

# LPA/AR11

DIAGNOSTIC TEST 3  
MD-11-DRLPE-A

EP-DRLPE-A-DL  
COPYRIGHT © 1978  
FICHE 1 OF 1

MAR 1978  
**digital**  
MADE IN USA

The image displays a grid of 150 small, illegible data tables or charts, arranged in 10 columns and 15 rows. Each cell in the grid contains a small, dark, rectangular area with faint, unreadable text or patterns, likely representing diagnostic test results or data points. The overall appearance is that of a dense, multi-page document where the individual data points are too small to be discernible.

E0F1DRLP0A8BQ411

00010000  
-----

780223 IDENTIFICATION 00F10 411

H0R1DRLPEASEQ

00010000

780223  
SEQ 0001

PRODUCT CODE: MAINDEC-11-DRLPE-A-D  
PRODUCT NAME: LPA/AR11 DIAGNOSTIC TEST III  
DATE: JAN 1978  
MAINTAINER: DIAGNOSTIC GROUP

COPYRIGHT (C) 1978  
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 TO 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.

## ABSTRACT

SEQ 0002

THE WRAPAROUND DIAGNOSTIC ENABLES THE AR11 TO TEST ITSELF, IN CONJUNCTION WITH A G5036 WRAPAROUND MODULE, WHICH IS CONNECTED TO THE AR11 THROUGH A BCOB-R CABLE. THE TWO D/A CONVERTERS ARE COMBINED IN VARIOUS MANNERS USING RESISTIVE DIVIDERS ON THE G5036 MODULE AND SENT BACK INTO THE A/D INPUTS. CONSEQUENTLY, BOTH X AND Y D/A'S ARE TESTED BY THE A/D CONVERTER, AND THE A/D IS TESTED BY THE D/A CONVERTERS. BECAUSE THE D/A'S ARE DIVIDED DOWN BEFORE GOING INTO THE A/D, TEST RESOLUTION FAR BETTER THAN 1 LSB IS ATTAINED. IN ADDITION, THE ANALOG POWER SUPPLY LEVELS (+/- 14 VOLTS) ARE DIVIDED DOWN AND USED AS INPUTS, ALLOWING TEST OF THE DC-TO-DC CONVERTER; THE FOUR SCOPE CONTROL LOGIC OUTPUTS (ERASE L, NON-STORE L, CH02 L, AND WRITE-THRU L) ARE USED AS INPUTS, ALLOWING PARAMETRIC TESTS OF THEIR HIGH AND LOW OUTPUT LEVELS; THE CH02 L OUTPUT IS FED BACK INTO THE ERASE RET INPUT, ALLOWING TEST OF THE STORAGE SCOPE ERASE RETURN HANDSHAKING LOGIC; AND THE SCOPE INTENSIFY OUTPUT IS FED BACK INTO THE A/D EXTERNAL START INPUT, ALLOWING TESTS OF BOTH FUNCTIONS. SINCE ALL A/D INPUT CHANNELS ARE USED (AT DIFFERENT INPUT VOLTAGE LEVELS), THE WRAPAROUND MODULE ALSO PROVIDES A COMPLETE FUNCTIONAL CHECK ON THE A/D INPUT MULTIPLEXER.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZARC-B". IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE AR11 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-DZARC-B". YOU SHOULD RUN "MD-11-DRLPA" BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 11.

## 2. REQUIREMENTS

## 2.1 EQUIPMENT

PDP-11 FAMILY COMPUTER WITH 16K WORDS OF MEMORY  
AR11 HEX OPTION MODULE INSTALLED  
G5036 WRAPAROUND MODULE AND BCOB-R CABLE  
TELETYPE

## 2.2 STORAGE

THIS PROGRAM USES LESS THAN 8K OF MEMORY.

## 3. LOADING PROCEDURE

PROCEDURE FOR NORMAL BINARY TAPES SHOULD BE FOLLOWED.

4. STARTING PROCEDURE  
-----

4.1 CONTROL SWITCH SETTINGS

THIS PROGRAM HAS BEEN MODIFIED TO RUN WITH OR WITHOUT A HARDWARE SWITCH REGISTER.

STANDARD PDP-11 FORMAT

```

SW 15 = 1      HALT ON ERROR
SW 14 = 1      LOOP ON TEST
SW 13 = 1      INHIBIT ERROR TYPEOUTS
SW 12 = 1      FORCED PRINTOUT ENABLE
SW 11 = 1      INHIBIT ITERATIONS
SW 10 = 1      BELL ON ERROR
SW 09 = 1      LOOP ON ERROR
SW 08 = 1      LOOP ON TEST IN SWR <7:0>
    
```

REFER TO 9. FOR SOFTWARE SWITCH REGISTER CONTROL

4.2 STARTING ADDRESS OR ADDRESSES

```

200    IS THE STARTING ADDRESS OF THE WRAPAROUND TEST.
204    IS THE RESTART ADDRESS OF THE WRAPAROUND TEST.
210    IS THE STARTING ADDRESS FOR THE OPTION TEST AREA.
    
```

5. OPERATING PROCEDURE  
-----

THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH REGISTER WHEN REQUESTED BY THE PROGRAM OR AN ERROR WILL OCCUR. ONCE STARTED THE TEST WILL RUN IN ITS NORMAL MANNER WITHOUT OPERATOR INTERVENTION OR SWITCH SELECTION.

6. ERRORS  
-----

THIS PROGRAM USES THE DIAGNOSTIC 'SYSMAC' PACKAGE FOR ERROR REPORTING AND TYPEOUT. REFER TO THE "ERROR POINTER TABLE" FOR TYPE AND DESCRIPTION OF ERRORS.

7. RESTRICTIONS  
-----

SWITCH ON G5036 MUST BE IN THE '0' POSITION.

```

*****
THE WRAPAROUND (G5036) MODULE MUST BE CONNECTED AS FOLLOWS:
  AR11 TO BC08R CONNECTION A-A, VV-VV
  BC08R TO G5036 CONNECTION "UPSIDE-DOWN" A-VV, VV-A
*****
    
```

8. MISCELLANEOUS  
-----

## 8.1 EXECUTION TIME

THE FIRST PASS WILL TAKE APPROXIMATELY 10 MINUTES FOR COMPLETION.  
ADDITIONAL PASSES WILL TAKE APPROXIMATELY 40 MIN. FOR COMPLETION.  
THE END OF A PASS IS INDICATED BY THE 'END PASS \*X' MESSAGE.

## 8.2 DEVICE ADDRESS PROGRAM LOCATIONS

LOCATION \$BASE CONTAINS THE AR11 BASE DEVICE ADDRESS <170400>  
LOCATION \$VECT1 CONTAINS THE AR11 BASE INTERRUPT VECTOR <340>  
LOCATION \$FILLS CONTAINS THE TTY FILLER CHARACTER COUNT  
LOCATION \$NULL CONTAINS THE TTY FILLER CHARACTER

\*NOTE: IF THESE LOCATIONS ARE CHANGED, THE OPERATOR MUST START  
THE TEST AGAIN AT LOC. 200. THE PROGRAM WILL USE THE BASE  
ADDRESS AND VECTOR AND UPDATE THE ACTUAL PROGRAM VALUES.

## 8.3 XXDP/ACT/APT NOTES

THIS PROGRAM IS A CHAINABLE PROGRAM UNDER XXDP/ACT.  
THE APT HOOKS HAVE BEEN INSTALLED BUT NOT TESTED.

## 8.4 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 \$UTK:
- 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.

9. SOFTWARE SWITCH REGISTER OPERATION  
-----

THE PROGRAM SUPPORTS NON-SWITCH REGISTER CPU TYPES.  
A CHANGE IN SWR VALUE IS ACCOMPLISHED BY TYPING A "CTRL G".  
THE RESPONSE WILL BE "SWR = " AND WAIT FOR A NEW VALUE.  
THE OPERATOR NOW INPUTS THE NEW VALUE AND TERMINATES WITH A "CR".

10. TABLE OF CONTENTS  
-----

ATTACHED

11. 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-DRLPA. WHEN THE USER RUNS DRLPA HE CAN GENERALLY TELL WHICH OPTION DIAGNOSTIC (LEVEL 3) TO RUN NEXT. M8254 AND M8200-YC ERRORS MAY "LOOK" ALIKE AND DRLPA 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 DRLPA 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:

SEQ 0007

OPTION -----	GROUP -----	DIAG. # -----	DIAG. TITLE -----
LPA11-KX	LEVEL 2	MD-11-DRLPA	LPA11-K SYSTEM DIAG.
M8254	"B"	MD-11-DRLPN	M8254 (IPBM) DIAG.
AA11-K	A	MD-11-DRLPB	AA11-K DIAG.
	B	MD-11-DZAAC	AA11-K DIAG.
AR11	A	MD-11-DRLPC	LPA/AR11 DIAG. #1
	A	MD-11-DRLPD	LPA/AR11 DIAG. #2
	A	MD-11-DRLPE	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-DRLPF
B		MD-11-DZDRG	DR11-K DIAG.
KW11-K	A	MD-11-DRLPG	LPA/KW11-K DIAG.
	B	MD-11-DZKWK	KW11-K DIAG.
LPS11	A	MD-11-DRLPH	LPA/LPS11 DIAG. #1
	A	MD-11-DRLPI	LPA/LPS11 DIAG. #2
	A	MD-11-DRLPJ	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-DRLPK	LPA/AD11-K DIAG.
	B	MD-11-DZADL	AD11-K DIAG.
M8200-YC	B	MD-11-DZLPL	LPA/M8200-YC BASIC MICRO-CPU R/W TEST
	B	MD-11-DZLPM	LPA/M8200-YC JMP+ROM READ TEST

41	BASIC DEFINITIONS
155	OPERATIONAL SWITCH SETTINGS
167	TRAP CATCHER
176	STARTING ADDRESS(ES)
181	ACT11 HOOKS
192	APT PARAMETER BLOCK
214	COMMON TAGS
262	APT MAILBOX-ETABLE
330	ERROR POINTER TABLE
467	PROGRAM START-UP
492	INITIALIZE THE COMMON TAGS
535	TYPE PROGRAM NAME
542	GET VALUE FOR SOFTWARE SWITCH REGISTER
620	A TO D CHANNEL NUMBERS
659	
660	TEST NUMBER
661	ABSTRACT
662	T1 TEST THAT "ERASE" AND DONE CAN BE SET AND CLEARED
705	T2 EXTERNAL A TO D START FROM INTENSIFY
748	T3 TEST THAT CH 0 IS A BIPOLAR GROUND
768	T4 TEST THAT CH 40 IS A UNIPOLAR GROUND
784	T5 TEST THAT CH 2 IS AT +1 VOLT BIPOLAR
800	T6 TEST THAT CH 42 IS AT +1 VOLT UNIPOLAR
816	T7 TEST THAT CH 3 IS AT +2.5 BIPOLAR
832	T10 TEST THAT CH 43 IS AT +2.5 UNIPOLAR
848	T11 TEST THAT CH 4 IS AT -2.5 BIPOLAR
863	T12 TEST THAT CH 57 IS AT +4 VOLTS UNIPOLAR
878	T13 TEST THAT CH 41 IS LESS THAN +200 MVOLTS UNIPOLAR
926	T14 TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH
943	T15 TEST THAT ERASE L (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
963	T16 TEST THAT WRITE-THRU (CH 54) IS GREATER THAN +3 VOLTS WHEN HIGH
980	T17 TEST THAT WRITE-THRU (CH 54) IS LESS THAN +400 MVOLTS WHEN LOW
998	T20 TEST THAT NON-STORE (CH 55) IS LESS THAN +400 MVOLTS WHEN LOW
1016	T21 TEST THAT NON-STORE L (CH 55) IS GREATER THAN +3 VOLTS WHEN HIGH
1034	T22 TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH
1052	T23 TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW
1073	T24 A TO D DIFFERENTIAL LINEARITY TEST
1101	T25 X D/A DIFFERENTIAL LINEARITY USING CH 5
1151	T26 Y D/A DIFFERENTIAL LINEARITY USING CH 6
1211	T27 A TO D POSITIVE MULTIPLEXER SETTling TEST
1244	T30 A TO D NEGATIVE MULTIPLEXER SETTling TEST
1289	T31 A/D RMS NOISE TEST ON CHANNEL 7 - BIPOLAR
1310	T32 A/D PEAK NOISE TEST ON CHANNEL 7 - BIPOLAR
1331	T33 A/D RMS NOISE TEST ON CHANNEL 47 - UNIPOLAR
1352	T34 A/D PEAK NOISE TEST ON CHANNEL 47 - UNIPOLAR
1373	T35 X DAC RMS NOISE TEST ON CHANNEL 5
1396	T36 X DAC PEAK NOISE TEST ON CHANNEL 5
1422	T37 Y DAC RMS NOISE TEST ON CHANNEL 6
1446	T40 Y DAC PEAK NOISE TEST ON CHANNEL 6
1490	T41 BIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
1510	T42 UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
1529	T43 BIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
1556	T44 UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
1582	T45 X D/A OFFSET USING CH 5



MAINDEC-11-DRLPE-A MACY11 27(654) 14-DEC-77 20:24  
DRLPE.P11 TABLE OF CONTENTS

SEQ 0009

1603	T46	Y D/A OFFSET USING CH 6
1623	T47	CALIBRATION USING CHANNEL 11 - X DAC VS. A/D
1649	T50	CALIBRATION USING CHANNEL 12 - Y DAC VS. A/D
1674	T51	LINEARITY TEST USING CH 6 - Y DAC VS. A/D
1739	T52	LINEARITY TEST USING CH 5 - X DAC VS. A/D
1814	T53	DETERMINE IF MORE ARI1'S ARE TO BE TESTED
1833		
1835		END OF PASS ROUTINE
1876		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1943		SUCCESSIVE APPROX. SUBROUTINE
2053		MISC. SUBROUTINES
2163		ASCII MESSAGES AND ERROR POINTERS
2475		DIFFERENTIAL LINEARITY SUBROUTINE
2778		PARAMETER ADJUSTMENT ROUTINE
2825		SUBROUTINE TO CONVERT RD TO DECIMAL PERCENTAGE OF 1022.
2856		SCOPE HANDLER ROUTINE
2922		ERROR HANDLER ROUTINE
2976		TTY INPUT ROUTINE
3115		READ AN OCTAL NUMBER FROM THE TTY
3154		ERROR MESSAGE TYPEOUT ROUTINE
3202		POWER DOWN AND UP ROUTINES
3253		BINARY TO OCTAL (ASCII) AND TYPE
3330		TYPE ROUTINE
3409		APT COMMUNICATIONS ROUTINE
3466		TRAP DECODER
3489		TRAP TABLE

