LOADY;
2400 LOADY: TST R2 ;ROUTINE TO LOAD VLAUE INTO R2
2401 011370 005702 BPL PLUSR2 ;AS A VT55 Y-VALUE
2402 011372 100001 CLR R2
2403 011374 005002 PLUSR2: CMP R2,#235.
2404 011376 020227 000353 BLT LESS
2405 011402 002402 MOV #235.,R2
2406 011404 012702 000353 LESS: MOV R2,R3
2407 011410 010203 BIC #177740,R2
2408 011412 042702 177740 BIS #40,R2
2409 011416 052702 000040 B10: TSTB @$TPS ;PRINT CHARACTER
2410 011422 105777 167522 BPL B10
2411 011426 100375 MOVB R2,@$TPB
2412 011430 110277 167516 ASR R3
2413 011434 006203 ASR R3
2414 011436 006203 ASR R3
2415 011440 006203 ASR R3
2416 011442 006203 ASR R3
2417 011444 006203 ASR R3
2418 011446 042703 177770 BIC #177770,R3
2419 011452 052703 000040 B11: BIS #40,R3 ;PRINT CHARACTER
2420 011456 105777 167466 B11: TSTB @$TPS
2421 011462 100375 RPL B11
2422 011464 110377 167462 MOVB R3,@$TPB
2423 011470 000207 RTS PC
2424
2425
```



```
2427      ::SUBROUTINE TO TYPE DECIMAL VALUE;;
2428      ::IN R2 AS X.XX;;
2429 011472 005702      DECTYP: TST      R2          ;TEST VALUE TO BE TYPED
2430 011474 100003      BPL      POS
2431 011476 104401 012235      TYPE      ,MINUS      ;TYPE MINUS SIGN
2432 011502 005402      NEG      R2
2433 011504 020227 001747      POS:    CMP      R2,#999.      ;>999. REPLACE IT WITH 999.
2434 011510 003402      BLE      OKAYD
2435 011512 012702 001747      MOV      #999.,R2
2436 011516 105037 014566      OKAYD: CLRB     ONES      ;CLEAR ONES
2437 011522 105037 014565      CLRB     TENS      ;CLEAR TENS
2438 011526 105037 014563      CLRB     HUNS      ;CLEAR HUNS
2439 011532 005702      TESTR2: TST      R2          ;CONVERT VALUE TO A DECIMAL VALUE
2440 011534 001424      BEQ      TYP0UT
2441 011536 005302      DEC      R2
2442 011540 105237 014566      INCB     ONES
2443 011544 123727 014566 000012      CMPB     ONES,#10.
2444 011552 001367      BNE      TESTR2
2445 011554 105037 014566      CLRB     ONES
2446 011560 105237 014565      INCB     TENS
2447 011564 123727 014565 000012      CMPB     TENS,#10.
2448 011572 001357      BNE      TESTR2
2449 011574 105037 014565      CLRB     TENS
2450 011600 105237 014563      INCB     HUNS
2451 011604 000752      BR       TESTR2
2452 011606 152737 000060 014563      TYP0UT: BISB     #60,HUNS      ;PREPARE FOR TYP0UT
2453 011614 152737 000060 014565      BISB     #60,TENS
2454 011622 152737 000060 014566      BISB     #60,ONES
2455 011630 104401 014563      TYPE     ,HUNS      ;TYPE VALUE
2456 011634 000207      RTS      PC
2457
2458 011636 012701 011730      WFADJ:  MOV     #VNR,R1      ;SUBROUTINE TO SET UP LIMITS
2459 011642 005737 001332      TST     BASECH      ;TESTING AN AM11K?
2460 011646 001403      BEQ     1$          ;;
2461 011650 012702 011762      MOV     #VARLT3,R2    ;BASECH NOT ZERO, USE AM11K LIMITS
2462 011654 000410      BR      3$          ;;
2463 011656 005737 001416      1$:    TST     WFTEST
2464 011662 001003      BNE     2$
2465 011664 012702 011742      MOV     #VARLT1,R2    ;WFTEST=0,USE NORMAL LIMITS
2466 011670 000402      BR      3$
2467 011672 012702 011752      2$:    MOV     #VARLT2,R2    ;WFTEST=1,USE OPTION AREA LIMITS
2468 011676 012221      3$:    MOV     (R2)+,(R1)+
2469 011700 005711      TST     (R1)
2470 011702 100375      BPL     3$
2471 011704 000207      RTS     PC
```