.REM [

LPA.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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30

30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83

001100

000011  
000012  
000015  
000200  
177776

177774  
177772  
177570  
177570

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000006  
000007

000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340

100000  
040000

.TITLE MAINDEC-11-DRLPE-A  
;\*COPYRIGHT (C) 1978  
;\*DIGITAL EQUIPMENT CORP.  
;\*MAYNARD, MASS. 01754  
\*  
;\*PROGRAM BY EDWARD C. BADGER  
\*  
;\*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
;\*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.  
\*  
.SBTTL BASIC DEFINITIONS

.\*INITIAL ADDRESS OF THE STACK POINTER \*\*\* 1100 \*\*\*

STACK= 1100  
.EQUIV EMT,ERROR ;;;BASIC DEFINITION OF ERROR CALL  
.EQUIV IOT,SCOPE ;;;BASIC DEFINITION OF SCOPE CALL

.\*MISCELLANEOUS DEFINITIONS

HT= 11 ;;;CODE FOR HORIZONTAL TAB  
LF= 12 ;;;CODE FOR LINE FEED  
CR= 15 ;;;CODE FOR CARRIAGE RETURN  
CRLF= 200 ;;;CODE FOR CARRIAGE RETURN-LINE FEED  
PS= 177776 ;;;PROCESSOR STATUS WORD  
.EQUIV PS,PSW  
STKLMT= 177774 ;;;STACK LIMIT REGISTER  
PIRQ= 177772 ;;;PROGRAM INTERRUPT REQUEST REGISTER  
DSWR= 177570 ;;;HARDWARE SWITCH REGISTER  
DDISP= 177570 ;;;HARDWARE DISPLAY REGISTER

.\*GENERAL PURPOSE REGISTER DEFINITIONS

R0= %0 ;;;GENERAL REGISTER  
R1= %1 ;;;GENERAL REGISTER  
R2= %2 ;;;GENERAL REGISTER  
R3= %3 ;;;GENERAL REGISTER  
R4= %4 ;;;GENERAL REGISTER  
R5= %5 ;;;GENERAL REGISTER  
R6= %6 ;;;GENERAL REGISTER  
R7= %7 ;;;GENERAL REGISTER  
SP= %6 ;;;STACK POINTER  
PC= %7 ;;;PROGRAM COUNTER

.\*PRIORITY LEVEL DEFINITIONS

PR0= 0 ;;;PRIORITY LEVEL 0  
PR1= 40 ;;;PRIORITY LEVEL 1  
PR2= 100 ;;;PRIORITY LEVEL 2  
PR3= 140 ;;;PRIORITY LEVEL 3  
PR4= 200 ;;;PRIORITY LEVEL 4  
PR5= 240 ;;;PRIORITY LEVEL 5  
PR6= 300 ;;;PRIORITY LEVEL 6  
PR7= 340 ;;;PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000  
SW14= 40000

84 020000  
 85 010000  
 86 004000  
 87 002000  
 88 001000  
 89 000400  
 90 000200  
 91 000100  
 92 000040  
 93 000020  
 94 000010  
 95 000004  
 96 000002  
 97 000001  
 98  
 99  
 100  
 101  
 102  
 103  
 104  
 105  
 106  
 107  
 108  
 109  
 110 100000  
 111 040000  
 112 020000  
 113 010000  
 114 004000  
 115 002000  
 116 001000  
 117 000400  
 118 000200  
 119 000100  
 120 000040  
 121 000020  
 122 000010  
 123 000004  
 124 000002  
 125 000001  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137

SW13= 20000  
 SW12= 10000  
 SW11= 4000  
 SW10= 2000  
 SW09= 1000  
 SW08= 400  
 SW07= 200  
 SW06= 100  
 SW05= 40  
 SW04= 20  
 SW03= 10  
 SW02= 4  
 SW01= 2  
 SW00= 1  
 .EQUIV SW09,SW9  
 .EQUIV SW08,SW8  
 .EQUIV SW07,SW7  
 .EQUIV SW06,SW6  
 .EQUIV SW05,SW5  
 .EQUIV SW04,SW4  
 .EQUIV SW03,SW3  
 .EQUIV SW02,SW2  
 .EQUIV SW01,SW1  
 .EQUIV SW00,SW0

;\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000  
 BIT14= 40000  
 BIT13= 20000  
 BIT12= 10000  
 BIT11= 4000  
 BIT10= 2000  
 BIT09= 1000  
 BIT08= 400  
 BIT07= 200  
 BIT06= 100  
 BIT05= 40  
 BIT04= 20  
 BIT03= 10  
 BIT02= 4  
 BIT01= 2  
 BIT00= 1  
 .EQUIV BIT09,BIT9  
 .EQUIV BIT08,BIT8  
 .EQUIV BIT07,BIT7  
 .EQUIV BIT06,BIT6  
 .EQUIV BIT05,BIT5  
 .EQUIV BIT04,BIT4  
 .EQUIV BIT03,BIT3  
 .EQUIV BIT02,BIT2  
 .EQUIV BIT01,BIT1  
 .EQUIV BIT00,BIT0

;\*BASIC "CPU" TRAP VECTOR ADDRESSES

DRLPE.P:1 BASIC DEFINITIONS

```

138      000004      ERRVEC= 4      ;; TIME OUT AND OTHER ERRORS
139      000010      RESVEC= 10     ;; RESERVED AND ILLEGAL INSTRUCTIONS
140      000014      TBITVEC=14     ;; "T" BIT
141      000014      TRIVEC= 14     ;; TRACE TRAP
142      000014      BPTVEC= 14     ;; BREAKPOINT TRAP (BPT)
143      000020      IOTVEC= 20     ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
144      000024      PWRVEC= 24     ;; POWER FAIL
145      000030      EMTVEC= 30     ;; EMULATOR TRAP (EMT) **ERROR**
146      000034      TRAPVEC=34     ;; "TRAP" TRAP
147      000060      TKVEC= 60      ;; TTY KEYBOARD VECTOR
148      000064      TPVEC= 64      ;; TTY PRINTER VECTOR
149      000240      PIRQVEC=240    ;; PROGRAM INTERRUPT REQUEST VECTOR
150      170400      ABASE=170400
151      000340      AVECT1=340
152      000200      APRIOR=200

153      .SBTTL OPERATIONAL SWITCH SETTINGS
154      *
155      *          SWITCH          USE
156      *          -----
157      *          15          HALT ON ERROR
158      *          14          LOOP ON TEST
159      *          13          INHIBIT ERROR TYPEOUTS
160      *          12          FORCED PRINTOUT
161      *          11          INHIBIT ITERATIONS
162      *          10          BELL ON ERROR
163      *          9           LOOP ON ERROR
164      *          8           LOOP ON TEST IN SWR<7:0>
165      *
166      .SBTTL TRAP CATCHER
167
168      000000      .=0
169      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
170      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
171      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
172      .=174
173      000174      000000      DISPREG: .WORD 0      ;; SOFTWARE DISPLAY REGISTER
174      000176      000000      SWREG: .WORD 0      ;; SOFTWARE SWITCH REGISTER
175      .SBTTL STARTING ADDRESS(ES)
176      000200      000137      001612      JMP @#BEGIN ;; JUMP TO STARTING ADDRESS OF PROGRAM
177      000204      000137      001634      JMP BEGIN1  ;; RESTART ADDRESS
178      000210      000137      001626      JMP BEGIN2  ;; STARTING ADDRESS OF OPTION TEST AREA
179
180      .SBTTL ACT11 HOOKS
181
182      ;*****
183      ;HOOKS REQUIRED BY ACT11
184      000214      000214      $SVPC=.      ;SAVE PC
185      000046      000046      .=46
186      000046      011436      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
187      000052      000052      .=52
188      000052      000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
189      000214      000214      .=$$VPC      ;; RESTORE PC
190      001000      001000      .=1000
191      .SBTTL APT PARAMETER BLOCK

```

```

192
193
194
195
196      001000
197      000024
198 000024 000200
199      000044
200 000044 001000
201      001000
202
203
204
205
206 001000
207 001000 000000
208 001002 001202
209 001004 000020
210 001006 000030
211 001010 000120
212 001012 000052

```

```

;*****
;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:
$HIPTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD 20     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 30     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 120    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

213  
214  
215  
216  
217  
218  
219 001100  
220 001100 001100  
221 001100 000000  
222 001102 000  
223 001103 000  
224 001104 000000  
225 001106 000000  
226 001110 000000  
227 001112 000000  
228 001114 000  
229 001115 001  
230 001116 000000  
231 001120 000000  
232 001122 000000  
233 001124 000000  
234 001126 000000  
235 001130 000000  
236 001132 000000  
237 001134 000  
238 001135 000  
239 001136 000000  
240 001140 177570  
241 001142 177570  
242 001144 177560  
243 001146 177562  
244 001150 177564  
245 001152 177566  
246 001154 000  
247 001155 002  
248 001156 012  
249 001157 000  
250 001160 000000  
251  
252 001162 000000  
253 001164 000000  
254 001166 000000  
255 001170 000000  
256 001172 177607 000377  
257 001176 077  
258 001177 015  
259 001200 000012  
260  
261  
262  
263  
264  
265 001202  
266 001202 000000

```

.SBTTL COMMON TAGS
;*****
;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
;USED IN THE PROGRAM.