```
2511 .SBTTL ASCII MESSAGES
2512 012130 005015 047516 051511 NOIMSG: .ASCIZ <15><12>/NOISE TEST-- /
      012136 020105 042524 052123
      012144 026455 000040
2513 012150 005015 042523 052124 SETMSG: .ASCIZ <15><12>/SETTLING TEST-- TYPE DESIRED 'FROM' CHANNEL & CR: /
      012156 044514 043516 052040
      012164 051505 026524 020055
      012172 054524 042520 042040
      012200 051505 051111 042105
      012206 023440 051106 046517
      012214 020047 044103 047101
      012222 042516 020114 020046
      012230 051103 020072 000
2514 012235 055 000 MINUS: .BYTE 55,0
2515 012237 077 000 QUEST: .BYTE 77,0
2516 012241 136 101 040 AMSG: .BYTE 136,101,40,40,0
      012244 040 000
2517 012246 136 103 040 CMSG: .BYTE 136,103,40,40,0
      012251 040 000
2518 012253 136 107 015 GMSG: .BYTE 136,107,15,12,123,127,122,105,107,72,0
      012256 012 123 127
      012261 122 105 107
      012264 072 000
2519 012266 046040 041123 005015 LSBMSG: .ASCIZ / LSB/<15><12>
      012274 000
2520 012275 055 020055 000 DASH: .ASCIZ /-- /
2521 012301 123 040524 042524 STATE: .ASCIZ /STATE-- WIDTH/<15><12>
      012306 026455 053440 042111
      012314 044124 005015 000
2522 012321 103 000110 CH: .ASCIZ /CH/
2523 012324 020040 020040 000 SPACE: .ASCIZ / /
2524 012331 040 051514 020102 LSB: .ASCIZ / LSB ON CH/
      012336 047117 041440 000110
2525 012344 051440 052105 046124 SETCH: .ASCIZ / SETTLING FROM CH/
      012352 047111 020107 051106
      012360 046517 041440 000110
2526 012366 040440 020124 000 ATMSG: .ASCIZ / AT /
2527 012373 116 044517 042523 NOI: .ASCIZ /NOISE: /
      012400 020072 000
2528 012403 040 047117 041440 CHAN: .ASCIZ / ON CHANNEL /
      012410 040510 047116 046105
      012416 000040
2529 012420 020040 020040 047504 DONE: .ASCIZ / DONE/<15><12>
      012426 042516 005015 000
2530 012433 057 000 SLASH: .ASCIZ #/#
2531 012435 124 050131 020105 TOMSG: .ASCIZ /TYPE DESIRED 'TO' CHANNEL & CR: /
      012442 042504 044523 042522
      012450 020104 052047 023517
      012456 041440 040510 047116
      012464 046105 023040 041440
      012472 035122 000040
2532 012476 020040 020040 045517 OKMSG: .ASCIZ / OK/<15><12>
      012504 005015 000
```









2558 013610 040440 020124 000  
013615 015 041412 046101 HEAD5: .ASCII <15><12>/CALIBRATION--/  
013622 041111 040522 044524  
2559 013630 047117 026455  
013634 051440 052105 041440 ASKCH: .ASCIZ / SET CHANNEL IN SWR LOW BYTE/<15><12>  
013642 040510 047116 046105  
013650 044440 020116 053523  
013656 020122 047514 020127  
013664 054502 042524 005015  
013672 000  
2560 013673 033 000132 CO: .ASCIZ <33><132>  
2561 013676 000055 C1: .ASCIZ <55>  
2562 013700 031033 000 C2: .ASCIZ <33><62>  
2563 013703 112 000 C3: .ASCIZ <112>  
2564 013705 015 047412 043106 MOFSET: .ASCIZ <15><12>/OFFSET =/  
013712 042523 020124 000075  
2565 013720 046040 041123 000040 MLSB: .ASCIZ / LSB /  
2566 013726 040440 020124 000 MAT: .ASCIZ / AT /  
2567 013733 015 020012 047105 METST: .ASCIZ <15><12>/ ENTERING TEST /  
013740 042524 044522 043516  
013746 052040 051505 020124  
013754 000  
2568 013755 033 061 101 BUFF1: .BYTE 33,61,101,61,111,62,114,41,60,45,63,51,66,55,71,61,74,110,41,40,112,0  
013760 061 111 062  
013763 114 041 060  
013766 045 063 051  
013771 066 055 071  
013774 061 074 110  
013777 041 040 112  
014002 000  
2569 014003 033 061 101 BUFF2: .BYTE 33,61,101,47,111,61,104,50,65,44,62,110,40,40,102,0  
014006 047 111 061  
014011 104 050 065  
014014 044 062 110  
014017 040 040 102  
014022 000  
2570 014023 033 110 033 VTINIT: .BYTE 33,110,33,112,33,61,101,40,33,62,0  
014026 112 033 061  
014031 101 040 033  
014034 062 000  
2571 014036 005015 041412 046122 HEAD1: .ASCII <15><12><12>#CRLPKC0 LPA/AD11-K DIAGNOSTIC#<15><12>  
014044 045520 030103 020040  
014052 020040 050114 027501  
014060 042101 030461 045455  
014066 042040 040511 047107  
014074 051517 044524 006503  
014102 012  
2572 014103 012 035101 040440 .ASCII <12>/A: AUTO TEST/  
014110 052125 020117 042524  
014116 052123  
2573 014120 005015 035103 041440 .ASCII <15><12>/C: CALIBRATION/  
014126 046101 041111 040522  
014134 044524 047117  
2574 014140 005015 035114 046040 .ASCII <15><12>/L: LOGIC TEST/  
014146 043517 041511 052040  
014154 051505 124

2575	014157	015	047012	020072		.ASCII	<15><12>/N: NOISE TEST/
	014164	047516	051511	020105			
	014172	042524	052123				
2576	014176	005015	035123	051440		.ASCII	<15><12>/S: SETTLE TEST/
	014204	052105	046124	020105			
	014212	042524	052123				
2577	014216	005015	035127	053440		.ASCIZ	<15><12>/W: WRAPAROUND TEST/<15><12>
	014224	040522	040520	047522			
	014232	047125	020104	042524			
	014240	052123	005015	000			
2578	014245	015	051412	040524	EM1:	.ASCIZ	<15><12>/STATUS REG. ERROR/<15><12>
	014252	052524	020123	042522			
	014260	027107	042440	051122			
	014266	051117	005015	000			
2579	014273	015	043012	044501	EM2:	.ASCIZ	<15><12>/FAILED TO INTERRUPT/<15><12>
	014300	042514	020104	047524			
	014306	044440	052116	051105			
	014314	052522	052120	005015			
	014322	000					
2580	014323	015	052412	042516	EM3:	.ASCIZ	<15><12>/UNEXPECTED INTERRUPT/<15><12>
	014330	050130	041505	042524			
	014336	020104	047111	042524			
	014344	051122	050125	006524			
	014352	000012					
2581	014354	005015	051105	047522	FM4:	.ASCIZ	<15><12>#ERROR ON A/D CHANNEL#<15><12>
	014362	020122	047117	040440			
	014370	042057	041440	040510			
	014376	047116	046105	005015			
	014404	000					
2582	014405	105	051122	041520	DH1:	.ASCIZ	/ERRPC STREG EXPECTED ACTUAL/<15><12>
	014412	051440	051124	043505			
	014420	042440	050130	041505			
	014426	042524	020104	041501			
	014434	052524	046101	005015			
	014442	000					
2583	014443	105	051122	041520	DH2:	.ASCIZ	/ERRPC STREG CHANNEL NOMINAL TOLERANCE ACTUAL/
	014450	020040	052123	042522			
	014456	020107	020040	044103			
	014464	047101	042516	020114			
	014472	047040	046517	047111			
	014500	046101	020040	047524			
	014506	042514	040522	041516			
	014514	020105	040440	052103			
	014522	040525	000114				
2584	014526	051105	050122	020103	DH3:	.ASCIZ	/ERRPC STREG ACTUAL/<15><12>
	014534	020040	020040	051440			
	014542	051124	043505	020040			
	014550	020040	041501	052524			
	014556	046101	005015	000			



LPA-AD11K TEST MD-11-CRLPKC  
CRLPKC.P11 14-AUG-80 13:59

MACY11 30G(1063) 24-OCT-80 09:48 L 5 PAGE 39  
ASCII MESSAGES

SEQ 0063

2586	014563	000			HUNS:	.BYTE	0
2587	014564	056			DECPNT:	.BYTE	56
2588	014565	000			TENS:	.BYTE	0
2589	014566	000	000		ONES:	.BYTE	0,0
2590					.EVEN		
2591							
2592	014570	001116	001316	001124	DT1:	\$ERRPC, STREG, \$GDDAT, \$BDDAT,0	
	014576	001126	000000				
2593	014602	001116	001316	001362	DT2:	\$ERRPC,STREG,CHANL,\$GDDAT,SPREAD,\$BDDAT,0	
	014610	001124	001402	001126			
	014616	000000					
2594	014620	001116	001316	001126	DT3:	\$ERRPC,STREG,\$BDDAT,0	
	014626	000000					
2595							
2596	014630	000000			DF1:	0	
2597							
2598							

```

2600      .SBTTL  TTY INPUT ROUTINE
(1)
(2)      ::*****
(1)      .ENABL  LSB
(1)
(1)      .DSABL  LSB
(1)
(2)      ::*****
(1)      ::THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)      ::*CALL:
(1)      ::*      RDCHR          ::INPUT A SINGLE CHARACTER FROM THE TTY
(1)      ::*      RETURN HERE      ::CHARACTER IS ON THE STACK
(1)      ::*                               ::WITH PARITY BIT STRIPPED OFF
(1)      ::
(1)      $RDCHR: MOV      (SP),-(SP)      ::PUSH DOWN THE PC
(1)      014632 011646      MOV      4(SP),2(SP)      ::SAVE THE PS
(1)      014634 016666 000004 000002 1$:      TSTB    @STKS          ::WAIT FOR
(1)      014642 105777 164276      BPL     1$              ::A CHARACTER
(1)      014646 100375      MOVB   @STKB,4(SP)      ::READ THE TTY
(1)      014650 117766 164272 000004      BIC    #^C<177>,4(SP)  ::GET RID OF JUNK IF ANY
(1)      014656 042766 177600 000004      CMP    4(SP),#23       ::IS IT A CONTROL-S?
(1)      014664 026627 000004 000023      BNE    3$              ::BRANCH IF NO
(1)      014672 001013      TSTB  @STKS          ::WAIT FOR A CHARACTER
(1)      014674 105777 164244      BPL    2$              ::LOOP UNTIL ITS THERE
(1)      014700 100375      MOVB  @STKB,-(SP)      ::GET CHARACTER
(1)      014702 117746 164240      BIC    #^C177,(SP)     ::MAKE IT 7-BIT ASCII
(1)      014706 042716 177600      CMP    (SP)+,#21       ::IS IT A CONTROL-Q?
(1)      014712 022627 000021      BNE    2$              ::IF NOT DISCARD IT
(1)      014716 001366      BR     1$              ::YES, RESUME
(1)      014720 000750      CMP   4(SP),#140      ::IS IT UPPER CASE?
(1)      014722 026627 000004 000140 3$:      BLT    4$              ::BRANCH IF YES
(1)      014730 002407      CMP   4(SP),#175      ::IS IT A SPECIAL CHAR?
(1)      014732 026627 000004 000175      BGT    4$              ::BRANCH IF YES
(1)      014740 003003      BIC   #40,4(SP)       ::MAKE IT UPPER CASE
(1)      014742 042766 000040 000004 4$:      RTI     ::GO BACK TO USER
(1)      014750 000002
(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)      $RDLIN: MOV      R3,-(SP)      ::SAVE R3
(1)      014752 010346      MOV    #TTYIN,R3      ::GET ADDRESS
(1)      014754 012703 015060 1$:      CMP    #TTYIN+8.,R3   ::BUFFER FULL?
(1)      014760 022703 015070 2$:      BLOS   4$              ::BR IF YES
(1)      014764 101405      RDCHR          ::GO READ ONE CHARACTER FROM THE TTY
(1)      014766 104406      MOVB  (SP)+,(R3)      ::GET CHARACTER
(1)      014770 112613      CMPB  #177,(R3)      ::IS IT A RUBOUT
(1)      014772 122713 000177 10$:     BNE    3$              ::SKIP IF NOT
(1)      014776 001003      TYPE  $QUES          ::TYPE A '?'
(1)      015000 104401 001170 4$:      BR     1$              ::CLEAR THE BUFFER AND LOOP
(1)      015004 000763      MOVB  (R3),9$         ::ECHO THE CHARACTER
(1)      015006 111337 015056 3$:      TYPE  .9$
(1)      015012 104401 015056

```



(1)	015016	122723	000015			CMPB	#15,(R3)+	::CHECK FOR RETURN
(1)	015022	001356				BNE	2\$	::LOOP IF NOT RETURN
(1)	015024	105063	177777			CLRB	-1(R3)	::CLEAR RETURN (THE 15)
(1)	015030	104401	001172			TYPE	,SLF	::TYPE A LINE FEED
(1)	015034	012603				MOV	(SP)+,R3	::RESTORE R3
(1)	015036	011646				MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
(1)	015040	016666	000004	000002		MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
(1)	015046	012766	015060	000004		MOV	#\$TTYIN,4(SP)	
(1)	015054	000002				RTI		::RETURN
(1)	015056	000			9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
(1)	015057	000				.BYTE	0	::TERMINATOR
(1)	015060	000010			\$TTYIN:	.BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT
(1)	015070	052536	005015	000	\$CNTLU:	.ASCIZ	/^U/<15><12>	::CONTROL "U"
(1)	015075	136	006507	000012	\$CNTLG:	.ASCIZ	/^G/<15><12>	::CONTROL "G"
(1)	015102	005015	053523	020122	\$MSWR:	.ASCIZ	<15><12>/SWR = /	
(1)	015110	020075	000					
(1)	015113	040	047040	053505	\$MNEW:	.ASCIZ	/ NEW = /	
(1)	015120	036440	000040					











```
(1) 015674 000000
(1) 015676
(1) 015676 000002
2606
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 015700
(1) 015700 104401 001171
(1) 015704 010046
(1) 015706 005000
(1) 015710 153700 001114
(1) 015714 001004
(1)
(2) 015716 013746 001116
(2)
(2) 015722 104402
(1) 015724 000426
(1) 015726 005300
(1) 015730 006300
(1) 015732 006300
(1) 015734 006300
(1) 015736 062700 001256
(1) 015742 012037 015752
(1) 015746 001404
(1) 015750 104401
(1) 015752 000000
(1) 015754 104401 001171
(1) 015760 012037 015770
(1) 015764 001404
(1) 015766 104401
(1) 015770 000000
(1) 015772 104401 001171
(1) 015776 011000
(1) 016000 001004
(1) 016002 012600
(1) 016004 104401 001171
(1) 016010 000207
(1) 016012
(2) 016012 013046
(2) 016014 104402
(1) 016016 005710
(1) 016020 001770
(1) 016022 104401 016030
(1) 016026 000771
(1) 016030 020040 000
(1) 016034
```

```
HALT ;;YES
6$: RTI ;;RETURN
.SBttl ERROR MESSAGE TIMEOUT ROUTINE

;*****
;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

$ERRTYP:
        TYPE      , $CRLF      ;; 'CARRIAGE RETURN' & 'LINE FEED'
        MOV       RO, -(SP)    ;; SAVE RO
        CLR       RO           ;; PICKUP THE ITEM INDEX
        BISB      @#$ITEMB, RO
        BNE      1$           ;; IF ITEM NUMBER IS ZERO, JUST
                               ;; TYPE THE PC OF THE ERROR
        MOV       $ERRPC, -(SP) ;; SAVE $ERRPC FOR TIMEOUT
                               ;; ERROR ADDRESS
        TYPoc     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        BR       6$           ;; GET OUT
1$:     DEC       RO           ;; ADJUST THE INDEX SO THAT IT WILL
        ASL      RO           ;; WORK FOR THE ERROR TABLE
        ASL      RO
        ASL      RO
        ADD      #$ERRTB, RO   ;; FORM TABLE POINTER
        MOV      (RO)+, 2$    ;; PICKUP "ERROR MESSAGE" POINTER
        BEQ      3$           ;; SKIP TIMEOUT IF NO POINTER
        TYPE     ;; TYPE THE "ERROR MESSAGE"
        .WORD    0           ;; "ERROR MESSAGE" POINTER GOES HERE
2$:     TYPE     , $CRLF      ;; 'CARRIAGE RETURN' & 'LINE FEED'
        MOV      (RO)+, 4$    ;; PICKUP "DATA HEADER" POINTER
        BEQ      5$           ;; SKIP TIMEOUT IF 0
        TYPE     ;; TYPE THE "DATA HEADER"
        .WORD    0           ;; "DATA HEADER" POINTER GOES HERE
3$:     MOV      (RO), RO     ;; "CARRIAGE RETURN" & "LINE FEED"
        BEQ      5$           ;; PICKUP "DATA TABLE" POINTER
        TYPE     ;; TYPE THE "DATA"
        .WORD    0           ;; GO TYPE THE DATA
4$:     MOV      (RO), RO     ;; RESTORE RO
        BNE      7$           ;; "CARRIAGE RETURN" & "LINE FEED"
5$:     MOV      (SP)+, RO    ;; RETURN
        TYPE     , $CRLF      ;; 'CARRIAGE RETURN' & 'LINE FEED'
        RTS      PC
7$:     MOV      @ (RO)+, -(SP) ;; SAVE @ (RO)+ FOR TIMEOUT
        TYPoc     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        TST      (RO)        ;; IS THERE ANOTHER NUMBER?
        BEQ      6$           ;; BR IF NO
        TYPE     , 8$        ;; TYPE TWO(2) SPACES
        BR       7$           ;; LOOP
8$:     .ASCIZ   / /         ;; TWO(2) SPACES
        .EVEN
```





```
(1) 016220 000770 BR 7$ ;;LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1) 016222 112716 000040 8$: MOVB #' (SP) ;;REPLACE TAB WITH SPACE
(1) 016226 004737 016246 9$: JSR PC,$TYPEC ;;TYPE A SPACE
(1) 016232 132737 000007 016312 BITB #7,$SCHARCNT ;;BRANCH IF NOT AT
(1) 016240 001372 BNE 9$ ;;TAB STOP
(1) 016242 005726 TST (SP)+ ;;POP SPACE OFF STACK
(1) 016244 000724 BR 2$ ;;GET NEXT CHARACTER
(1) 016246 105777 162676 $TYPEC: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
(1) 016252 100375 BPL $TYPEC
(1) 016254 116677 000002 162670 MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 016262 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
(1) 016270 001003 BNE 1$ ;;BRANCH IF NO
(1) 016272 105037 016312 CLRB $SCHARCNT ;;YES--CLEAR CHARACTER COUNT
(1) 016276 000406 BR $TYPEX ;;EXIT
(1) 016300 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
(1) 016306 001402 BEQ $TYPEX ;;BRANCH IF YES
(1) 016310 105227 INCB (PC)+ ;;COUNT THE CHARACTER
(1) 016312 000000 $SCHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
(1) 016314 000207 $TYPEX: RTS PC
```

2609 .SBTTL APT COMMUNICATIONS ROUTINE

```
(1)
(2) ;*****
(1) 016316 112737 000001 016562 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
(1) 016324 112737 000001 016560 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
(1) 016332 000403 BR $ATYC
(1) 016334 112737 000001 016562 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) 016342 $ATYC:
(3) 016342 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(5) 016344 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(1) 016346 105737 016560 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
(1) 016352 001450 BEQ 5$ ;;IF NOT: BR
(1) 016354 122737 000001 001214 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 016362 001031 BNE 3$ ;;IF NOT: BR
(1) 016364 132737 000100 001215 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 016372 001425 BEQ 3$ ;;IF NOT: BR
(1) 016374 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
(1) 016400 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 016406 005737 001174 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) 016412 001375 BNE 1$ ;;IF NOT: WAIT
(1) 016414 010037 001210 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
(1) 016420 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
(1) 016422 001376 BNE 2$
(1) 016424 163700 001210 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 016430 006200 ASR R0 ;;GET MESSAGE LNTH IN WORDS
(1) 016432 010037 001212 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
(1) 016436 012737 000004 001174 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 016444 000413 BR 5$
(1) 016446 017637 000004 016472 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) 016454 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 016462 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 016466 004737 016034 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 016472 000000 4$: .WORD 0
```