.=1100
SCMTAG:                ;; START OF COMMON TAGS
                        .WORD    0
$TSTNM: .BYTE          00      ;; CONTAINS THE TEST NUMBER
$ERFLG: .BYTE          00      ;; CONTAINS ERROR FLAG
$ICNT:  .WORD          00      ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD          00      ;; CONTAINS SCOPE LOOP ADDRESS
$LPERR: .WORD          00      ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD          00      ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .BYTE          00      ;; CONTAINS ITEM CONTROL BYTE
$ERMAX: .BYTE          1       ;; CONTAINS MAX. ERRORS PER TEST
$ERrPC: .WORD          00      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD          00      ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD          00      ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDADR: .WORD          00      ;; CONTAINS 'GOOD' DATA
$BDADR: .WORD          00      ;; CONTAINS 'BAD' DATA
                        .WORD    00      ;; RESERVED--NOT TO BE USED
                        .WORD    00
$AUTOB: .BYTE          00      ;; AUTOMATIC MODE INDICATOR
$INTAG: .BYTE          00      ;; INTERRUPT MODE INDICATOR
                        .WORD    0
$SWR:   .WORD          DSWR     ;; ADDRESS OF SWITCH REGISTER
$DISP:  .WORD          DDI$P    ;; 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)
$REGAD: .WORD          0       ;; CONTAINS THE ADDRESS FROM WHICH ($REGD) WAS OBTAINED
$REGD:  .WORD          0       ;; CONTAINS (( $REGAD)+0)
$REGI:  .WORD          0       ;; CONTAINS (( $REGAD)+2)
$TIMES: 0               ;; MAX. NUMBER OF ITERATIONS
$ESCAPE: 0              ;; ESCAPE ON ERROR ADDRESS
$BELL:   .ASCIZ <207><377><377> ;; CODE FOR BELL
$QUES:   .ASCII  /?/     ;; QUESTION MARK
$CRLF:   .ASCII <15>     ;; CARRIAGE RETURN
$LF:     .ASCIZ '12'     ;; LINE FEED
;*****
.SBTTL APT MAILBOX-ETABLE
;*****
.EVEN
$MAIL:   ;; APT MAILBOX
$MSGTY:  .WORD    MSGTY    ;; MESSAGE TYPE CODE

```

267 001204 000000  
268 001206 000000  
269 001210 000000  
270 001212 000000  
271 001214 000000  
272 001216 000000  
273 001220 000000  
274 001222  
275 001222 000  
276 001223 000  
277 001224 000000  
278 001226 000000  
279 001230 000000  
280  
281  
282  
283  
284  
285  
286 001232 000  
287 001233 000  
288  
289  
290  
291  
292 001234 000000  
293  
294 001236 000  
295 001237 000  
296 001240 000000  
297 001242 000  
298 001243 000  
299 001244 000000  
300 001246 000  
301 001247 000  
302 001250 000000  
303 001252 000340  
304 001254 000000  
305 001256 170400  
306 001260 000000  
307 001262 000000  
308 001264 000000  
309 001266 000000  
310 001270 000000  
311 001272 000000  
312 001274 000000  
313 001276 000000  
314 001300 000000  
315 001302 000000  
316 001304 000000  
317 001306 000000  
318 001310 000000  
319 001312 000000  
320 001314 000000

\$FATAL: .WORD AFATAL  
\$TESTN: .WORD ATESTN  
\$PASS: .WORD APASS  
\$DEVCT: .WORD ADEVCT  
\$UNIT: .WORD AUNIT  
\$MSGAD: .WORD AMSGAD  
\$MSGLG: .WORD AMSGLG  
\$ETABLE: .WORD  
\$ENV: .BYTE AENV  
\$ENVM: .BYTE AENVM  
\$SWREG: .WORD ASWREG  
\$USWR: .WORD AUSWR  
\$CPUOP: .WORD ACPUOP  
\*  
\*  
\*  
\*  
\*  
\*  
\$MAMS1: .BYTE AMAMS1  
\$MTYP1: .BYTE AMTYP1  
\*  
\*  
\*  
\*  
\$MADR1: .WORD AMADR1  
\*  
\$MAMS2: .BYTE AMAMS2  
\$MTYP2: .BYTE AMTYP2  
\$MADR2: .WORD AMADR2  
\$MAMS3: .BYTE AMAMS3  
\$MTYP3: .BYTE AMTYP3  
\$MADR3: .WORD AMADR3  
\$MAMS4: .BYTE AMAMS4  
\$MTYP4: .BYTE AMTYP4  
\$MADR4: .WORD AMADR4  
\$VECT1: .WORD AVECT1  
\$VECT2: .WORD AVECT2  
\$BASE: .WORD ABASE  
\$DEVm: .WORD ADEVm  
\$CDW1: .WORD ACDW1  
\$CDW2: .WORD ACDW2  
\$DDW0: .WORD ADDW0  
\$DDW1: .WORD ADDW1  
\$DDW2: .WORD ADDW2  
\$DDW3: .WORD ADDW3  
\$DDW4: .WORD ADDW4  
\$DDW5: .WORD ADDW5  
\$DDW6: .WORD ADDW6  
\$DDW7: .WORD ADDW7  
\$DDW8: .WORD ADDW8  
\$DDW9: .WORD ADDW9  
\$DDW10: .WORD ADDW10  
\$DDW11: .WORD ADDW11

::: FATAL ERROR NUMBER  
::: TEST NUMBER  
::: PASS COUNT  
::: DEVICE COUNT  
::: I/O UNIT NUMBER  
::: MESSAGE ADDRESS  
::: MESSAGE LENGTH  
::: APT ENVIRONMENT TABLE  
::: ENVIRONMENT BYTE  
::: ENVIRONMENT MODE BITS  
::: APT SWITCH REGISTER  
::: USER SWITCHES  
::: CPU TYPE, OPTIONS  
BITS 15-11=CPU TYPE  
11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05  
11/70=06, PQQ=07, Q=10  
BIT 10=REAL TIME CLOCK  
BIT 9=FLOATING POINT PROCESSOR  
BIT 8=MEMORY MANAGEMENT  
::: HIGH ADDRESS, M.S. BYTE  
::: MEM. TYPE, BLK#1  
MEM. TYPE BYTE -- (HIGH BYTE)  
900 NSEC CORE=001  
300 NSEC BIPOLAR=002  
500 NSEC MOS=003  
::: HIGH ADDRESS, BLK#1  
MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE  
::: HIGH ADDRESS, M.S. BYTE  
::: MEM. TYPE, BLK#2  
MEM. LAST ADDRESS, BLK#2  
::: HIGH ADDRESS, M.S. BYTE  
::: MEM. TYPE, BLK#3  
MEM. LAST ADDRESS, BLK#3  
::: HIGH ADDRESS, M.S. BYTE  
::: MEM. TYPE, BLK#4  
MEM. LAST ADDRESS, BLK#4  
::: INTERRUPT VECTOR#1, BUS PRIORITY#1  
::: INTERRUPT VECTOR#2, BUS PRIORITY#2  
::: BASE ADDRESS OF EQUIPMENT UNDER TEST  
::: DEVICE MAP  
::: CONTROLLER DESCRIPTION WORD#1  
::: CONTROLLER DESCRIPTION WORD#2  
::: DEVICE DESCRIPTOR WORD#0  
::: DEVICE DESCRIPTOR WORD#1  
::: DEVICE DESCRIPTOR WORD#2  
::: DEVICE DESCRIPTOR WORD#3  
::: DEVICE DESCRIPTOR WORD#4  
::: DEVICE DESCRIPTOR WORD#5  
::: DEVICE DESCRIPTOR WORD#6  
::: DEVICE DESCRIPTOR WORD#7  
::: DEVICE DESCRIPTOR WORD#8  
::: DEVICE DESCRIPTOR WORD#9  
::: DEVICE DESCRIPTOR WORD#10  
::: DEVICE DESCRIPTOR WORD#11



321 001316 000000  
322 001320 000000  
323 001322 000000  
324 001324 000000  
325  
326  
327 001326  
328

\$DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12  
\$DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13  
\$DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14  
\$DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15

\$ETEND:

329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382

.SBTTL ERROR POINTER TABLE

;\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
;\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
;\*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
;\*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
;\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;\* EM ;:POINTS TO THE ERROR MESSAGE  
;\* DH ;:POINTS TO THE DATA HEADER  
;\* DT ;:POINTS TO THE DATA  
;\* DF ;:POINTS TO THE DATA FORMAT

\$ERRTB:

001326

; ITEM 1  
EM1 ;:ERROR ON A/D CHANNEL  
DH1 ;:ERRPC ARADD CHANNEL NOMINAL TOLERANCE ACTUAL  
DT1 ;:SERRPC ARBADD CHANL \$GDDAT SPREAD \$BDDAT  
0

001326 013062  
001330 013575  
001332 015716  
001334 000000

; ITEM 2  
EM2 ;:VC LOGIC SIGNAL HIGH OUTPUT TOO LOW  
DH2 ;:ERRPC ARADD CHANNEL 3V LEV. OUTPUT  
DT2 ;:SERRPC ARBADD CHANL \$GDDAT \$BDDAT  
0

001336 013107  
001340 013654  
001342 015734  
001344 000000

; ITEM 3  
EM3 ;:VC STATUS REGISTER IN ERROR  
DH3 ;:ERRPC ARADD GOOD BAD  
DT3 ;:SERRPC ARBADD \$GDDAT \$BDDAT  
0

001346 013153  
001350 013723  
001352 015750  
001354 000000

; ITEM 4  
EM4 ;:EXTERNAL AD START FAILED  
DH3 ;:ERRPC ARADD GOOD BAD  
DT3 ;:SERRPC ARBADD \$GDDAT \$BDDAT  
0

001356 013207  
001360 013723  
001362 015750  
001364 000000

; ITEM 5  
EM5 ;:A/D DIFFERENTIAL LINEARITY ERROR  
DH5 ;:ERRPC ARADD # OF STATES ALLOWED  
DT5 ;:SERRPC ARBADD \$BDDAT \$GDDAT  
0

001366 013244  
001370 013760  
001372 015762  
001374 000000

; ITEM 6  
EM6 ;:NOISE LEVEL EXCEEDED LIMIT  
DH6 ;:ERRPC ARADD LIMIT MEASURED  
DT3 ;:SERRPC ARBADD \$GDDAT \$BDDAT  
0

001376 013305  
001400 014020  
001402 015750  
001404 000000

; ITEM 7  
EM7 ;:VC LOGIC SIGNAL LOW OUTPUT TOO HIGH

001406 013340

383	001410	014061	DH7	;ERRPC ARADD A/D CHAN +.4V LEV. OUTPUT
384	001412	015734	DT2	;SERRPC ARBADD CHANL \$GDDAT \$BDDAT
385	001414	000000	0	
386				
387			;ITEM 10	
388	001416	013404	EM10	;D/A DIFFERENTIAL LINEARITY ERROR
389	001420	014130	DH10	;ERRPC ARADD STEP NOMINAL SPRFAD ACTUAL
390	001422	015774	DT10	;SERRPC ARBADD EDGE \$GDDAT SPRLAD \$BDDAT
391	001424	000000	0	
392				
393			;ITEM 11	
394	001426	013445	EM11	;A/D INTER-CHANNEL SETTLING ERROR
395	001430	014207	DH11	;ERRPC ARADD NOMINAL SPREAD ACTUAL
396	001432	016012	DT11	;SERRPC ARBADD \$GDDAT SPREAD \$BDDAT
397	001434	000000	0	
398				
399			;ITEM 12	
400	001436	013506	EM12	;OFFSET ERROR
401	001440	014207	DH11	;ERRPC ARADD NOMINAL SPREAD ACTUAL
402	001442	016012	DT11	;SERRPC ARBADD \$GDDAT SPREAD \$BDDAT
403	001444	000000	0	
404				
405			;ITEM 13	
406	001446	013523	EM13	;CALIBRATION ERROR
407	001450	014207	DH11	;ERRPC ARADD NOMINAL SPREAD ACTUAL
408	001452	016012	DT11	;SERRPC ARBADD \$GDDAT SPREAD \$BDDAT
409	001454	000000	0	
410				
411			;ITEM 14	
412	001456	013545	EM14	;LINEARITY ERROR AT XX00
413	001460	014207	DH11	;ERRPC ARADD NOMINAL SPREAD ACTUAL
414	001462	016012	DT11	;SERRPC ARBADD \$GDDAT SPREAD \$BDDAT
415	001464	000000	0	
416				
417				
418			; ADDRESS OF KMC-11 OF LPA-11	THE ADDR FOR KMADO MAY BE
419			...	CHANGED BY THE USER TO REFLECT
420			...	A DIFFERENT KMC-11 ADDR. THE
421			...	REST OF THE ADDRESSES WILL
422			...	BE CHANGED BY THE PROGRAM.
423			...	
424			...	
425	001466		LPCI:	
426	001466	170460	KMADO: .WORD 170460	;BASE KMC ADDR. MAY BE PATCHED BY USER.
427				
428	001470		LPMR:	
429	001470	170461	KMAD1: .WORD 170460+1	; >DO NOT <;KMC-CSR ADDR
430	001472		LPCO:	
431	001472	170462	KMAD2: .WORD 170460+2	; >PATCH <;
432	001474		LPSO:	
433	001474	170463	KMAD3: .WORD 170460+3	; >THIS AREA <
434	001476		LPADL:	
435	001476	170464	KMAD4: .WORD 170460+4	;
436	001500		LADH:	

```

437 001500 170465 KMAD5: .WORD 170460+5 ;>DO NOT <
438 001502 LPMS1:
439 001502 170466 KMAD6: .WORD 170460+6 ;>PATCH <
440 001504 LPMS2:
441 001504 170467 KMAD7: .WORD 170460+7 ;>THIS AREA <
442
443 001506 000340 VECTOR: .WORD AVECT1&777 ;BASE VECTOR OF KMC
444 001510 000344 VECTPS: .WORD 4+AVECT1&777 ;VECTOR ADDR.+2
445
446 001512 000004 VERSN: .WORD 4 ;CURRENT VERSION NUMBER OF MICROCODE.
447
448 001514 000000 .DVLS: .WORD 0 ;/DEVICE LIST OF I/O ADDR. DEFINED
449 001516 000020 .BLKW 16. ;/BY INIT.
450
451 001556 170400 ARBADD: 170400
452 001560 000340 ARBVCT: 340
453 001562 000000 NMBEXT: 0
454 001564 000000 $TMDAT: 0
455 001566 000000 NBEXT: 0
456
457 001570 170400 ADCS: 170400 ;A TO D STATUS/CONTROL REGISTER
458 001572 170401 ADCS1: 170401 ;A TO D STATUS REGISTER <HIGH BYTE>
459 001574 170402 ADDR: 170402 ;A TO D CONVERTED VALUE <READ>
460 001576 170404 CSR: 170404 ;CLOCK STATUS REGISTER
461 001600 170406 CSB: 170406 ;CLOCK PRESET BUFFER
462 001602 170410 VCSTAT: 170410 ;DAC STATUS REGISTER
463 001604 170412 VCXREG: 170412 ;X BUFFER
464 001606 170414 VCYREG: 170414 ;Y BUFFER
465 001610 170416 CSC: 170416 ;CLOCK COUNTER
466 .SBTTL PROGRAM START-UP
467
468 001612 000240 BEGIN: NOP
469 001614 005037 001126 CLR $BDDAT ;CLEAR RO
470 001620 005037 020264 CLR WFTEST ;CLEAR TOLERANCE FLAG
471 001624 000406 RBEG
472 001626 012737 000001 020264 BEGIN2: MOV #1,WFTEST ;SET TOLERANCE FLAG
473 001634 012737 177777 001126 BEGIN1: MOV #-1,$BDDAT ;LOAD RO
474 001642 RBEG:
475
476 ;THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
477 ;
478
479 001642 010046 MOV RO,-(SP)
480 001644 010146 MOV R1,-(SP)
481 001646 013700 001466 MOV KMADO,RO ;GET KMC-11 ADDRESS.
482 001652 012701 001470 MOV #KMAD1,R1 ;GET ADDR. OF ADDR. LIST.
483
484 001656 005200 64$: INC RO ;UPDATE ADDR.
485 001660 010021 MOV RO,(1)+ ;WRITE ADDR.
486 001662 020127 001506 CMP R1,#KMAD7+2 ;DONE ALL ADDRESSES?
487 001666 001373 BNE 64$ ;NO - DO NEXT ADDR.
488 001670 005037 001514 CLR .DVLS ;CLR ADDR. LIST.
489 001674 012601 MOV (SP)+,R1
490 001676 012600 MOV (SP)+,RO
    
```

```

491 .SBTTL INITIALIZE THE COMMON TAGS
492 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
493 MOV $CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
494 CLR (R6)+ ;;CLEAR MEMORY LOCATION
495 CMP $SWR,R6 ;;DONE?
496 BNE -6 ;;LOOP BACK IF NO
497 MOV $STACK,SP ;;SETUP THE STACK POINTER
498 ;;INITIALIZE A FEW VECTORS
499 MOV $SCOPE,$IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
500 MOV $340,$IOTVEC+2 ;;LEVEL 7
501 MOV $ERROR,$EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
502 MOV $340,$EMTVEC+2 ;;LEVEL 7
503 MOV $TRAP,$TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
504 MOV $340,$TRAPVEC+2 ;;LEVEL 7
505 MOV $SPWRDN,$PWRVEC ;;POWER FAILURE VECTOR
506 MOV $340,$PWRVEC+2 ;;LEVEL 7
507 MOV $ENDCT,$SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
508 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
509 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
510 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
511 MOV $,$LPAOR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
512 MOV $,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
513 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
514 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
515 MOV $ERRVEC,-(SP) ;;SAVE ERROR VECTOR
516 MOV $65,$ERRVEC ;;SET UP ERROR VECTOR
517 MOV $DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
518 MOV $DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
519 CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
520 BNE 67$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
521 ;;AND THE HARDWARE SWR IS NOT = -1
522 BR 66$ ;;BRANCH IF NO TIMEOUT
523 MOV $65$,(SP) ;;SET UP FOR TRAP RETURN
524 RTI
525 MOV $SWREG,$SWR ;;POINT TO SOFTWARE SWR
526 MOV $DISPREG,$DISPLAY
527 MOV (SP)+,$ERRVEC ;;RESTORE ERROR VECTOR
528
529 CLR $PASS ;;CLEAR PASS COUNT
530 BITB $APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
531 BEQ 68$ ;;YES,USE NON-APT SWITCH
532 MOV $SSWREG,$SWR ;;NO,USE APT SWITCH REGISTER
533
534 .SBTTL TYPE PROGRAM NAME
535 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
536 INC #-1 ;;FIRST TIME?
537 BNE 69$ ;;BRANCH IF NO
538 CMP $SENDAD,$#42 ;;ACT-11?
539 BEQ 69$ ;;BRANCH IF YES
540 TYPE 70$ ;;TYPE ASCIZ STRING
541 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
542 TST $#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
543 BNE 71$ ;;BRANCH IF YES
544 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?

```

```

545 002206 001406          BEQ      71$          ;; BRANCH IF YES
546 002210 023727 001140 000176  CMP      SWP, #SWREG  ;; SOFTWARE SWITCH REG SELECTED?
547 002216 001005          BNE      72$          ;; BRANCH IF NO
548 002220 104406          GTSWR                    ;; GET SOFT-SWR SETTINGS
549 002222 000403          BR       72$
550 002224 112737 000001 001134 71$:   MOVB    #1, $AUTOB    ;; SET AUTO-MODE INDICATOR
551 002232          72$:
552 002232 000412          BR       69$          ;; GET OVER THE ASCIZ
553          ;; 70$: .ASCIZ <CRLF><15><12> #MD-11-DRLPE-A#<15><12><CRLF>
554 002260          69$:
555 002260 005037 001514          CLR     .DVLS
556 002264 005037 177776          CLR     PS
557 002270 012706 001100          MOV     #STACK, SP    ; LOAD STACK
558 002274 013702 001256          MOV     $BASE, R2     ; LOAD STARTING ADDRESS
559 002300 005003          CLR     R3           ; CLEAR COUNT
560 002302 010237 001564          2$:   MOV     R2, $TMDAT
561
562          ; *
563 002316 005737 024326          MOV     $TMDAT, $GDDAT ; /READ DEVICE REG $TMDAT, PUT DATA IN $GDDAT.
564 002322 001004          TST     $AERR
565 002324 062702 000020          BNE     1$
566 002330 005203          ADD     #20, R2      ; EXIST, UPDATE TEST ADDRESS
567 002332 000763          INC     R3          ; UPDATE # OF AR11'S
568 002334 005703          BR      2$
569 002336 001003          1$:   TST     R3          ; TEST IF FIRST DOE# EXIST
570 002340 000000          BNE     3$          ; BR
571 002342 000137 001612          HALT                    ; FIRST AR11 DOES NOT EXIST
572 002346 005303          JMP     BEGIN
573 002350 010337 001562          3$:   DEC     R3          ; ADJUST R3
574 002354 012737 000006 000004  MOV     R3, NMBEXT    ; SAVE THE NUMBER OF ADDITIONAL AR11'S
575 002362 005037 000006          MOV     #6, R#4      ; RESET BUS ERROR
576 002366 013737 001256 001556  MOV     $BASE, ARBADD ; LOAD FIRST ADDRESS
577 002374 013737 001252 001560  MOV     $VECT1, ARBVCT ; LOAD FIRST VECTOR
578 002402 013737 001562 001566  MOV     NMBEXT, NBEXT ; LOAD NUMBER OF ADDITIONAL AR11'S
579 002410 013700 001126          MOV     $BDDAT, R0
580 002414 004737 025300          RBEG2: JSR     PC, $RESET
581 002420 012702 000232          MOV     #232, R2     ; LOAD R2
582 002424 012701 000230          MOV     #230, R1     ; LOAD R1
583 002430 010221          5$:   MOV     R2, (R1)+    ; LOAD .+2
584 002432 005021          CLR     (R1)+        ; LOAD HALT
585 002434 010102          MOV     R1, R2      ; LOAD R2
586 002436 005722          TST     (R2)+        ; BUMP R2
587 002440 020227 001002          CMP     R2, #1002   ; TEST FOR LAST
588 002444 001371          BNE     5$          ; BR UNTIL DONE
589 002446 004737 020150          JSR     PC, WFADJ    ; ADJUST SOME TOLERANCES
590 002452 005700          TST     R0          ; TEST R0
591 002454 001402          BEQ     2$          ; BR IF CLEARED
592 002456 000137 002652          JMP     4$
593 002462 005737 000042          2$:   TST     R#42
594 002466 001402          BEQ     3$          ; TEST ACT-11 OR DDP
595 002470 000137 002652          JMP     4$          ; BR IF CLEARED
596 002474 013746 001562          3$:   MOV     NMBEXT, -(SP) ; INHIBIT TYPDUT
597 002500 104403          TYPOS   ; PUSH ON STACK
598 002502 000002          .WORD  2           ; TYPE OCTAL
    
```

599 002504 104401 002512  
600 002510 000420  
601  
602 002552  
603 002552 104401 002560  
604 002556 000435  
605  
606 002652  
607  
608 002652 012700 001570  
609 002656 013720 001556  
610 002662 022700 001612  
611 002666 001373  
612 002670 005237 001572  
613 002674 012700 001574  
614 002700 012701 000002  
615 002704 060120  
616 002706 062701 000002  
617 002712 022701 000020  
618 002716 001372

```

TYPE 65$          ;;TYPE ASCIZ STRING
BR 64$           ;;GET OVER THE ASCIZ
;;65$: .ASCIZ / (8) ADDITIONAL AR11'S CONNECTED/
64$:
TYPE 67$          ;;TYPE ASCIZ STRING
BR 66$           ;;GET OVER THE ASCIZ
;;67$: .ASCIZ <15,<12>/ALL AR-11'S MUST HAVE G5036 WRAPAROUND MODULE INSTALLED/
66$:
4$:  MOV #ADCS,RO          ;LOAD POINTER
10$: MOV ARBADD,(RO)+
     CMP #BEGIN,RO        ;TEST FOR END
     BNE 10$
     INC ADCS1
     MOV #ADDBR,RO
12$: MOV #2,R1
     ADD R1,(RO)+
     ADD #2,R1
     CMP #20,R1
     BNE 12$
     .SBTTL A TO D CHANNEL NUMBERS

```

;BIPOLAR CHANNEL NUMBERS

623	000000	CH0=	0
624	000001	CH1=	1
625	000002	CH2=	2
626	000003	CH3=	3
627	000004	CH4=	4
628	000005	CH5=	5
629	000006	CH6=	6
630	000007	CH7=	7
631	000010	CH10=	10
632	000011	CH11=	11
633	000012	CH12=	12
634	000013	CH13=	13
635	000014	CH14=	14
636	000015	CH15=	15
637	000016	CH16=	16
638	000017	CH17=	17

;UNIPOLAR CHANNEL NUMBERS

642	000040	CH40=	40
643	000041	CH41=	41
644	000042	CH42=	42
645	000043	CH43=	43
646	000044	CH44=	44
647	000045	CH45=	45
648	000046	CH46=	46
649	000047	CH47=	47
650	000050	CH50=	50
651	000051	CH51=	51
652	000052	CH52=	52

653 000053  
654 000054  
655 000055  
656 000056  
657 000057

CH53= 53  
CH54= 54  
CH55= 55  
CH56= 56  
CH57= 57

.SBTTL TEST NUMBER ABSTRACT  
.SBTTL  
.SBTTL

```

661 ;*****
662 ;*TEST 1 TEST THAT "ERASE" AND DONE CAN BE SET AND CLEARED
663 ;*****
664 †ST1: SCOPE
665 002720 000004 MOV #10,$TIMES ;;DO 10 ITERATIONS
666 002722 012737 000010 001166 MOV #0,$TMDAT ;CLEAR
667 002730 012737 000000 001564
668 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
669 002746 012737 010000 001124 MOV #BIT12,$GDDAT ;LOAD EXPECTED
670
671 ;* MOV $GDDAT,$VCSTAT ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCSTAT
672
673 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
674 002774 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
675 003002 001401 BEQ 1$ ;;BR IF SET
676 003004 104003 ERROR 3 ;ERASE FAILED TO SET, CHECK G5036-
;BCOBR CONNECTION: A-VV, VV-A
677 ;LOAD EXPECTED
678 003006 012737 001000 001124 1$: MOV #BIT9,$GDDAT
679
680 ;* MOV $VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
681 003024 052737 001000 001564 BIC #BIT9,$TMDAT
682
683 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
684
685 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
686 003052 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
687 003060 001401 BEQ 2$ ;;BR IF CLEARED
688 003062 104003 ERROR 3 ;ERASE BIT FAILED TO CLEAR BY ERASE
;RETURN (SETTING OF THE CH 2 BIT)
689
690 003064 012737 000200 001124 2$: MOV #BIT7,$GDDAT ;LOAD EXPECTED
691
692 ;* MOV $VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
693 003102 042737 001012 001564 BIC #BIT9:12,$TMDAT
694
695 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
696
697 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
698 003130 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
699 003136 001401 BEQ TST2 ;;BR IF EQUAL
700 003140 104003 ERROR 3 ;READY FAILED TO SET ON THE END
; OF ERASE RETURN (CLEARING CH 2)
701
702 ;*****
703 ;*TEST 2 EXTERNAL A TO D START FROM INTENSIFY
704 ;*****
705 †ST2: SCOPE
706 003142 000004

```



```

707 003144 012737 000010 001166      MOV    #10,$TIMES      ;;DO 10 ITERATIONS
708 003152 012737 000000 001564      MOV    #0,$TMDAT
709                                     ;*
710                                     MOV    $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
711 003170 000240      NOP
712 003172 000240      NOP
713                                     ;*
714 003204 012737 000220 001124      MOV    $TMDAT,@ADCS   ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
715                                     MOV    #BIT7:BIT4,$GDDAT ;LOAD EXPECTED
716                                     ;*
717                                     MOV    $GDDAT,@ADCS   ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
718                                     ;*
719                                     MOV    @VCSTAT,$TMDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
720 003232 005237 001564      INC    $TMDAT
721                                     ;*
722                                     MOV    $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
723 003246      1$:
724                                     ;*
725                                     MOV    @VCSTAT,$TMDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
726 003256 105737 001564      TSTB  $TMDAT
727 003262 100371      BPL   1$
728 003264 012737 000400 016342      MOV    #BIT8,TEMP     ;SET UP A COUNTER
729 003272 005337 016342      DEC   TEMP           ;DELAY
730 003276 100375      BPL   2$
731                                     ;*
732                                     MOV    @ADCS,$BDDAT   ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
733 003310 042737 000200 001124      BIC   #BIT7,$GDDAT
734 003316 042737 000200 001126      BIC   #BIT7,$BDDAT
735 003324 023737 001124 001126      CMP   $GDDAT,$BDDAT  ;COMPARE RESULTS
736 003332 001401      BEQ   3$             ;;BR IF EQUAL
737 003334 104004      ERROR 4             ;INTENSIFY PULSE FAILED TO START
738 003336 000240      3$:  NOP
739 003340 000240      NOP
740 003342 012737 000000 001564      MOV    #0,$TMDAT
741                                     ;*
742                                     MOV    $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
743                                     ;*
744                                     MOV    $TMDAT,@ADCS   ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
745                                     ;*****
746                                     ;*TEST 3 TEST THAT CH 0 IS A BIPOLAR GROUND
747                                     ;*****
748 003370 000004      ST3: SCOPE
749 003372 012737 000010 001166      MOV    #10,$TIMES    ;;DO 10 ITERATIONS
750 003400 012737 001000 001124      MOV    #1000,$GDDAT ;LOAD EXPECTED VALUE
751                                     ;*
752                                     MOV    $TMDAT,@ADCS   ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
753                                     ;*
754                                     MOV    $TMDAT,@ADDBR   ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADDBR
755 003426 004537 012610      JSR   R5,CONVRT      ;CONVERT
756 003432 000000      CHO   ;CH 0
757 003434 013737 013002 001126      MOV    ADEND,$BDDAT  ;LOAD VALUE READ
758 003442 012737 000001 013010      MOV    #1,SPREAD     ;LOAD + OR - COUNT SPREAD
759 003450 004737 013012      JSR   PC,COMPAR     ;COMPARE $GDDAT AND $BDDAT
760 003454 000401      BR    TS#4          ;;BR IF WITHIN TOLERANCE

```

761 003456 104001  
 762  
 763  
 764  
 765  
 766  
 767 003460 000004  
 768 003462 012737 000010 001166  
 769 003470 012737 000000 001124  
 770 003476 004537 012610  
 771 003502 000040  
 772 003504 013737 013002 001126  
 773 003512 012737 000001 013010  
 774 003520 004737 013012  
 775 003524 000401  
 776 003526 104001  
 777  
 778  
 779  
 780  
 781  
 782 003530 000004  
 783 003532 012737 000010 001166  
 784 003540 012737 001315 001124  
 785 003546 004537 012610  
 786 003552 000002  
 787 003554 013737 013002 001126  
 788 003562 012737 000024 013010  
 789 003570 004737 013012  
 790 003574 000401  
 791 003576 104001  
 792  
 793  
 794  
 795  
 796  
 797 003600 000004  
 798 003602 012737 000010 001166  
 799 003610 012737 000315 001124  
 800 003616 004537 012610  
 801 003622 000042  
 802 003624 013737 013002 001126  
 803 003632 012737 000024 013010  
 804 003640 004737 013012  
 805 003644 000401  
 806 003646 104001  
 807  
 808  
 809  
 810  
 811  
 812 003650 000004  
 813 003652 012737 000010 001166  
 814 003660 013737 020266 001124

```

ERROR 1 ;CH 0 FAILED TO EQUAL EXPECTED
; VALUE
;*****
; *TEST 4 TEST THAT CH 40 IS A UNIPOLAR GROUND
;*****
†ST4: SCOPE
MOV #10,STIMES ;;DO 10 ITERATIONS
MOV #0,$GDDAT ;LOAD EXPECTED VALUE
JSR RS,CONVRT ;CONVERT
CH40 ;CH 40
MOV ADEND,$BDDAT ;LOAD VALUE READ
MOV #1,SPREAD ;LOAD + OR - COUNT SPREAD
JSR PC,COMPAR ;COMPARE $GDDAT AND $BDDAT
BR TST5 ;;BR IF WITHIN TOLERANCE
ERROR 1 ;CH 40 FAILED TO EQUAL EXPECTED
; VALUE
;*****
; *TEST 5 TEST THAT CH 2 IS AT +1 VOLT BIPOLAR
;*****
†ST5: SCOPE
MOV #10,STIMES ;;DO 10 ITERATIONS
MOV #1315,$GDDAT ;LOAD EXPECTED VALUE
JSR RS,CONVRT ;CONVERT
CH2 ;CH 2
MOV ADEND,$BDDAT ;LOAD VALUE READ
MOV #24,SPREAD ;LOAD + OR - COUNT SPREAD
JSR PC,COMPAR ;COMPARE $GDDAT AND $BDDAT
BR TST6 ;;BR IF WITHIN TOLERANCE
ERROR 1 ;CH 2 FAILED TO EQUAL EXPECTED
; VALUE
;*****
; *TEST 6 TEST THAT CH 42 IS AT +1 VOLT UNIPOLAR
;*****
†ST6: SCOPE
MOV #10,STIMES ;;DO 10 ITERATIONS
MOV #315,$GDDAT ;LOAD EXPECTED VALUE
JSR RS,CONVRT ;CONVERT
CH42 ;CH 42
MOV ADEND,$BDDAT ;LOAD VALUE READ
MOV #24,SPREAD ;LOAD + OR - COUNT SPREAD
JSR PC,COMPAR ;COMPARE $GDDAT AND $BDDAT
BR TST7 ;;BR IF WITHIN TOLERANCE
ERROR 1 ;CH 42 FAILED TO EQUAL EXPECTED
; VALUE
;*****
; *TEST 7 TEST THAT CH 3 IS AT +2.5 BIPOLAR
;*****
†ST7: SCOPE
MOV #10,STIMES ;;DO 10 ITERATIONS
MOV V1754,$GDDAT ;LOAD EXPECTED VALUE

```