```

(2)                                ;CERTAIN TESTING.
(2) 016572 012737 016616 000004  MOV    #30$,4
(2) 016600 005237 170000          INC    170000
(3) 016604 104401 016612          TYPE  ,65$      ;;TYPE ASCIZ STRING
(3) 016610 000401                  BR     64$      ;;GET OVER THE ASCIZ
(3)                                ;;65$: .ASCIZ <7>##
(3) 016614                        64$:
(2) 016614 000401                  BR     31$
(2) 016616 022626          30$:    CMP    (SP)+,(SP)+
(2) 016620 012637 000004          31$:    MOV    (SP)+,4      ;ALL THIS JUNK MUST BE REMOVED!!
(2) 016624 005037 017252          CLR    $AERR
(2) 016630 004537 017254          JSR    R5,$LOAD    ;LOAD MICRO-CODE.
(2) 016634 000000G             .WORD  DRLPX2      ;FILE 'DRLPX2.OBJ'
(2) 016636 052777 040000 162570  BIS    #BIT14,@KMADO ;ISSUE KMC+DMC INIT.
(2)                                1$:
(2) 016644                                ;'HANGS' HERE THEN KMC-11 ERROR.
(2) 016644 010146          MOV    R1,-(SP)
(2) 016646 005001          CLR    R1
(2) 016650 005201          2$:    INC    R1      ;STALL FOR DMC-UP
(2) 016652 001376          BNE   2$
(2) 016654 012777 104000 162552  MOV    #BIT15!BIT11,@KMADO ;SET RUN, AND ENABLE ARBITRATION.
(2) 016662 105201          25$:    INCB  R1
(2) 016664 001376          BNE   25$
(2) 016666 032777 000040 162540  BIT    #BIT5,@KMADO ;SLAVE READY? (READING IPBM SR)
(2) 016674 001401          BEQ   3$
(2)                                ;FATAL LPA-11 ERROR SLAVE NOT READY.
(2) 016676 104000          ERROR
(2) 016700 012777 000004 162532  3$:    MOV    #4,@KMAD2 ;READ FAST PATH
(2) 016706                        4$:
(3) 016706 004537 020164          JSR    R5,$TOUT   ;-TOUT-CHECK FOR TIMEOUT
(3) 016712 104000          ERROR ;/TIME-OUT ERROR
(3)                                ;/WE FAILED TO COMPLETE
(3)                                ;/CURRENT OPERATION.
(3)                                ;/CONTINUES IN THIS LOOP
(3)                                ;/WOULD MAKE US 'HANG' HERE
(3) 016714 000774          BR     4$
(3)                                ;/RETURNS HERE-FROM-TIMED OUT.
(2) 016716 122777 000377 162514  CMPB   #377,@KMAD2 ;WAIT TILL KMC DONE COMMAND.
(2) 016724 001370          BNE   4$
(2) 016726 122777 000377 162510  CMPB   #377,@KMAD4 ;IF FAST PATH=377 THEN ERROR.
(2) 016734 001001          BNE   35$
(2) 016736 104000          ERROR ;IPBM ERROR (SLAVE SIDE)
(2)                                ;YOU MUST RUN IPBM DIAGNOSTIC.
(2) 016740 117737 162500 017220  35$:    MOVB   @KMAD4,11$ ;GET THE VERSION NUMBER FROM DMC-11
(2) 016746 005227 177777          INC    #-1
(2) 016752 001045          BNE   5$
(2) 016754 005227 177777          INC    #-1
(2) 016760 001042          BNE   5$

```

```

(3) 016762 104401 016770          TYPE      ,67$          ;;TYPE ASCIZ STRING
(3) 016766 000426          BR          ,66$          ;;GET OVER THE ASCIZ
(3)          ;;67$: .ASCIZ <200>'M8200-YC (DMC) MICROCODE VERSION NUMBER = ''
(3) 017044          66$:          MOV          11$,-(SP)
(2) 017044 013746 017220          TYPOS
(2) 017050 104403          .BYTE      2,0
(2) 017052 002 000          TYPE      ,69$          ;;TYPE ASCIZ STRING
(3) 017054 104401 017062          BR          ,68$          ;;GET OVER THE ASCIZ
(3) 017060 000402          ;;69$: .ASCIZ <200>' '
(3) 017066          68$:
(2) 017066 112737 177777 017220 5$:  MOVB      #0-1,11$      ;DAC CODE FOR SLAVE.
(2) 017074 012501          MOV          (5)+,R1      ;GET NEXT DEVICE ADDR.
(2) 017076 021127 000000 6$:  CMP          (R1),#0      ;TERM REACHED?
(2) 017102 001444          BEQ          10$
(2) 017104 105237 017220          INCB          11$
(2) 017110 113777 017220 162326  MOVB      11$,@KMAD4      ;FIFO DATA
(2) 017116 004737 017222          JSR          PC,20$      ;ISSUE SEND
(2) 017122 112177 162316  MOVB      (R1)+,@KMAD4      ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
(2) 017126 004737 017222          JSR          PC,20$      ;ISSUE SEND
(2) 017132 112177 162306  MOVB      (R1)+,@KMAD4      ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
(2) 017136 004737 017222          JSR          PC,20$
(2) 017142 032777 000002 162264 7$:  BIT          #BIT1,@KMAD0      ;WAIT FOR FIFO DATA
(2) 017150 001374          BNE          7$          ;=1 NO DATA. =0 DATA.
(2) 017152 112777 000002 162260  MOVB      #2,@KMAD2      ;READ FIFO.
(2) 017160          8$:
(3) 017160 004537 020164          JSR          R5, $TOUT      ;--TOUT-CHECK FOR TIMEOUT
(3) 017164 104000          ERROR          ;/TIME-OUT ERROR
(3)          ;/WE FAILED TO COMPLETE
(3)          ;/CURRENT OPERATION.
(3)          ;/CONTINUES IN THIS LOOP
(3)          ;/WOULD MAKE US 'HANG' HERE
(3) 017166 000774          BR          8$
(3) 017170 122777 000377 162242  CMPB      #377,@KMAD2      ;/RETURNS HERE-FROM-TIMED OUT.
(2) 017176 001370          BNE          8$          ;WAIT FOR READ.
(2) 017200 105777 162240          TSTB      @KMAD4          ;WAS A ZERO RETURNED?
(2) 017204 001734          BEQ          6$          ;YES GET NEXT ADDR.
(2) 017206 005237 017252          INC          $AERR          ;SLAVE WILL RETURN CODE 0 IF
(2) 017212 005041          CLR          -(1)          ;DEV PRESENT. ELSE
(2) 017214 012601          MOV          (SP)+,R1      ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
(2) 017216 000205          RTS          R5          ;GET RID OF REFERENCE TO BAD ADDR.
(2) 017220 000000          10$:          .WORD      0          ;RETURN ALL ADDR. CHECKED.
(2) 017222 112777 000003 162210 20$:  MOVB      #3,@KMAD2      ;HOLDS DAC CODE PLUS OFFSET
(2) 017230          21$:          ;TO SLAVES ADDR. TABLE.

```