```

815 003666 004537 012610 JSR RS,CONVRT ; CONVERT
816 003672 000003 CH3 ; CH 3
817 003674 013737 013002 001126 MOV ADEND,$BDDAT ; LOAD VALUE READ
818 003702 013737 020270 013010 MOV V24,SPREAD ; LOAD + OR - COUNT SPREAD
819 003710 004737 013012 JSR PC,COMPAR ; COMPARE $GDDAT AND $BDDAT
820 003714 000401 BR TST10 ;;BR IF WITHIN TOLERANCE
821 003716 104001 ERROR 1 ;CH 3 FAILED TO EQUAL EXPECTED
; VALUE
822
823
824
825 ;*****
826 ;*TEST 10 TEST THAT CH 43 IS AT +2.5 UNIPOLAR
827 003720 000004 TST10: SCOPE
828 003722 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
829 003730 012737 001000 001124 MOV #1000,$GDDAT ;LOAD EXPECTED VALUE
830 003736 004537 012610 JSR RS,CONVRT ;CONVERT
831 003742 000043 CH43 ;CH 43
832 003744 013737 013002 001126 MOV ADEND,$BDDAT ;LOAD VALUE READ
833 003752 013737 020272 013010 MOV V50,SPREAD ;LOAD + OR - COUNT SPREAD
834 003760 004737 013012 JSR PC,COMPAR ;COMPARE $GDDAT AND $BDDAT
835 003764 000401 BR TST11 ;;BR IF WITHIN TOLERANCE
836 003766 104001 ERROR 1 ;CH 43 FAILED TO EQUAL EXPECTED
; VALUE
837
838
839 ;*****
840 ;*TEST 11 TEST THAT CH 4 IS AT -2.5 BIPOLAR
841 003770 000004 TST11: SCOPE
842 003772 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
843 004000 013737 020270 001124 MOV V24,$GDDAT ;LOAD EXPECTED VALUE
844 004006 004537 012610 JSR RS,CONVRT ;CONVERT
845 004012 000004 CH4 ;CH 4
846 004014 013737 013002 001126 MOV ADEND,$BDDAT ;LOAD VALUE READ
847 004022 013737 020270 013010 MOV V24,SPREAD ;LOAD + OR - COUNT SPREAD
848 004030 004737 013012 JSR PC,COMPAR ;COMPARE $GDDAT AND $BDDAT
849 004034 000401 BR TST12 ;;BR IF WITHIN TOLERANCE
850 004036 104001 ERROR 1 ;CH 4 FAILED TO EQUAL EXPECTED VALUE
851
852
853 ;*****
854 ;*TEST 12 TEST THAT CH 57 IS AT +4 VOLTS UNIPOLAR
855 004040 000004 TST12: SCOPE
856 004042 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
857 004050 012737 001463 001124 MOV #1463,$GDDAT ;LOAD EXPECTED VALUE
858 004056 004537 012610 JSR RS,CONVRT ;CONVERT
859 004062 000057 CH57 ;CH 57
860 004064 013737 013002 001126 MOV ADEND,$BDDAT ;LOAD VALUE READ
861 004072 012737 000120 013010 MOV #120,SPREAD ;LOAD + OR - COUNT SPREAD
862 004100 004737 013012 JSR PC,COMPAR ;COMPARE $GDDAT AND $BDDAT
863 004104 000401 BR TST13 ;;BR IF WITHIN TOLERANCE
864 004106 104001 ERROR 1 ;CH 57 FAILED TO EQUAL EXPECTED VALUE
865
866
867 ;*****
868 ;*TEST 13 TEST THAT CH 41 IS LESS THAN +200 MVOLTS UNIPOLAR

```

B03

MAINDEC-11-DRLPE-A  
DRLPE.P11 T13

MACY11 27(654) 14-DEC-77 20:24 PAGE 19  
TEST THAT CH 41 IS LESS THAN +200 MVOLTS UNIPOLAR

SEQ 0028

```

869
870 004110 000004
871 004112 012737 000004 001166
872 004120 012737 000041 013006
873 004126 112737 000041 001564
874 004134 000337 001564
875
876
877 004150 013737 020274 001124
878 004156 013737 020274 013010
879 004164 012737 000200 016342
880 004172 005337 016342
881 004176 001375
882
883
884 004210 005237 001564
885
886
887 004224
888
889
890 004234 105737 001564
891 004240 100371
892
893
894 004252 013737 001126 016310
895 004260 004737 013012
896 004264 000401
897 004266 104001
898
899 004270 032777 010000 174642
900 004276 001432
901 004300 104401 004306
902 004304 000412
903
904 004332
905 004332 013746 001556
906 004336 104402
907 004340 104401 020040
908 004344 104401 014256
909 004350 013746 016310
910 004354 104403
911 004356 003 000
912 004360 104401 014272
913
914
915
916
917 004364 000004
918 004366 012737 000010 001166
919 004374 012737 005000 001564
920
921
922 004412 012737 001146 001124

```

```

*****
↑ST13: SCOPE
MOV #4,STIMES ;;DO 4 ITERATIONS
MOV #CH41,CHANL ;LOAD CHANNEL # FOR ERROR TYP0UT
MOVB #CH41,$TMDAT ;LOAD MUX
SWAB $TMDAT
;* MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
MOV VA24,$GDDAT ;LOAD EXPECTED VALUE
MOV VA24,$SPREAD ;LOAD + OR - COUNT SPREAD
MOV #BIT7,TEMP ;LOAD DELAY
DEC TEMP ;DELAY
BNE 2$
2$:
;* MOV $ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
INC $TMDAT
;* MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
3$:
;* MOV $ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
TSTB $TMDAT
BPL 3$
;* MOV $ADDBR,$BDDAT ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
MOV $BDDAT,$BIASC1 ;SAVE RESULTS
JSR PC,COMPAR ;COMPARE $GDDAT AND $BDDAT
BR 1$ ;;BR IF WITHIN TOLERANCE
ERROR 1 ;CH 41 FAILED TO EQUAL EXPECTED
;VALUE, BIAS CURRENT TOO HIGH
1$: BIT #BIT12,$SWR ;TEST FORCED SW
BEQ TST14 ;;BR IF NOT FORCED PRINTOUT
TYPE 65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
65$: .ASCIZ <15><12><12>/AR-11 ADDRESS = /
64$: MOV ARBADD,-(SP) ;SAVE ADDRESS
TYPC
TYPE ,ACRLF
TYPE ,BIASST
MOV BIASC1,-(SP)
TYPOS
.BYTE 3,0
TYPE ,BIASDN
*****
;*TEST 14 TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH
*****
↑ST14: SCOPE
MOV #10,STIMES ;;DO 10 ITERATIONS
MOV #BIT11:BIT9,$TMDAT ;LOAD VC STATUS
;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
MOV #1146,$GDDAT ;LOAD EXPECTED VALUE

```

C03

MAINDEC-11-DRLPE-A  
DRLPE.P11 T14

MACY11 27(654) 14-DEC-77 20:24 PAGE 20  
TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH

SEQ 0029

```

923 004420 004537 012610 JSR RS,CONVRT ;CONVERT
924 004424 000053 CH53 ;CH 53
925 004426 013737 013002 001126 MOV ADEND,$BDDAT ;LOAD VALUE READ
926 004434 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
927 004442 003401 BLE TST15 ;;BR IF LESS THAN OR EQUAL
928 004444 104002 ERROR 2 ;CH 53 FAILED TO EQUAL EXPECTED VALUE
929
930 ;*****
931 ;*TEST 15 TEST THAT ERASE L (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
932 ;*****
933 004446 000004 †TST15: SCOPE
934 004450 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
935 004456 012737 000000 001564 MOV #0,$TMDAT ;CLEAR VC STATUS
936
937 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
938 004474 012737 012000 001564 MOV #BIT12!BIT10,$TMDAT ;SET ERASE
939
940 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
941 004512 012737 000120 001124 MOV #120,$GDDAT ;LOAD EXPECTED VALUE
942 004520 004537 012610 JSR RS,CONVRT ;CONVERT
943 004524 000053 CH53 ;CH 53
944 004526 013737 013002 001126 MOV ADEND,$BDDAT ;LOAD VALUE READ
945 004534 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
946 004542 002001 BGE TST16 ;;BR IF GREATER THAN OR EQUAL
947 004544 104007 ERROR 7 ;CH 53 FAILED TO EQUAL EXPECTED VALUE
948
949 ;*****
950 ;*TEST 16 TEST THAT WRITE-THRU (CH 54) IS GREATER THAN +3 VOLTS WHEN HIGH
951 ;*****
952 004546 000004 †TST16: SCOPE
953 004550 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
954 004556 012737 011000 001564 MOV #BIT12!BIT9,$TMDAT ;LOAD VC STATUS
955
956 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
957 004574 012737 001146 001124 MOV #1146,$GDDAT ;LOAD EXPECTED VALUE
958 004602 004537 012610 JSR RS,CONVRT ;CONVERT
959 004606 000054 CH54 ;CH 54
960 004610 013737 013002 001126 MOV ADEND,$BDDAT ;LOAD VALUE READ
961 004616 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
962 004624 003401 BLE TST17 ;;BR IF LESS THAN OR EQUAL
963 004626 104002 ERROR 2 ;CH 54 FAILED TO EQUAL EXPECTED VALUE
964
965 ;*****
966 ;*TEST 17 TEST THAT WRITE-THRU (CH 54) IS LESS THAN +400 MVOLTS WHEN LOW
967 ;*****
968 004630 000004 †TST17: SCOPE
969 004632 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
970 004640 012737 006000 001564 MOV #BIT11!BIT10,$TMDAT ;SET WRITE-THRU
971
972 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
973 004656 012737 000120 001124 MOV #120,$GDDAT ;LOAD EXPECTED VALUE
974 004664 004537 012610 JSR RS,CONVRT ;CONVERT
975 004670 000054 CH54 ;CH 54
976 004672 013737 013002 001126 MOV ADEND,$BDDAT ;LOAD VALUE READ

```



E03

MAINDEC-11-DRLPE-A  
DRLPE.P11 T22

MACY11 27(654) 14-DEC-77 20:24 PAGE 22  
TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH

SEQ 0031

```

1031                                     ; VALUE
1032
1033                                     ;*****
1034                                     ;*TEST 23      TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW
1035                                     ;*****
1036 005140 000004 †ST23: SCOPE
1037 005142 012737 000010 001166      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
1038 005150 012737 003000 001564      MOV      #BIT9:BIT10,$TMDAT ;SET CHANNEL 2
1039
1040                                     ;*      MOV      $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1041 005166 012737 000120 001124      MOV      #120,$GDDAT      ;LOAD EXPECTED VALUE
1042 005174 004537 012610              JSR      RS,CONVRT        ;CONVERT
1043 005200 000056              CH56          ;CH 56
1044 005202 013737 013002 001126      MOV      ADEND,$BDDAT      ;LOAD VALUE READ
1045 005210 023737 001124 001126      CMP      $GDDAT,$BDDAT     ;COMPARE
1046 005216 002001              BGE      1$              ;;BR IF GREATER THAN
1047 005220 104007              ERROR      7              ;CH 56 FAILED TO EQUAL EXPECTED
1048                                     ; VALUE
1049 005222 012737 000000 001564 1$:  MOV      #0,$TMDAT        ;CLEAR BIT 9
1050
1051                                     ;*      MOV      $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1052
1053                                     ;*****
1054                                     ;*TEST 24      A TO D DIFFERENTIAL LINEARITY TEST
1055                                     ;*****
1056 005240 000004 †ST24: SCOPE
1057 005242 012737 000001 001166      MOV      #1,$TIMES        ;;DO 1 ITERATION
1058
1059 005250 004537 016344              JSR      RS,DIFLIN        ;START TEST
1060 005254 001604              VCXREG        ;X COARSE
1061 005256 001606              VCYREG        ;Y FINE
1062 005260          001          005          .BYTE      1,5          ;GO AND CHANNEL 5
1063
1064 005262 005037 001124              CLR      $GDDAT          ;CLEAR EXPECTED
1065 005266 013737 017170 001126      MOV      SKIPST,$BDDAT    ;LOAD # OF SKIPPED STATES
1066 005274 001401              BEQ      1$              ;;BR IF NO SKIPPED STATES
1067 005276 104005              ERROR      5              ;SKIPPED STATE(S) DETECTED
1068
1069 005300 013737 017174 001126 1$:  MOV      EXCESS,$BDDAT    ;LOAD # OF >2LSB STATES
1070 005306 001401              BEQ      2$              ;;BR IF NO GREATER THAN 2LSB WIDE STATES
1071 005310 104005              ERROR      5              ;> 2LSB WIDE STATE(S) DETECTED
1072
1073 005312 013737 017172 001126 2$:  MOV      DIFERR,$BDDAT    ;LOAD # OUTSIDE +- 5/2 LSB
1074 005320 012737 000064 001124      MOV      #52,$GDDAT      ;LOAD MAX # OF ALLOWABLE STATES OUTSIDE +- 1/2 L
1075 005326 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;ANY ERRORS
1076 005334 002001              BGE      TST25          ;;BR IF NO ERROR
1077 005336 104005              ERROR      5              ;DIFFERENTIAL LINEARITY ERROR, >5% OF STATE
1078                                     ;WIDTHS OUTSIDE +- 1/2 LSB
1079
1080                                     ;*****
1081                                     ;*TEST 25      X D/A DIFFERENTIAL LINEARITY USING CH 5
1082                                     ;*****
1083 005340 000004 †ST25: SCOPE
1084 005342 012737 000010 001166      MOV      #10,$TIMES      ;;DO 10 ITERATIONS

```