```

(3) 017230 004537 020164      JSR    R5, $TOUT      ;:-TOUT-CHECK FOR TIMEOUT
(3)
(3) 017234 104000              ERROR                ;/TIME-OUT ERROR
(3)                          ;/WE FAILED TO COMPLETE
(3)                          ;/CURRENT OPERATION.
(3)                          ;/CONTINUES IN THIS LOOP
(3)                          ;/WOULD MAKE US "HANG" HERE
(3)
(3) 017236 000774              BR                21$
(3)
(2) 017240 122777 000377 162172  CMPB   #377,@KMAD2    ;/RETURNS HERE-FROM-TIMED OUT.
(2) 017246 001370              BNE    21$           ;KMC CODE WILL RETURN A "377"
(2) 017250 000207              RTS    PC            ;WHEN DONE COMMAND.
(2)
(2) 017252 000000              $AERR: .WORD 0      ;=0 IF ADDR. LIST OK,=1 IF BAD.
(2)
(2)                          ;*
(2)                          ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
(2)                          ;*   CALL = JSR    R5,$LOAD
(2)                          ;*           .WORD  XX           ;ADDR. OF MICRO CODE.
(2)                          ;*           ;RETURNS HERE
(2)                          ;*   NOTE:  MICRO CODE FILE MUST END IN -1 DATA.
(2)                          ;*
(2)
(2) 017254 010446              $LOAD: MOV    R4,-(SP)   ;SAVE R4.
(2) 017256 010046              MOV    R0,-(SP)     ;SAVE R0.
(2) 017260 012500              1$:   MOV    (5)+,R0  ;GET PROG. ADDR.
(2) 017262 005077 162146      CLR    @KMAD0        ;CLEAR CSR
(2) 017266 005077 162152      CLR    @KMAD4        ;CLEAR CRAM ADDR.
(2) 017272 052777 002000 162134 2$:   BIS    #2000,@KMAD0  ;SELECT CRAM.
(2) 017300 012077 162144      MOV    (0)+,@KMAD6   ;WRITE DATA.
(2) 017304 052777 020000 162122  BIS    #20000,@KMAD0 ;SET CRAM WRITE
(2) 017312 005077 162116      CLR    @KMAD0        ;DISABLE CRAM.
(2) 017316 005277 162122      INC    @KMAD4        ;UPDATE CRAM ADDR.
(2) 017322 021027 177777      CMP    (0),#-1       ;ALL DONE?
(2) 017326 001361              BNE    2$            ;NO LOOP.
(2) 017330 005077 162110      CLR    @KMAD4        ;CLEAR CRAM ADDR.
(2) 017334 016500 177776      MOV    -2(5),R0     ;GET MICRO CODE ADDR.
(2)
(2) 017340 052777 002000 162066 3$:   BIS    #2000,@KMAD0  ;SELECT CRAM
(2) 017346 022077 162076      CMP    (R0)+,@KMAD6  ;DATA OK?
(2) 017352 001013              BNE    5$            ;NO - REPORT AN ERROR.
(2) 017354 021027 177777      CMP    (0),#-1       ;ALL DONE?
(2) 017360 001405              BEQ    4$            ;YES - EXIT
(2) 017362 005077 162046      CLR    @KMAD0        ;NO - DESELECT CRAM.
(2) 017366 005277 162052      INC    @KMAD4        ;UPDATE CRAM ADDR.
(2) 017372 000762              BR     3$
(2)
(2) 017374 012600              4$:   MOV    (SP)+,R0     ;RESTORE R0
(2) 017376 012604              MOV    (SP)+,R4     ;RESTORE R4
(2) 017400 000205              RTS    R5            ;EXIT
(2)
(2) 017402              5$:   ;COME HERE ON LOAD ERROR
(2) 017402 005745              TST    -(5)
(2) 017404 105204              INCB   R4            ;UPDATE ERROR COUNTER.

```

```
(2) 017406 100324      BPL      1$      ;IF NOT TOO MANY, TRY AGAIN.
(2) 017410 000000      HALT                    ;MICRO CODE LOAD ERROR.
(2)                                ;KMC-11 FAULT. YOU COULD TRY
(2) 017412 000722      BR        1$      ;TO PRESS CONTINUE TO GIVE IT
(2)                                ;ANOTHER CHANCE, BUT I DOUBT
(2)                                ;THAT THAT WOULD WORK. SINCE I'VE
(2)                                ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
(2)                                ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
```

```
(2)                                ;*THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
(2)                                ;*
(2)                                ;*      CALL = JSR      R5,$TLKW
(2)                                ;*      .WORD      0      ;OFFSET OF DEVICE ADDR.
(2)                                ;*      .WORD      0      ;DATA TO BE WRITTEN
(2)                                ;*
(2) 017414 010046      $TLKW: MOV      R0,-(SP)      ;SAVE R0
(2) 017416 012500      MOV      (5)+,R0      ;GET DEVICE OFFSET
(2) 017420 052700 000340  BIS      #340,R0      ;ADD WRITE CODE.
(2) 017424 004737 017676  JSR      PC,$LPW      ;WAIT FOR FAST PATH READY
(2) 017430 010037 017522  MOV      R0,W1
(2) 017434 010077 162004  MOV      R0,@KMAD4
(2) 017440 112777 000005 161772  MOVB     #5,@KMAD2      ;ISSUE FAST PATH WRITE
(2) 017446 004737 017676  JSR      PC,$LPW      ;WAIT FOR RDY
(2) 017452 011537 017524  MOV      (5),W2
(2) 017456 112577 161762  MOVB     (5)+,@KMAD4    ;WRITE LOW BYTE DATA.
(2)                                ;*
(2) 017462 112777 000005 161750  MOVB     #5,@KMAD2      ;FP WRITE
(2) 017470 004737 017676  JSR      PC,$LPW
(2) 017474 111537 017526  MOVB     (5),W3
(2) 017500 112577 161740  MOVB     (5)+,@KMAD4    ;WRITE HIGH BYTE
(2) 017504 112777 000005 161726  MOVB     #5,@KMAD2
(2) 017512 004737 017676  JSR      PC,$LPW
(2) 017516 012600  MOV      (SP)+,R0
(2) 017520 000205  RTS      R5      ;EXIT DONE.
(2) 017522 000000      W1:      0
(2) 017524 000000      W2:      0
(2) 017526 000000      W3:      0
```

```
(2)                                ;*
(2)                                ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
(2)                                ;*
(2)                                ;*      CALL = JSR      R5,$TLKR
(2)                                ;*      .WORD      0      ;OFFSET OF DEVICE
(2)                                ;*      ;RETURNS HERE
(2)                                ;*DATA IN WORD $DATR
(2)                                ;*
(2) 017530 010046      $TLKR: MOV      R0,-(SP)      ;SAVE R0
(2) 017532 012500      MOV      (5)+,R0      ;GET OFFSET
(2) 017534 052700 000300  BIS      #300,R0      ;ADD READ CODE
(2) 017540 004737 017676  JSR      PC,$LPW      ;WAIT TILL READY
(2) 017544 110077 161674  MOVB     R0,@KMAD4
(2) 017550 112777 000005 161662  MOVB     #5,@KMAD2      ;ISSUE WRITE FP
(2) 017556 004737 017676  JSR      PC,$LPW
```



```

(2) 017562 010037 017672
(2) 017566
(3) 017566 004537 020164
(3)
(3) 017572 104000
(3)
(3)
(3)
(3)
(3) 017574 000774
(3)
(3)
(2) 017576 032777 000040 161630
(2) 017604 001370
(2) 017606 112777 000004 161624
(2) 017614 004737 017676
(2) 017620 117737 161620 017674
(2) 017626
(3) 017626 004537 020164
(3)
(3) 017632 104000
(3)
(3)
(3)
(3) 017634 000774
(3)
(3)
(2) 017636 032777 000040 161570
(2) 017644 001370
(2) 017646 112777 000004 161564
(2) 017654 004737 017676
(2) 017660 117737 161560 017675
(2) 017666 012600
(2) 017670 000205
(2) 017672 000000
(2) 017674 000000
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2) 017676 010146
(2) 017700 005001
(2) 017702 122777 000377 161530
(2) 017710 001403
(2) 017712 005201
(2) 017714 001372
(2) 017716 000411

1$: MOV R0,RD1
JSR R5,$TOUT ; -TOUT-CHECK FOR TIMEOUT
ERROR ; /TIME-OUT ERROR
; /WE FAILED TO COMPLETE
; /CURRENT OPERATION.
; /CONTINUES IN THIS LOOP
; /WOULD MAKE US 'HANG' HERE

BR 1$

; /RETURNS HERE-FROM-TIMED OUT.
; FAST PATH GOT DATA?
BIT #BITS,@KMADO
BNE 1$
MOVB #4,@KMAD2 ; ISSUE FAST PATH READ
JSR PC,$LPW
MOVB @KMAD4,$DATR ; GET LOW BYTE

2$: JSR R5,$TOUT ; -TOUT-CHECK FOR TIMEOUT
ERROR ; /TIME-OUT ERROR
; /WE FAILED TO COMPLETE
; /CURRENT OPERATION.
; /CONTINUES IN THIS LOOP
; /WOULD MAKE US 'HANG' HERE

BR 2$

; /RETURNS HERE-FROM-TIMED OUT.
; FAST PATH READY?
BIT #BITS,@KMADO
BNE 2$
MOVB #4,@KMAD2 ; ISSUE FAST PATH READ
JSR PC,$LPW
MOVB @KMAD4,$DATR+1 ; SAVE HIGH BYTE
MOV (SP)+,R0
RTS R5