```

1085 005350 012701 000001      MOV      #1,R1          ;LOAD EDGE VALUE
1086 005354 012702 016146      MOV      #RSLT2,R2     ;LOAD RESULT POINTER
1087
1088 005360 010137 001564      1$:     MOV      R1,$TMDAT ;LOAD X DAC WITH PRESET
1089
1090      ;*     MOV      $TMDAT,$VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1091 005374 010137 005410      MOV      R1,$IOS      ;LOAD EDGE VALUE
1092
1093 005400 004437 011716      JSR      R4,SAR        ;SOFTWARE SUCCESSIVE APPROX. ROUTINE
1094 005404 001606
1095 005406      000      005      VCYREG
1096 005410 000000      10$:    0           ;EDGE VALUE
1097 005412 000200      BIT7      ;50-50
1098
1099 005414 013737 016246 005552      MOV      DACSAV,12$    ;SAVE VALUE
1100
1101      ;*     MOV      $VCXREG,$TMDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $TMDAT.
1102 005432 005337 001564      DEC      $TMDAT
1103
1104      ;*     MOV      $TMDAT,$VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1105 005446 010137 005462      MOV      R1,11$      ;LOAD SAME EDGE
1106
1107 005452 004437 011716      JSR      R4,SAR        ;SOFTWARE S.A.R.
1108 005456 001606
1109 005460      000      005      VCYREG
1110 005462 000000      11$:    0           ;EDGE VALUE
1111 005464 000200      BIT7
1112
1113 005466 013737 016246 005554      MOV      DACSAV,13$    ;SAVE VALUE
1114 005474 013737 016246 001126      MOV      DACSAV,$BDDAT ;LOAD VALUE INTO $BDDAT
1115 005502 163737 005552 001126      SUB      12$, $BDDAT   ;STEP HEIGHT IN $BDDAT
1116 005510 013722 001126      MOV      $BDDAT,(R2)+ ;SAVE RESULT
1117 005514 012737 000062 001124      MOV      #62,$GDDAT   ;LOAD EXPECTED VALUE FOR 1 LSB STEP
1118 005522 013737 020276 013010      MOV      V62,$SPREAD  ;LOAD TOLERANCE
1119 005530 004737 013012      JSR      PC,COMPAR    ;TEST LIMITS
1120 005534 000401      BR      2$           ;;BR IF WITHIN LIMITS
1121 005536 104010      ERROR   10          ;D/A DIFFERENTIAL LINEARITY ERROR
1122 005540 006301      2$:     ASL      R1      ;MOVE LEFT
1123 005542 020127 002000      CMP      R1,#2000    ;DONE
1124 005546 001304      BNE     1$          ;;BR OF NOT DONE
1125 005550 000402      BR      TST26
1126
1127 005552 000000      12$:    0
1128 005554 000000      13$:    0
1129
1130      ;*****
1131      ;*TEST 26 Y D/A DIFFERENTIAL LINEARITY USING CH 6
1132      ;*****
1132 005556 000004      †ST26:  SCOPE
1133 005560 012737 000010 001166      MOV      #10,$TIMES   ;;DO 10 ITERATIONS
1134 005566 012701 000001      MOV      #1,R1        ;LOAD EDGE VALUE
1135 005572 012702 016204      MOV      #RSLT3,R2   ;LOAD RESULT POINTER
1136
1137 005576 010137 001564      1$:     MOV      R1,$TMDAT ;LOAD Y DAC WITH PRESET
1138

```



```

1139          ;*      MOV      $TMDAT, @VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1140 005612 010137 005626          MOV      R1, 10$          ;LOAD EDGE VALUE
1141
1142 005616 004437 011716          JSR      R4, SAR          ;SOFTWARE S.A.R.
1143 005622 001604          VCXREG
1144 005624          000          .BYTE 0, CH6
1145 005626 000000          10$: 0          ;EDGE VALUE
1146 005630 000200          BIT7
1147
1148 005632 013737 016246 006034          MOV      DACSAV, 12$          ;SAVE VALUE
1149
1150          ;*      MOV      @VCYREG, $TMDAT  ;/READ DEVICE REG VCYREG, PUT DATA IN $TMDAT.
1151 005650 005337 001564          DEC      $TMDAT
1152
1153          ;*      MOV      $TMDAT, @VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1154 005664 010137 005700          MOV      R1, 11$          ;LOAD EDGE
1155
1156 005670 004437 011716          JSR      R4, SAR          ;SOFTWARE S.A.R.
1157 005674 001604          VCXREG
1158 005676          000          .BYTE 0, CH6
1159 005700 000000          11$: 0
1160 005702 000200          BIT7
1161
1162 005704 013737 016246 006036          MOV      DACSAV, 13$          ;SAVE RESULTS
1163 005712 013737 016246 001126          MOV      DACSAV, $BDDAT      ;LOAD $BDDAT
1164 005720 163737 006034 001126          SUB      12$, $BDDAT          ;STEP HEIGHT IN $BDDAT
1165 005726 013722 001126          MOV      $BDDAT, (R2)+       ;SAVE RESULT
1166 005732 012737 000062 001124          MOV      #62, $GDDAT         ;LOAD EXPECTED
1167 005740 013737 020276 013010          MOV      V62, SPREAD        ;LOAD TOLERANCE
1168 005746 004737 013012          JSR      PC, COMPAR          ;TEST IF WITHIN LIMITS
1169 005752 000401          BR      2$                  ;;BR IF WITHIN LIMITS
1170 005754 104010          ERROR 10                    ;D/A DIFFERENTIAL LINEARITY ERROR
1171 005756 006301          2$: ASL      R1
1172 005760 020127 002000          CMP      R1, #2000          ;DONE ?
1173 005764 001304          BNE      1$                  ;;BR IF NOT DONE
1174 005766 032777 010000 173144          BIT      #BIT12, @SWR        ;TEST FORCED PRINTOUT
1175 005774 001421          BEQ      TST27              ;;BR IF NOT FORCED PRINTOUT
1176 005776 104401 014327          TYPE    DIFMSG
1177 006002 004537 012564          JSR      R5, TYPSTG          ;TYPE A STRING OF OCTAL #
1178 006006 01614E          RSLT2          ;STARTING
1179 006010 000012          IO.          ; # OF LOCATIONS
1180 006012 104401 014513          TYPE    DIYMSG
1181 006016 004537 012564          JSR      R5, TYPSTG          ;TYPE A STRING OF OCTAL #
1182 006022 016204          RSLT3          ; # OF LOCATIONS
1183 006024 000012          IO.
1184 006026 104401 014523          TYPE    DIEMSG
1185 006032 000402          BR      †TST27              ;;
1186 006034 000000          12$: 0
1187 006036 000000          13$: 0
1188          ;*****
1189          ;*TEST 27 A TO D POSITIVE MULTIPLEXER SETTLING TEST
1190          ;*****
1191 006040 000004          †TST27: SCOPE
1192 006042 012737 000010 001166          MOV      #10, $TIMES        ;;DO 10 ITERATIONS

```

```

1193 006050 012737 001770 001564      MOV      #1770,$TMDAT      ;LOAD Y DAC
1194
1195      ;*      MOV      $TMDAT,@VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1196 006066 004437 011716      JSR      R4,SAR          ;SOFTWARE S.A.R.
1197 006072 001604      VCXREG
1198 006074      000      006      .BYTE 0,6
1199 006076 001771      1771
1200 006100 000200      BIT7
1201
1202 006102 013737 016246 006174      MOV      DACSAV,10$      ;SAVE RESULTS
1203 006110 004437 011716      JSR      R4,SAR          ;SOFTWARE S.A.R.
1204 006114 001604      VCXREG
1205 006116      004      006      .BYTE CH4,CH6          ;-2.5V CH AND CH 6
1206 006120 001771      1771
1207 006122 000200      BIT7          ;EDGE
1208
1209 006124 013737 016246 001126      MOV      DACSAV,$BDDAT   ;SAVE RESULT IN $BDDAT
1210 006132 163737 006174 001126      SUB      10$,$BDDAT      ;SUB FIRST VALUE
1211 006140 013737 001126 016304      MOV      $BDDAT,ADPMUX   ;SAVE RESULT
1212 006146 012737 000000 001124      MOV      #0,$GDDAT      ;LOAD EXPECTED SETTling ERROR
1213 006154 012737 000077 013010      MOV      #77,SPREAD     ;LOAD TOLERANCE
1214 006162 004737 013012      JSR      PC,COMPAR      ;TEST IF WITHIN LIMITS
1215 006166 000403      BR      TST30           ;;BR IF WITHIN
1216 006170 104011      ERROR  11              ;A/D POSITIVE SETTling ERROR
1217 006172 000401      BR      TST30           ;;
1218 006174 000000      10$: 0
1219
1220      ;*****
1221      ;*TEST 30 A TO D NEGATIVE MULTIPLEXER SETTling TEST
1222      ;*****
1223 006176 000004      †TST30: SCOPE
1224 006200 012737 000010 001166      MOV      #10,$TIMES     ;;DO 10 ITERATIONS
1225 006206 012737 000010 001564      MOV      #10,$TMDAT     ;LOAD Y DAC
1226
1227      ;*      MOV      $TMDAT,@VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1228 006224 004437 011716      JSR      R4,SAR          ;SOFTWARE S.A.R.
1229 006230 001604      VCXREG
1230 006232      000      006      .BYTE 0,6
1231 006234 000010      10
1232 006236 000200      BIT7
1233
1234 006240 013737 016246 006376      MOV      DACSAV,10$      ;SAVE RESULTS
1235 006246 004437 011716      JSR      R4,SAR          ;SOFTWARE S.A.R.
1236 006252 001604      VCXREG
1237 006254      003      006      .BYTE CH3,CH6          ;+2.5V CH AND CH 6
1238 006256 000010      10
1239 006260 000200      BIT7          ;EDGE
1240
1241 006262 013737 016246 001126      MOV      DACSAV,$BDDAT   ;SAVE RESULT IN $BDDAT
1242 006270 163737 006376 001126      SUB      10$,$BDDAT      ;SUB FIRST VALUE
1243 006276 013737 001126 016306      MOV      $BDDAT,ADNMUX   ;SAVE RESULTS
1244 006304 012737 000000 001124      MOV      #0,$GDDAT      ;LOAD EXPECTED ZERO SETTling ERROR
1245 006312 012737 000077 013010      MOV      #77,SPREAD     ;LOAD TOLERANCE
1246 006320 004737 013012      JSR      PC,COMPAR      ;TEST IF WITHIN LIMITS

```

```

1247 006324 000401 BR 1$ ;:BR IF WITHIN
1248 006326 104011 ERROR 11 ;A/D NEGATIVE SETTLING ERROR
1249 006330 032777 010000 172602 1$: BIT #BIT12,2SWR ;TEST FOR FORCED TYPEOUT
1250 006336 001420 BEQ TST31 ;:BR IF NOT FORCED
1251 006340 104401 014562 TYPE ADMSG
1252 006344 013746 016304 MOV ADPMUX,-(SP) ;SAVE MUX SETTLING
1253 006350 104403 TYPOS
1254 006352 003 000 .BYTE 3,0
1255 006354 104401 014651 TYPE ADEMSG
1256 006360 013746 016306 MOV ADNMUX,-(SP) ;SAVE MUX SETTLING
1257 006364 104403 TYPOS
1258 006366 003 000 .BYTE 3,0
1259 006370 104401 020040 TYPE ACRLF
1260 006374 000401 BR †TST31 ;:
1261
1262 006376 000000 10$: 0
1263
1264 ;:*****
1265 ;*TEST 31 A/D RMS NOISE TEST ON CHANNEL 7 - BIPOLAR
1266 ;:*****
1267 006400 000004 †TST31: SCOPE
1268 006402 012737 000010 001166 MOV #10,$TIMES ;:DO 10 ITERATIONS
1269
1270 006410 004437 011716 JSR R4,SAR ;S A R ROUTINE
1271 006414 001606 VCYREG ;Y AXIS
1272 006416 000 007 .BYTE 0,7 ;CHANNEL 7, FINE Y
1273 006420 001000 1000 ;EDGE VALUE
1274 006422 000000 0 ;RMS RESULTS
1275
1276 006424 010537 001126 MOV R5,$BDDAT ;SAVE RESULTS (2X RMS NOISE, 1=1/50 LSB)
1277 006430 010537 016254 MOV R5,CH7RMS ;SAVE RESULTS (RMS NOISE, 1 = 1/100 LSB)
1278 006434 013737 016250 001124 MOV RMSMAX,$GDDAT ;LOAD EXPECTED
1279 006442 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULTS
1280 006450 002001 BGE TST32 ;:BR IF NOISE WITHIN SPEC
1281 006452 104006 ERROR 6 ;A/D RMS NOISE (BIPOLAR) IS GREATER
1282 ; THAN SPEC
1283
1284 ;:*****
1285 ;*TEST 32 A/D PEAK NOISE TEST ON CHANNEL 7 - BIPOLAR
1286 ;:*****
1287 006454 000004 †TST32: SCOPE
1288 006456 012737 000010 001166 MOV #10,$TIMES ;:DO 10 ITERATIONS
1289
1290 006464 004437 011716 JSR R4,SAR ;S A R ROUTINE
1291 006470 001606 VCYREG ;Y AXIS
1292 006472 000 007 .BYTE 0,7 ;CHANNEL 7, FINE Y
1293 006474 001000 1000 ;EDGE VALUE
1294 006476 100000 BIT15 ;PEAK RESULTS
1295
1296 006500 010537 001126 MOV R5,$BDDAT ;SAVE RESULTS (PEAK NOISE, 1 = .01 LSB)
1297 006504 010537 016256 MOV R5,CH7PEK ;SAVE RESULTS
1298 006510 013737 016252 001124 MOV PEKMAX,$GDDAT ;LOAD EXPECTED
1299 006516 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULTS
1300 006524 002001 BGE TST33 ;:BR IF NOISE WITHIN SPEC

```

```

1301 006526 104006          ERROR 6          ;A/D PEAK NOISE (BIPOLAR) IS GREATER
1302                                     ; THAN SPEC
1303
1304
1305
1306
1307 006530 000004          ;*****
1308 006532 012737 000010 001166 ;*TEST 33      A/D RMS NOISE TEST ON CHANNEL 47 - UNIPOLAR
1309                                     ;*****
1310 JSR      R4,SAR          ;S A R ROUTINE
1311 VCYREG          ;Y AXIS
1312 .BYTE 0,CH47          ;CHANNEL 47, FINE Y
1313 3              ;EDGE VALUE
1314 0              ;RMS RESULTS
1315
1316 006554 010537 001126          ;SAVE RESULTS (RMS NOISE, 1=.01 LSB)
1317 006560 010537 016260          ;SAVE RESULT
1318 006564 013737 016250 001124 ;LOAD EXPECTED
1319 006572 023737 001124 001126 ;COMPARE RESULTS
1320 006600 002001          ;BR IF NOISE WITHIN SPEC
1321 006602 104006          ;A/D RMS NOISE (UNIPOLAR) IS GREATER
1322                                     ; THAN SPEC
1323
1324
1325
1326
1327 006604 000004          ;*****
1328 006606 012737 000010 001166 ;*TEST 34      A/D PEAK NOISE TEST ON CHANNEL 47 - UNIPOLAR
1329                                     ;*****
1330 JSR      R4,SAR          ;S A R ROUTINE
1331 VCYREG          ;Y AXIS
1332 .BYTE 0,CH47          ;CHANNEL 47, FINE Y
1333 3              ;EDGE VALUE
1334 006626 100000          ;PEAK RESULTS
1335
1336 006630 010537 001126          ;SAVE RESULTS (PEAK NOISE, 1 = .01 LSB)
1337 006634 010537 016262          ;SAVE RESULT
1338 006640 013737 016252 001124 ;LOAD EXPECTED
1339 006646 023737 001124 001126 ;COMPARE RESULTS
1340 006654 002001          ;BR IF NOISE WITHIN SPEC
1341 006656 104006          ;A/D PEAK NOISE (UNIPOLAR) IS GREATER
1342                                     ; THAN SPEC
1343
1344
1345
1346
1347 006660 000004          ;*****
1348 006662 012737 000010 001166 ;*TEST 35      X DAC RMS NOISE TEST ON CHANNEL 5
1349 006670 012737 001000 001564 ;*****
1350
1351 ;*      MOV      $TMDAT,$VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1352
1353 006706 004437 011716          ;S A R ROUTINE
1354 006712 001606          ;Y AXIS

```

```

1355 006714 000 005 .BYTE 0,5 ;CHANNFL 5, COARSE X AND FINE Y
1356 006716 001000 1000 ;EDGE VALUE
1357 006720 000000 0 ;RMS RESULTS
1358
1359 006722 010537 001126 MOV R5,$BDDAT ;SAVE RESULTS (X DAC + A/D RMS NOISE, 1=.01 LSB)
1360 006726 163737 016254 001126 SUB CH7RMS,$BDDAT ;X DAC RMS NOISE ONLY
1361 006734 013737 001126 016264 MOV $BDDAT,XDACRM ;SAVE RESULT
1362 006742 013737 016250 001124 MOV RMSMAX,$GDDAT ;LOAD EXPECTED
1363 006750 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULTS
1364 006756 002001 BGE TST36 ;;BR IF NOISE WITHIN SPEC
1365 006760 104006 ERROR 6 ;X D/A RMS NOISE IS GREATER
1366 ; THAN SPEC
1367
1368 ;*****
1369 ;*TEST 36 X DAC PEAK NOISE TEST ON CHANNEL 5
1370 ;*****
1371 006762 000004 †ST36: SCOPE
1372 006764 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1373
1374
1375 ;* MOV $TMDAT,@VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1376 007002 004437 011716 JSR R4,SAR ;S A R ROUTINE
1377 007006 001606 VCYREG ;Y AXIS
1378 007010 000 005 .BYTE 0,5 ;CHANNEL 5, COARSE X AND FINE Y
1379 007012 001000 1000 ;EDGE VALUE
1380 007014 100000 BIT15 ;PEAK RESULTS
1381
1382 007016 010537 001126 MOV R5,$BDDAT ;SAVE RESULTS (X DAC + A/D PEAK NOISE, 1=.01 LSB)
1383 007022 163737 016256 001126 SUB CH7PEK,$BDDAT ;X DAC PEAK NOISE ONLY
1384 007030 013737 001126 016266 MOV $BDDAT,XDACPK ;SAVE RESULT
1385 007036 013737 016252 001124 MOV PEKMAX,$GDDAT ;LOAD EXPECTED
1386 007044 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULTS
1387 007052 002001 BGE TST37 ;;BR IF NOISE WITHIN SPEC
1388 007054 104006 ERROR 6 ;X D/A PEAK NOISE IS GREATER
1389 ; THAN SPEC
1390
1391 ;*****
1392 ;*TEST 37 Y DAC RMS NOISE TEST ON CHANNEL 6
1393 ;*****
1394 007056 000004 †ST37: SCOPE
1395 007060 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1396
1397
1398 ;* MOV $TMDAT,@VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1399 007076 004437 011716 JSR R4,SAR ;S A R ROUTINE
1400 007102 001604 VCXREG ;X AXIS
1401 007104 000 006 .BYTE 0,6 ;CHANNEL 6, COARSE Y AND FINE X
1402 007106 001000 1000 ;EDGE VALUE
1403 007110 000000 0 ;RMS RESULTS
1404
1405 007112 010537 001126 MOV R5,$BDDAT ;SAVE RESULTS (Y DAC + A/D RMS NOISE, 1=.01 LSB)
1406 007116 163737 016254 001126 SUB CH7RMS,$BDDAT ;Y DAC RMS NOISE ONLY
1407 007124 013737 001126 016270 MOV $BDDAT,YDACRM ;SAVE RESULT
1408 007132 013737 016250 001124 MOV RMSMAX,$GDDAT ;LOAD EXPECTED

```

```

1409 007140 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE RESULTS
1410 007146 002001                    BGE      TST40              ;;BR IF NOISE WITHIN SPEC
1411 007150 104006                    ERROR    6                  ;Y D/A RMS NOISE IS GREATER
1412                                     ;      THAN SPEC
1413
1414                                     ;*****
1415 ;*TEST 40      Y DAC PEAK NOISE TEST ON CHANNEL 6
1416 ;*****
1417 007152 000004      †ST40:  SCOPE
1418 007154 012737      000010 001166      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
1419
1420
1421 ;*      MOV      $TMDAT,$VCYREG      ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1422 007172 004437      011716      JSR      R4,$SAR          ;S A R ROUTINE
1423 007176 001604                    VCXREG                    ;X AXIS
1424 007200 000000      006      .BYTE    0,6            ;CHANNEL 6, COARSE Y AND FINE X
1425 007202 001000                    1000                      ;EDGE VALUE
1426 007204 100000                    BITIS                      ;PEAK RESULTS
1427
1428 007206 010537      001126      MOV      R5,$BDDAT        ;SAVE RESULTS (Y DAC + A/D PEAK NOISE, 1=.01 LSB
1429 007212 163737      016256      001126      SUB      CH7PEK,$BDDAT    ;Y DAC PEAK NOISE ONLY
1430 007220 013737      001126      016272      MOV      $BDDAT,$DACPK    ;SAVE RESULT
1431 007226 013737      016252      001124      MOV      PEKMAX,$GDDAT    ;LOAD EXPECTED
1432 007234 023737      001124      001126      CMP      $GDDAT,$BDDAT    ;COMPARE RESULTS
1433 007242 002001                    BGE      1$                ;;BR IF NOISE WITHIN SPEC
1434 007244 104006                    ERROR    6                  ;Y DAC PEAK NOISE IS GREATER
1435                                     ;      THAN SPEC
1436
1437 007246 032777      010000 171664 1$:  BIT      #BIT12,$SWR      ;TEST FORCED PRINTOUT
1438 007254 001432                    BEQ      TST41              ;;BR IF NOT FORCED
1439 007256 104401      014667      TYPE     NOIMSG            ;TYPE 2 OCTAL COL.
1440 007262 004537      012542      JSR      R5,BY TWO
1441 007266 016254                    CH7RMS
1442 007270 016256                    CH7PEK
1443 007272 104401      015010      TYPE     NOIMG1            ;TYPE 2 OCTAL COL.
1444 007276 004537      012542      JSR      R5,BY TWO
1445 007302 016260                    CH47RM
1446 007304 016262                    CH47PK
1447 007306 104401      015033      TYPE     NOIMG2            ;TYPE 2 OCTAL COL.
1448 007312 004537      012542      JSR      R5,BY TWO
1449 007316 016264                    XDACRM
1450 007320 016266                    XDACPK
1451 007322 104401      015056      TYPE     NOIMG3            ;TYPE 2 OCTAL COL.
1452 007326 004537      012542      JSR      R5,BY TWO
1453 007332 016270                    YDACRM
1454 007334 016272                    YDACPK
1455 007336 104401      020040      TYPE     ,ACRLF
1456
1457                                     ;*****
1458 ;*TEST 41      BIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
1459 ;*****
1460 007342 000004      †ST41:  SCOPE
1461 007344 012737      000010 001166      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
1462 007352 004437      011716      JSR      R4,$SAR          ;SOFTWARE S.A.R.

```

M03

MAINDEC-11-DRLPE-A  
DRLPE P11 T41

MACY11 27(654) 14-DEC-77 20:24 PAGE 30  
BIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT

SEQ 0039

```

1463 007356 001606 VCYREG
1464 007360 000 007 .BYTE 0,CH7 ;CHANNEL 7, FINE Y
1465 007362 001001 1001 ;EDGE 1000/1001 TRANSITION
1466 007364 000200 BIT7 ;50-50
1467
1468 007366 013737 016246 016274 MOV DACSAV,OFSTBX ;SAVE A/D OFFSET BIPOLAR
1469 007374 013737 016246 001126 MOV DACSAV,$BDDAT ;LOAD VALUE READ
1470 007402 012737 001014 001124 MOV #1014,$GDDAT ;LOAD EXPECTED
1471 007410 013737 020276 013010 MOV V62,SPREAD ;LOAD +- VARIABLE, 1LSB
1472 007416 004737 013012 JSR PC,COMPAR ;TEST IF WITHIN +- VALUE
1473 007422 000401 BR TST42 ;;BR IF WITHIN LIMITS
1474 007424 104012 ERROR 12 ;A/D BIPOLAR OFFSET ERROR
1475
1476 ;*****
1477 ;*TEST 42 UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
1478 ;*****
1479 007426 000004 TST42: SCOPE
1480 007430 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1481 007436 004437 011716 JSR R4,SAR ;SOFTWARE S.A.R.
1482 007442 001606 VCYREG
1483 007444 000 047 .BYTE 0,CH47 ;CHANNEL 7, UNIPOLAR FINE Y
1484 007446 000001 1 ;EDGE 0/1 TRANSITION
1485 007450 000200 BIT7 ;50-50
1486
1487 007452 013737 016246 016276 MOV DACSAV,OFSTUX ;SAVE A/D OFFSET UNIPOLAR
1488 007460 013737 016246 001126 MOV DACSAV,$BDDAT ;LOAD VALUE READ
1489 007466 012737 001014 001124 MOV #1014,$GDDAT ;LOAD EXPECTED
1490 007474 013737 020276 013010 MOV V62,SPREAD ;LOAD +- VARIABLE, 1 LSB
1491 007502 004737 013012 JSR PC,COMPAR ;TEST IF WITHIN +- VALUE
1492 007506 000401 BR TST43 ;;BR IF WITHIN LIMITS
1493 007510 104012 ERROR 12 ;A/D UNIPOLAR OFFSET ERROR
1494
1495 ;*****
1496 ;*TEST 43 BIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
1497 ;*****
1498 007512 000004 TST43: SCOPE
1499 007514 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1500 007522 004437 011716 JSR R4,SAR ;SOFTWARE S.A.R.
1501 007526 001606 VCYREG
1502 007530 000 007 .BYTE 0,CH7 ;CHANNEL 7, FINE Y
1503 007532 001001 1001 ;EDGE 1000/1001 TRANSITION
1504 007534 000201 BIT7:BIT0 ;50-50, WAIT LOOP
1505
1506 007536 013737 016246 016300 MOV DACSAV,OFSLBX ;SAVE A/D OFFSET BIPOLAR
1507 007544 013737 016246 001126 MOV DACSAV,$BDDAT ;LOAD VALUE READ
1508 007552 012737 001014 001124 MOV #1014,$GDDAT ;LOAD EXPECTED
1509 007560 013737 020276 013010 MOV V62,SPREAD ;LOAD +- VARIABLE, 1 LSB
1510 007566 004737 013012 JSR PC,COMPAR ;TEST IF WITHIN +- VALUE
1511 007572 000401 BR 15 ;;BR IF WITHIN LIMITS
1512 007574 104012 ERROR 12 ;A/D BIPOLAR OFFSET ERROR
1513 007576 163737 016274 001126 15: SUB OFSTBX,$BDDAT ;OFFSET DUE TO WAIT LOOP
1514 007604 005037 001124 CLR $GDDAT ;
1515 007610 012737 000031 013010 MOV #31,SPREAD ;1/2 LSB ALLOWED
1516 007616 004737 013012 JSR PC,COMPAR ;WITHIN LIMITS

```

```

1517 007622 000401 BR TST44 ;;BR IF WITHIN LIMITS
1518 007624 104012 ERROR 12 ;A/D BIPOLAR OFFSET ERROR DUE TO WAIT LOOP
1519
1520
1521 ;:*****
1522 ;*TEST 44 UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
1523 ;:*****
1524 007626 000004 †ST44: SCOPE
1525 007630 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1526 007636 004437 011716 JSR R4,SAR ;SOFTWARE S.A.R.
1527 007642 001606 VCYREG
1528 007644 000 .BYTE 0,CH47 ;CHANNEL 7, UNIPOLAR, FINE Y
1529 007650 000001 i ;EDGE 0/1 TRANSITION
1530 000201 BIT7!BIT0 ;50-50, WAIT LOOP
1531 007652 013737 016246 016302 MOV DACSAV,OFSLUX ;SAVE A/D OFFSET UNIPOLAR
1532 007660 013737 016246 001126 MOV DACSAV,$BDDAT ;LOAD VALUE READ
1533 007666 012737 001014 001124 MOV #1014,$GDDAT ;LOAD EXPECTED
1534 007674 013737 020276 013010 MOV V62,SPREAD ;LOAD +- VARIABLE, 1 LSB
1535 007702 004737 013012 JSR PC,COMPAR ;TEST IF WITHIN +- VALUE
1536 007706 000401 BR 15 ;;BR IF WITHIN LIMITS
1537 007710 104012 ERROR 12 ;A/D UNIPOLAR OFFSET ERROR
1538
1539 007712 163737 016276 001126 15: SUB OFSTUX,$BDDAT ;OFFSET DUE TO WAIT LOOP
1540 007720 005037 001124 CLR $GDDAT ;
1541 007724 012737 000031 013010 MOV #31,SPREAD ;1/2 LSB ALLOWED
1542 007732 004737 013012 JSR PC,COMPAR ;WITHIN LIMITS
1543 007736 000401 BR TST45 ;;BR IF WITHIN LIMITS
1544 007740 104012 ERROR 12 ;A/D UNIPOLAR OFFSET ERROR DUE TO WAIT LOOP
1545 ;:*****
1546 ;*TEST 45 X D/A OFFSET USING CH 5
1547 ;:*****
1548 007742 000004 †ST45: SCOPE
1549 007744 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1550
1551 ;* MOV $TMDAT,$VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1552 007762 004437 011716 JSR R4,SAR ;SOFTWARE S.A.R.
1553 007766 001606 VCYREG ;Y AXIS
1554 007770 000 005 .BYTE 0,CH5 ;CHANNEL 5, COARSE X AND FINE Y
1555 007772 001001 i001 ;EDGE VALUE, 1000/1001 TRANSITION
1556 007774 000200 BIT7 ;50-50
1557 007776 013737 016246 001126 MOV DACSAV,$BDDAT ;SAVE RESULT (X DAC + A/D OFFSET)
1558 010004 163737 016274 001126 SUB OFSTBX,$BDDAT ;OFFSET DUE TO X DAC
1559 010012 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
1560 010016 013737 020300 013010 MOV V77,SPREAD ;TOLERANCE DUE TO X DAC
1561 010024 004737 013012 JSR PC,COMPAR ;TEST IF WITHIN +- LIMITS
1562 010030 000401 BR TST46 ;;BR IF WITHIN LIMITS
1563 010032 104012 ERROR 12 ;X DAC OFFSET ERROR
1564
1565 ;:*****
1566 ;*TEST 46 Y D/A OFFSET USING CH 6
1567 ;:*****
1568 010034 000004 †ST46: SCOPE
1569 010036 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1570

```