RD1: 0
$DATR: .WORD 0

; THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
; AS FAST PATH TO BE READ.
;
; CALL = JSR PC,$LPW
;
; IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
; THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
;

$LPW: MOV R1,-(SP) ; SAVE R1
CLR R1
CMPB #377,@KMAD2 ; FINISHED INSTRUCTION?
BEQ 2$
INC R1 ; TIME OUT?
BNE 1$
BR 10$

```























(1)	021172	105716			TSTB	(SP)	:: STILL DOING LEADING 0'S?
(1)	021174	100407			BMI	7\$	:: BR IF YES
(1)	021176	106316		5\$:	ASLB	(SP)	:: MSD?
(1)	021200	103003			BCC	6\$	:: BR IF NO
(1)	021202	116663	000001	177777	MOVB	1(SP),-1(R3)	:: YES--SET THE SIGN
(1)	021210	052702	000060		BIS	#'0,R2	:: MAKE THE BCD DIGIT ASCII
(1)	021214	052702	000040		BIS	#',R2	:: MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)	021220	110223			MOVB	R2,(R3)+	:: PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)	021222	005720			TST	(R0)+	:: JUST INCREMENTING
(1)	021224	020027	000010		CMP	R0,#10	:: CHECK THE TABLE INDEX
(1)	021230	002746			BLT	2\$	:: GO DO THE NEXT DIGIT
(1)	021232	003002			BGT	8\$	:: GO TO EXIT
(1)	021234	010502			MOV	R5,R2	:: GET THE LSD
(1)	021236	000764			BR	6\$	:: GO CHANGE TO ASCII
(1)	021240	105726		8\$:	TSTB	(SP)+	:: WAS THE LSD THE FIRST NON-ZERO?
(1)	021242	100003			BPL	9\$	:: BR IF NO
(1)	021244	116663	177777	177776	MOVB	-1(SP),-2(R3)	:: YES--SET THE SIGN FOR TYPING
(1)	021252	105013		9\$:	CLRB	(R3)	:: SET THE TERMINATOR
(3)	021254	012605			MOV	(SP)+,R5	:: POP STACK INTO R5
(3)	021256	012603			MOV	(SP)+,R3	:: POP STACK INTO R3
(3)	021260	012602			MOV	(SP)+,R2	:: POP STACK INTO R2
(3)	021262	012601			MOV	(SP)+,R1	:: POP STACK INTO R1
(3)	021264	012600			MOV	(SP)+,R0	:: POP STACK INTO R0
(1)	021266	104401	021314		TYPE	,SDBLK	:: NOW TYPE THE NUMBER
(1)	021272	016666	000002	000004	MOV	2(SP),4(SP)	:: ADJUST THE STACK
(1)	021300	012616			MOV	(SP)+,(SP)	
(1)	021302	000002			RTI		:: RETURN TO USER
(1)	021304	023420			\$DTBL:	10000.	
(1)	021306	001750				1000.	
(1)	021310	000144				100.	
(1)	021312	000012				10.	
(1)	021314	000004			\$DBLK:	.BLKW 4	





```
(1) 021450 000000          HALT
(1) 021452 000776          BR      .-2          ;;HANG UP
(1)
(2)
(1)
*****
(1) 021454 012737 021546 000024 $PWRUP: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1) 021462 013706 021552          MOV    $$SAVR6,SP      ;;GET SP
(1) 021466 005037 021552          CLR    $$SAVR6        ;;WAIT LOOP FOR THE TTY
(1) 021472 005237 021552          1$:  INC    $$SAVR6    ;;WAIT FOR THE INC
(1) 021476 001375          BNE    1$             ;;OF WORD
(3) 021500 012677 157434          MOV    (SP)+,@SWR    ;;POP STACK INTO @SWR
(3) 021504 012605          MOV    (SP)+,R5     ;;POP STACK INTO R5
(3) 021506 012604          MOV    (SP)+,R4     ;;POP STACK INTO R4
(3) 021510 012603          MOV    (SP)+,R3     ;;POP STACK INTO R3
(3) 021512 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
(3) 021514 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
(3) 021516 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
(1) 021520 012737 021402 000024  MOV    #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 021526 012737 000340 000026  MOV    #340,@#PWRVEC+? ;;PRIO:7
(1) 021534 104401          TYPE                                ;;REPORT THE POWER FAILURE
(1) 021536 021554          $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
(1) 021540 012716          MOV    (PC)+,(SP)   ;;RESTART AT BEG2
(1) 021542 002402          $PWRAD: .WORD BEG2  ;;RESTART ADDRESS
(1) 021544 000002          RTI
(1) 021546 000000          $ILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
(1) 021550 000776          BR      .-2          ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 021552 000000          $$SAVR6: 0          ;;PUT THE SP HERE
2616 021554 005015 042522 052123  $PWRMSG: .ASCIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
      021562 051101 044524 043516
      021570 040440 052106 051105
      021576 040440 050040 053517
      021604 051105 043040 044501
      021612 052514 042522 005015
      021620 000
2617 021622
2618 021622 000310          .EVEN
2619
2620          .=42000
2621          ;THE MICRO-CODE FOR THIS PROGRAM RESIDES HERE.
2622          .=42300
2623
2624 042300 010000          BUFFER: .BLKW 4096.      ;BUFFER AREA
2625
2626          .END
```











ERRVEC=	000004	1163#	1348*	2604*						
FIRST	001342	1274#	2008*	2081	2083*					
FIXADR	005360	1782	1790#							
FIXONE	005364	1372	1791#							
FLAG	001400	1289#	1351*	1368*	2130	2184				
GETDAT	010106	2189#	2197							
GETEDG	006214	1840	1890	1911#	2252	2308				
GMSG	012253	1333	2518#							
GNS =	***** U	1215	2610	2614						
HAFMSG	012677	2124	2539#							
HALF	007566	2116	2118#							
HEAD1	014036	1371	2571#							
HEAD5	013615	1713	2558#							
HT =	000011	1163#	2608							
HUNS	014563	2438*	2450*	2452*	2455	2586#				
INRNGE	007312	2065	2068#							
IOTVEC=	000020	1163#	1348*							
ISERV	001546	1314#	1374							
KBVECT	001334	1271#	1373	1805						
KMADO	001434	1305#	1350	2610*						
KMAD1	001436	1305#	1350							
KMAD2	001440	1305#	1973*	1974	1977*	1978	2032*	2033	2610*	
KMAD3	001442	1305#	2610*							
KMAD4	001444	1305#	1972*	1976*	1980	2031*	2035	2610*		
KMAD5	001446	1305#								
KMAD6	001450	1305#	2610*							
KMAD7	001452	1305#	1350	2610*						
LAST	007430	2079	2092#							
LEND	004652	1701	1703#							
LESS	011410	2405	2407#							
LF =	000012	1163#	2608							
LINEA	013574	2174	2557#							
LOAD	010306	2223	2226	2230#						
LOADY	011370	2138	2140	2233	2401#					
LOADO	010312	2231#	2235							
LO2	010304	2185	2229#							
LPADH	001446	1305#								
LPADL	001444	1305#								
LPCI	001434	1305#								
LPCO	001440	1305#								
LPMR	001436	1305#								
LPMS1	001450	1305#								
LPMS2	001452	1305#								
LPSO	001442	1305#								
LSB	012331	1851	2524#							
LSBMSG	012266	2091	2519#							
MAT	013726	1853	2566#							
MAX	001420	1297#	1605*	1612	1616*	2205*	2210	2212*	2216	
MAXTST	010204	2208	2210#							
MEND	012777	1758	2542#							
MESP	012761	2275	2541#							
MESR	012746	2272	2540#							
METST	013733	2567#								
MIN	001414	1295#	1617*	1666	2204*	2207	2209*	2215		
MINUS	012235	2431	2514#							
MLSB	013720	1672	2565#							

























ADDM	1198#	1923													
BICM	1186#	1481													
CLRM	1181#	1552	1712												
CMPM	1204#														
COMMEN	1163#														
DUMWRN	1228#														
ENDCOM	1163#														
ERROR	1163#	1309	1450	1459	1464	1469	1473	1483	1494	1508	1535	1550	1559	1567	1575
	1584	1592	1604	1627	1639	1647	1654	2361	2610						
ESCAPE	1163#	1307													
GETPRI	1163#														
GETSWR	1163#														
INCRM	1174#	1477	1725	1920											
MOVEI	170#	1477	1481	1484	1490	1492	1499	1510	1518	1551	1725	1727	1729	1920	1921
	1923	2345	2352	2364	2366	2610									
MOVEM	157#	1419	1477	1481	1488	1498	1503	1517	1541	1545	1552	1596	1608	1712	1723
	1725	1861	1911	1914	1920	1962	2018	2346	2354						
MOVEMR	1210#	1723	1914	2018	2346										
MOVERO	1169#	1484	1492	1510	1551	1729									
MULT	1163#														
NEWTST	1163#	1446	1455	1461	1466	1470	1475	1486	1496	1526	1537	1553	1561	1569	1578
	1586	1594	1641	1648	1655	1682	1691	1699							
POP	1163#	2602	2609	2612	2615										
PUSH	1163#	2602	2609	2612	2615										
REPORT	1163#														
SCOPE	1163#	1455	1461	1466	1470	1475	1486	1496	1512	1537	1553	1561	1569	1578	1586
	1594	1641	1648	1655	1682	1691	1699								
SETPRI	1163#														
SETTRA	2614#														
SETUP	1163#	1348													
SKIP	1163#	1391	1394	1397	1400	1403	1406	1425	1436	1438	1440	1453	1520	1539	1679
	1718	2245	2460	2462											
SLASH	1163#														
SPACE	1163#														
STARS	1163#	1220	1222	1223	1446	1455	1461	1466	1470	1475	1486	1496	1526	1537	1553
	1561	1569	1578	1586	1594	1641	1648	1655	1682	1691	1699	2508	2600	2602	2604
	2605	2606	2608	2609	2611	2612	2614	2615							
SWRSU	1163#	1348#													
TOUT	1305#	2610													
TRMTRP	2614#														
TSTBM	1193#	1490	1499	1727	1921	2364									
TYPBIN	1163#														
TYPDEC	1163#	2508													
TYPNAM	1163#														
TYPNUM	1163#														
TYPOCS	1163#	1334	1428	1720	1735	1852	1856	1864	1933	1938	2088	2175	2178	2278	
TYPOCT	1163#	2606	2610												
TYPTXT	1163#	2610													
SCAL.	748#	2610													
SDMAST	914#														
SDMDT	1048#														
SDMAST	772#														
SSCMRE	1223#														
SSCMTM	1223#														
SSESCA	1163#														
SSNEWT	1163#	1446	1455	1461	1466	1470	1475	1486	1496	1526	1537	1553	1561	1569	1578



	1586	1594	1641	1648	1655	1682	1691	1699
\$\$SET	2614#							
\$\$SETM	1348#							
\$\$SKIP	1163#	1539	1679					
.EQUAT	1157#	1163						
.HEADE	1157#	1162						
.KMADR	55#	1305						
.KSIS	184#	1350						
.LOADL	459#	2610						
.LPAIN	209#	2610						
.PUTCS	418#	2610						
.RESET	329#	2610						
.SETUP	1159#	1224						
.SWRHI	1159#	1164						
.SWRLO	1164#							
.UTK	699#	2610						
.\$ACT1	1160#	1220						
.\$APT8	1160#	1223#						
.\$APTH	1160#	1222						
.\$APTY	1160#	2609						
.\$CATC	1157#	1215						
.\$CMTA	1157#	1223						
.\$EOP	1157#	2508						
.\$ERRO	1158#	2605						
.\$ERRT	1159#	2606						
.\$INLP	652#	2610						
.\$MMAC	141#							
.\$OUTL	610#	2610						
.\$PARM	1158#							
.\$POWE	1158#	2615						
.\$RAND	1160#							
.\$RDOC	1160#	2602						
.\$READ	1158#	2600						
.\$SAVE	1158#							
.\$SCOP	1158#	2604						
.\$SPAC	1159#							
.\$SWDO	1159#							
.\$TLKW	511#	2610						
.\$TOUT	1305#	2610						
.\$TRAP	1159#	2614						
.\$TYPD	1160#	2612						
.\$TYPE	1159#	2608						
.\$TYPO	1158#	2611						

. ABS. 062300 000 CON RW ABS GBL D  
 000000 001 CON RW REL LCL I

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0

CRLPKC,CRLPKC/CRF=CRLPAB.MAC,CRLPKC.P11  
 RUN-TIME: 24 12 1 SECONDS  
 RUN-TIME RATIO: 139/39=3.5  
 CORE USED: 36K (71 PAGES)

LPA-AD11K TEST MD-11-CRLPKC  
CRLPKC.P11 14-AUG-80 13:59

MACY11 30G(1063) 24-OCT-80 09:48 L 8  
PAGE 46-2  
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0102





```

460                                     ;*THIS CODE WILL BE DOWN LOADED INTO BOTH
461                                     ;*KMC-11'S. THE CODE RUNS ASYNCHRONOUS TO THE PDP-11 CODE
462                                     ;*WE SYNC THROUGH COMMANDS PASSED VIA THE OUT*/IBUS* REGS.
463                                     ;*
464
465
466 DRLPX2: ;JUMP TABLE USED FOR COMMANDS
467 042000 BR STARTU ;GOTO START
(2) 042000 100407 .WORD .$$$
468 042002 BR CMNOP ;NOP=1
(2) 042002 100420 .WORD .$$$
469 042004 BR RDSILO ;=2 READ SILO PUT IN BSEL4
(2) 042004 100430 .WORD .$$$
470 042006 BR WRSILO ;=3 READ BSEL4 PUT IN SILO.
(2) 042006 100432 .WORD .$$$
471 042010 BR RDCMND ;=4 READ FAST PATH PUT IN BSEL4
(2) 042010 100434 .WORD .$$$
472 042012 BR WRCMND ;=5 READ BSEL4, PUT IN FAST PATH.
(2) 042012 100436 .WORD .$$$
473 042014 BR SAMP ;=6 TAKE AN A/D SAMPLE
(2) 042014 100440 .WORD .$$$
474
475                                     ;START OF U CODED
476
477
478 STARTU:
479 042016 MOVE # 0,BREG
(3) 042016 000400 .WORD .$$$
480 042020 MOVE BREG,OUT1 <0> ;CLEAR UNIBUS CSRS
(3) 042020 061220 .WORD .$$$
481 042022 MOVE BREG,OUT1 <2>
(3) 042022 061222 .WORD .$$$
482 042024 MOVE BREG,OUT1 <3>
(3) 042024 061223 .WORD .$$$
483 042026 MOVE BREG,OUT1 <4>
(3) 042026 061224 .WORD .$$$
484 042030 MOVE BREG,OUT1 <5>
(3) 042030 061225 .WORD .$$$
485 042032 MOVE BREG,OUT1 <6>
(3) 042032 061226 .WORD .$$$
486 042034 MOVE BREG,OUT1 <7>
(3) 042034 061227 .WORD .$$$
487 042036 MOVE BREG,SPAD <6>
(3) 042036 063226 .WORD .$$$
488
489 CMNOP: MOVE INP0 <12>,OUT1 <0> ;READ STATUS
(3) 042040 021240 .WORD .$$$
490 042042 MOVE # 377,BREG
(3) 042042 000777 .WORD .$$$
491 042044 MOVE BREG,OUT1 <2> ;INDICATE READY FOR COMMAND.
(3) 042044 061222 .WORD .$$$
492
493 LOOP: MOVE INP0 <12>,OUT1 <0> ;READ STATUS
(3) 042046 021240 .WORD .$$$
494
495 MOVE INP1 <2>,SPAD <0> ;READ COMMAND REG.
  
```



```
(3) 042050 123040 .WORD .$$$.  
496 042052 101423 BZ LOOP ;NO COMMAND THEN LOOP  
(2) 042052 101423 .WORD .$$$.  
497  
498 042054 MOVE INP1 <2>,SPAD <0> ;RE-READ COMMAND.  
(3) 042054 123040 .WORD .$$$.  
499  
500 042056 BR SPAD <0> ;BR BASED ON CMND.  
(2) 042056 160600 .WORD .$$$.  
501 ;NO-USER PROTECTION OFFERED.  
502 ;IF YOU ENTER WRONG CODE -  
503 ;YOU LOSE.  
504  
505  
506 ;ROUTINE TO READ THE SILO, PUT IN  
507 ;*BUS REG 4  
508 ;CMD=2  
509  
510 042060 RDSILO: MOVE INP0 <10>,OUT1 <4> ;READ SILO.  
(3) 042060 021204 .WORD .$$$.  
511 ;WRITE *BUS  
512 042062 BR CMNOP ;RETURN.  
(2) 042062 100420 .WORD .$$$.  
513  
514 ;ROUTINE TO WRITE SILO, READ DATA FROM  
515 ;*BUS REG 4  
516 ;CMD=3  
517  
518  
519  
520 042064 WRSILO: MOVE INP1 <4>,OUT0 <10> ;READ DATA IN *BUS  
(3) 042064 122110 .WORD .$$$.  
521 ;WRITE SILO.  
522 042066 BR CMNOP  
(2) 042066 100420 .WORD .$$$.  
523  
524 ;ROUTINE TO READ FAST PATH (CMND) REG.  
525 ;PUT IN *BUS REG 4  
526 ;CMD=4  
527  
528  
529  
530 042070 RDCMND: MOVE INP0 <11>,OUT1 <4> ;READ FAST PATH  
(3) 042070 021224 .WORD .$$$.  
531 ;WRITE *BUS.  
532 042072 BR CMNOP ;RETURN  
(2) 042072 100420 .WORD .$$$.  
533  
534 ;ROUTINE TO WRITE FAST PATH (CMND) REG.  
535 ;TAKE DATA FROM *BUS REG 4.  
536 ;CMD=5  
537  
538  
539  
540 042074 WRCMND: MOVE INP1 <4>,OUT0 <11> ;READ DATA IN *BUS  
(3) 042074 122111 .WORD .$$$.
```

```

541                                     ;WRITE INTO FAST PATH.
542 042076 BR CMNOP ;RETURN.
(2) 042076 100420 .WORD .$$$.
```

543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556

```

; THIS ROUTINE TAKES AN A/D SAMPLE.
; CALL= CMND 6 IN BSEL2
; THESE REGS. MUST BE SET UP IN ADVANCE.
; BSEL 3 MUST CONTAIN READ CODE FOR A/D BUFFER.
; BSEL 4,5 MUST CONTAIN A/D CSR SETTING.
; BSEL 6 MUST CONTAIN WRITE CODE FOR A/D CSR
; BSEL 7 MUST CONTAIN READ CODE FOR A/D CSR
; BSEL 3,6,7 WILL REMAIN UNEFFECTED.
; BSEL 4,5 WILL CONTAIN A/D SAMPLE.
; BSEL2 WILL CONTAIN CODE 377 WHEN DONE.
```

567 042100 WPMC SAMP  
(4) 042100 020640 .WORD .\$\$\$.  
(3) 042102 103040 .WORD .\$\$\$.

```

568 042104 MOVE INP1 <6>,OUT0 <11> ;SEND A/D WRITE CODE.
(3) 042104 122151 .WORD .$$$.
```

569 042106 WPMC SAMP1  
(4) 042106 020640 .WORD .\$\$\$.  
(3) 042110 103043 .WORD .\$\$\$.

```

570 042112 MOVE INP1 <4>,OUT0 <11> ;SEND LOW BYTE CSR INFO.
(3) 042112 122111 .WORD .$$$.
```

571 042114 WPMC SAMP2  
(4) 042114 020640 .WORD .\$\$\$.  
(3) 042116 103046 .WORD .\$\$\$.

```

572 042120 MOVE INP1 <5>,OUT0 <11> ;SEND HIGH BYTE CSR INFO.
(3) 042120 122131 .WORD .$$$.
```

573 042122 WPMC SLOOP  
(4) 042122 020640 .WORD .\$\$\$.  
(3) 042124 103051 .WORD .\$\$\$.

```

574 042126 MOVE INP1 <7>,OUT0 <11> ;SEND READ CODE TO GET A/D CSR.
(3) 042126 122171 .WORD .$$$.
```

575 042130 WPMC SAMP3  
(4) 042130 020640 .WORD .\$\$\$.  
(3) 042132 103054 .WORD .\$\$\$.

576 042134 WTMM SLOOP1  
(4) 042134 020640 .WORD .\$\$\$.  
(4) 042136 061620 .WORD .\$\$\$.  
(3) 042140 103056 .WORD .\$\$\$.

```

577 042142 MOVE INP0 <11>,BREG
(3) 042142 020620 .WORD .$$$.
```

578 042144 MOVE BREG,SPAD <0>  
(3) 042144 063220 .WORD .\$\$\$.

579 042146 WTMM SLOOP2  
(4) 042146 020640 .WORD .\$\$\$.  
(4) 042150 061620 .WORD .\$\$\$.  
(3) 042152 103063 .WORD .\$\$\$.

```

580 042154 MOVE INP0 <11>,BREG
(3) 042154 020620 .WORD .$$$.
```

581 042156 BB7 CMNOP ;ABORT IF A/D BIT 15=1



(2)	042156	103420	.WORD	.\$\$\$.	
582	042160		MOVE	SPAD <0>,BREG	
(3)	042160	060600	.WORD	.\$\$\$.	
583	042162		BB7	LOPE	
(2)	042162	103473	.WORD	.\$\$\$.	
584	042164		BR	SLOOP	; IF A/D NOT DONE,EXIT.
(2)	042164	100451	.WORD	.\$\$\$.	
585	042166		LOPE: MOVE	INP1 <3>,OUT0 <11>	;ISSUE READ A/B BUFFER.
(3)	042166	122071	.WORD	.\$\$\$.	
586	042170		WTMM	SLOOP3	
(4)	042170	020640	.WORD	.\$\$\$.	
(4)	042172	061620	.WORD	.\$\$\$.	
(3)	042174	103074	.WORD	.\$\$\$.	
587	042176		MOVE	INP0 <11>,OUT1 <4>	
(3)	042176	021224	.WORD	.\$\$\$.	
588	042200		WTMM	SLOOP4	
(4)	042200	020640	.WORD	.\$\$\$.	
(4)	042202	061620	.WORD	.\$\$\$.	
(3)	042204	103100	.WORD	.\$\$\$.	
589	042206		MOVE	INP0 <11>,OUT1 <5>	
(3)	042206	021225	.WORD	.\$\$\$.	
590	042210		BR	CMNOP	
(2)	042210	100420	.WORD	.\$\$\$.	
591	042212	177777	.WORD	-1	
592		000001	.END		

ADDRES= 177777	SAMP 042100	.ADDWC= 000020	.DMEM = 002400	.SELB = 000220
CLK = 000020	SAMP1 042106	.AND = 000260	.DNOP = 000000	.SIMM = 000000
CMNOP 042040	SAMP2 042114	.BB0 = 002000	.DOUT0= 002000	.SINO = 020000
DRLPX2 042000 G	SAMP3 042130	.BB1 = 002400	.DOUT1= 001000	.SIN1 = 120000
LOOP 042046	SLOOP 042122	.BB4 = 003000	.DSPAD= 003000	.SMEM = 040000
LOPE 042166	SLOOP1 042134	.BB7 = 003400	.DSPBR= 003400	.SUB = 000340
MARHLD= 000000	SLOOP2 042146	.BC = 001000	.DO = 000400	.SUBWC= 000040
MARINC= 014000	SLOOP3 042170	.BR = 000400	.FO = 000020	.SUB2C= 000360
MARLD = 010000	SLOOP4 042200	.BSBRG= 160000	.INC = 000060	.SO = 020000
MARLDX= 004000	STARTU 042016	.BSIMM= 100000	.LORN = 000240	.XOR = 000320
PAGE0 = 000000	WRCMND 042074	.BSMEM= 140000	.MINUS= 000360	..\$\$\$ = 100420
PAGE1 = 001000	WRSILO 042064	.BZ = 001400	.MO = 004000	..LOC = 042040
PAGE2 = 002000	\$\$\$SER= 000001	.CO = 000400	.OR = 000300	.2A = 000120
PAGE3 = 003000	. = 042214	.DBR = 000400	.PLUS = 000000	.2AWC = 000140
RDCMND 042070	.ADC = 000100	.DBRSH= 001400	.SBREG= 060000	
RDSILO 042060	.ADD = 000000	.DEC = 000160	.SELA = 000200	

. ABS.	042214	000	OVR	RW	ABS	LCL	D
	000000	001	CON	RW	ABS	LCL	I
ABCODE	004000	002	CON	RW	REL	LCL	I

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0

CRLPX2,CRLPX2=CRLPX2  
 RUN-TIME: 3 3 0 SECONDS  
 RUN-TIME RATIO: 34/7=4.4  
 CORE USED: 36K (71 PAGES)