```

1571 ;* MOV $TMDAT,AVCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1572 JSR R4,SAR ;SOFTWARE S.A.R.
1573 VCXREG ;X AXIS
1574 .BYTE 000 006 ;CHANNEL 6, COARSE Y AND FINE X
1575 1001 ;EDGE VALUE, 1000/1001 TRANSITION
1576 BIT7 ;50-50
1577 MOV DACSAV,$BDDAT ;SAVE RESULT (Y DAC + A/D OFFSET)
1578 SUB OFSTBX,$BDDAT ;OFFSET DUE TO y DAC
1579 CLR $GDDAT ;CLEAR EXPECTED
1580 MOV V77,SPREAD ;TOLERANCE DUE TO Y DAC
1581 JSR PC,COMPAR ;TEST IF WITHIN +- LIMITS
1582 BR TST47 ;;BR IF WITHIN LIMITS
1583 ERROR 12 ;Y DAC OFFSET ERROR
1584 ;*****
1585 ;*TEST 47 CALIBRATION USING CHANNEL 11 - X DAC VS. A/D
1586 ;*****
1587 †ST47: SCOPE
1588 MOV #10,$TIMES ;;DO 10 ITERATIONS
1589 MOV #1300,$TMDAT ;LOAD X DAC
1590
1591 ;* MOV $TMDAT,AVCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1592 MOV #1600,$GDDAT ;LOAD EXPECTED
1593 JSR R5,CONVRT ;CONVERT 32X AND AVERAGE
1594 CH11
1595 MOV ADEND,$BDDAT ;READ VALUE OBTAINED
1596 MOV #3,SPREAD ;LOAD DEVIATION
1597 JSR PC,COMPAR ;TEST IF WITHIN
1598 BR 1$ ;BR IF WITHIN
1599 ERROR 13 ;CALIBRATION ERROR - X DAC VS. A/D
1600
1601 1$:
1602 ;* MOV AVCXREG,$TMDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $TMDAT.
1603 SUB #100,$TMDAT
1604
1605 ;* MOV $TMDAT,AVCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1606 SUB #200,$GDDAT ;DEC. EXPECTED
1607 BPL 2$ ;BR UNTIL DONE
1608
1609 ;*****
1610 ;*TEST 50 CALIBRATION USING CHANNEL 12 - Y DAC VS. A/D
1611 ;*****
1612 †ST50: SCOPE
1613 MOV #10,$TIMES ;;DO 10 ITERATIONS
1614 MOV #1300,$TMDAT ;LOAD Y DAC
1615
1616 ;* MOV $TMDAT,AVCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1617 MOV #1600,$GDDAT ;LOAD EXPECTED
1618 JSR R5,CONVRT ;CONVERT 32X AND AVERAGE
1619 CH12
1620 MOV ADEND,$BDDAT ;READ VALUE OBTAINED
1621 MOV #3,SPREAD ;LOAD DEVIATION
1622 JSR PC,COMPAR ;TEST IF WITHIN
1623 BR 1$ ;BR IF WITHIN
1624 ERROR 13 ;CALIBRATION ERROR - Y DAC VS. A/D

```

```

1625 010340          1$:
1626
1627
1628 010350 162737 000100 001564 ;*   MOV   @VCYREG,$TMDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
1629          SUB   #100,$TMDAT
1630
1631 010366 162737 000200 001124 ;*   MOV   $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1632 010374 100344          SUB   #200,$GDDAT      ;DEC. EXPECTED
1633          BPL   2$          ;BR UNTIL DONE
1634          ;*****
1635          ;*TEST 51 LINEARITY TEST USING CH 6 - Y DAC VS. A/D
1636          ;*****
1636 010376 000004          †ST51: SCOPE
1637 010400 012737 000010 001166   MOV   #10,$TIMES      ;;DO 10 ITERATIONS
1638 010406 012700 016026          MOV   #NUMBF2,R0      ;LOAD EDGE VALUE POINTER
1639 010412 012702 016052          MOV   #RSLTO,R2      ;LOAD RESULT POINTER
1640
1641 010416 011037 010446          1$:   MOV   (R0),10$        ;LOAD EDGE EXPECTED
1642 010422 012037 001564          MOV   (R0)+,$TMDAT   ;LOAD DAC
1643
1644          ;*   MOV   $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1645 010436 004437 011716          JSR   R4,SAR         ;SOFTWARE S.A.R.
1646 010442 001604          VCXREG
1647 010444          000          .BYTE 0,CH6          ;CHANNEL 6, COARSE Y AND FINE X
1648 010446 000000          10$: 0              ;EDGE VALUE
1649 010450 000200          BIT7              ;50-50
1650
1651 010452 013722 016246          MOV   DACSAV,(R2)+   ;SAVE RESULTS
1652 010456 005710          TST   (R0)          ;LAST RESULT ?
1653 010460 100356          BPL   1$            ;BR UNTIL DONE
1654 010462 005037 001124          CLR   $GDDAT        ;ZERO NOMINAL LINEARITY ERROR
1655 010466 013737 020300 013010   MOV   V77,SPREAD    ;TOLERANCE FOR A/D - DAC
1656 010474 013700 016072          MOV   RSLTO+20,R0   ;GET LAST VALUE
1657 010500 163700 016052          SUB   RSLTO,R0      ;SUBTRACT FIRST VALUE
1658 010504 006200          ASR   R0
1659 010506 006200          ASR   R0
1660 010510 006200          ASR   R0
1661 010512 005500          ADC   R0            ;AVERAGE DIFFERENCE IN R0
1662 010514 012701 000002          MOV   #2,R1
1663 010520 005037 016072          CLR   RSLTO+20      ;START RUNNING SUM OF ERRORS
1664 010524 060037 016052          2$:  ADD   R0,RSLTO      ;COMPUTE NOM. VALUE ON LINE BETW. END PTS.
1665 010530 163761 016052 016052   SUB   RSLTO,RSLTO(R1) ;COMPUTE ERROR WITH RESP. TO END PT. LINE
1666 010536 066137 016052 016072   ADD   RSLTO(R1),RSLTO+20 ;KEEP RUNNING SUM
1667 010544 005721          TST   (R1)+         ;BUMP R1
1668 010546 020127 000020          CMP   R1,#20        ;DONE ?
1669 010552 001364          BNE   2$            ;BR IF NOT
1670 010554 006237 016072          ASR   RSLTO+20
1671 010560 006237 016072          ASR   RSLTO+20
1672 010564 006237 016072          ASR   RSLTO+20
1673 010570 005537 016072          ADC   RSLTO+20      ;AVERAGE ERROR WITH RESP. TO END PT. LINE
1674 010574 005037 016052          CLR   RSLTO
1675 010600 005001          CLR   R1
1676 010602 163761 016072 016052   3$:  SUB   RSLTO+20,RSLTO(R1) ;ERROR WITH RESP. TO "BEST STRAIGHT LINE"
1677
1678 010610 016137 016052 001126          MOV   RSLTO(R1),$BDDAT

```

```

1679 010616 004737 013012 JSR PC,COMPAR ;COMPARE LINEARITY ERROR WITH + OR - TOLERANCE
1680 010622 000422 BR 4$
1681 010624 010102 MOV R1,R2
1682 010626 042702 177770 BIC #177770,R2 ;MASK BITS 0-2
1683 010632 062702 000060 ADD #60,R2 ;CONVERT TO ASCII
1684 010636 110237 013571 MOVB R2,EM14A
1685 010642 010102 MOV R1,R2 ;LOAD R2
1686 010644 006202 ASR R2
1687 010646 006202 ASR R2
1688 010650 006202 ASR R2
1689 010652 042702 177770 BIC #177770,R2 ;MASK
1690 010656 062702 000060 ADD #60,R2 ;MAKE ASCII
1691 010662 110237 013570 MOVB R2,EM14B ;SAVE #
1692 010666 104014 ERROR 14 ;LINEARITY ERROR USING CH 6 - YDAC VS. A/D
1693 010670 005721 4$: TST (R1)+ ;BUMP R1
1694 010672 020127 000020 CMP R1,#20 ;DONE ?
1695 010676 001341 BNE 3$ ;;BR IF NOT DONE
1696
1697 ;*****
1698 ;*TEST 52 LINEARITY TEST USING CH 5 - X DAC VS. A/D
1699 ;*****
1700 010700 000004 †ST52: SCOPE
1701 010702 012737 000010 001166 MOV #10,$TIMES ;;DO 10 ITERATIONS
1702 010710 012700 016026 MOV #NUMBF2,R0 ;LOAD EDGE VALUE POINTER
1703 010714 012702 016110 MOV #RSLT1,R2 ;LOAD RESULT POINTER
1704
1705 010720 011037 010750 1$: MOV (R0),10$ ;LOAD EDGE EXPECTED
1706 010724 012037 001564 MOV (R0)+,$TMDAT ;LOAD DAC
1707
1708 ;* MOV $TMDAT,$VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1709 010740 004437 011716 JSR R4,SAR ;SOFTWARE S.A.R.
1710 010744 001606 VCYREG
1711 010746 000 005 .BYTE 0,CH5 ;CHANNEL 5, COARSE X AND FINE Y
1712 010750 000000 10$: 0 ;EDGE VALUE
1713 010752 000200 BIT7 ;50-50
1714
1715 010754 013722 016246 MOV DACSAV,(R2)+ ;SAVE RESULTS
1716 010760 005710 TST (R0) ;LAST RESULT ?
1717 010762 100356 BPL 1$ ;BR UNTIL DONE
1718 010764 005037 001124 CLR $GDDAT ;ZERO NOMINAL LINEARITY ERROR
1719 010770 013737 020300 013010 MOV V77,SPREAD ;TOLERANCE FOR A/D + DAC
1720 010776 013700 016130 MOV RSLT1+20,R0 ;GET LAST VALUE
1721 011002 163700 016110 SUB RSLT1,R0 ;SUBTRACT FIRST VALUE
1722 011006 006200 ASR R0
1723 011010 006200 ASR R0
1724 011012 006200 ASR R0
1725 011014 005500 ADC R0 ;AVERAGE DIFFERENCE IN R0
1726 011016 012701 000002 MOV #2,R1
1727 011022 005037 016130 CLR RSLT1+20 ;START RUNNING SUM OF ERRORS
1728 011026 060037 016110 2$: ADD R0,RSLT1 ;COMPUTE NOM. VALUE ON LINE BETW. END PTS.
1729 011032 163761 016110 016110 SUB RSLT1,RSLT1(R1) ;COMPUTE ERROR WITH RESP. TO END PT. LINE
1730 011040 066137 016110 016130 ADD RSLT1(R1),RSLT1+20 ;KEEP RUNNING SUM
1731 011046 005721 TST (R1)+ ;BUMP R1
1732 011050 020127 000020 CMP R1,#20 ;DONE ?

```

```

1733 011054 001364 BNE 2$ ;BR IF NOT
1734 011056 006237 016130 ASR RSLT1+20
1735 011062 006237 016130 ASR RSLT1+20
1736 011066 006237 016130 ASR RSLT1+20
1737 011072 005537 016130 ADC RSLT1+20 ;AVERAGE ERROR WITH RESP. TO END PT. LINE
1738 011076 005037 016110 CLR RSLT1
1739 011102 005001 CLR R1
1740 011104 163761 016130 016110 3$: SUB RSLT1+20,RSLT1(R1) ;ERROR WITH RESP. TO "BEST STRAIGHT LINE"
1741 011112 016137 016110 001126 MOV RSLT1(R1),%BDDAT
1742 011120 004737 013012 JSR PC,COMPAR ;COMPARE LINEARITY ERROR WITH + OR - TOLERANCE
1743 011124 000422 BR 4$
1744 011126 010102 MOV R1,R2
1745 011130 042702 177770 BIC #177770,R2 ;MASK BITS 0-2
1746 011134 062702 000060 ADD #60,R2 ;CONVERT TO ASCII
1747 011140 110237 013571 MOVB R2,%M14A
1748 011144 010102 MOV R1,R2 ;LOAD R2
1749 011146 006202 ASR R2
1750 011150 006202 ASR R2
1751 011152 006202 ASR R2
1752 011154 042702 177770 BIC #177770,R2 ;MASK
1753 011160 062702 000060 ADD #60,R2 ;MAKE ASCII
1754 011164 110237 013570 MOVB R2,%M14B ;SAVE #
1755 011170 104014 ERROR 14 ;LINEARITY ERROR USING CH 5 - XDAC VS. A/D
1756 011172 005721 4$: TST (R1)+ ;BUMP R1
1757 011174 020127 000020 CMP R1,#20 ;DONE ?
1758 011200 001341 BNE 3$ ;;BR IF NOT DONE
1759 011202 032777 010000 167730 BIT #BIT12,%SWR ;TEST IF FORCED TYPEOUT
1760 011210 001416 BEQ TST53 ;;BR IF NOT FORCED
1761 011212 104401 TYPE LINMSG
1762 011216 004537 012564 JSR AS,TYPSTG ;TYPE OCTAL STRING
1763 011222 016052 RSLTO ;STARTING AT
1764 011224 000010 B. ; # OF LOCATIONS
1765 011226 104401 015264 TYPE LINMG1
1766 011232 004537 012564 JSR AS,TYPSTG ;TYPE OCTAL STRING
1767 011236 016110 RSLT1 ;STARTING AT
1768 011240 000010 B. ; # OF LOCATIONS
1769 011242 104401 020040 TYPE ,ACRLF
1770
1771 ;*****
1772 ;*TEST 53 DETERMINE IF MORE AR11'S ARE TO BE TESTED
1773 ;*****
1774 011246 000004 †T53: SCOPE
1775 011250 012737 000001 001166 MOV #1,$TIMES ;;DO 1 ITERATION
1776 011256 005737 001566 TST NBEXT ;TEST IF ANY
1777 011262 001411 BEQ 1$ ;BR IF NONE
1778 011264 062737 000020 001556 ADD #20,ARBADD ;UPDATE DEVICE ADDRESS
1779 011272 062737 000020 001560 ADD #20,ARBVCT ;UPDATE DEVICE VECTOR
1780 011300 005337 001566 DEC NBEXT ;ANOTHER ONE ?
1781 011304 000415 BR BYPAS1 ;BR IF ANOTHER
1782 011306 013737 001256 001556 1$: MOV $BASE,ARBADD ;RELOAD ADDRESS
1783 011314 013737 001252 001560 MOV $VECT1,ARBVCT ;RELOAD VECTOR
1784 011322 013737 001562 001566 MOV NMBEXT,NBEXT ;RELOAD NUMBER
1785 011330 004737 025300 JSR PC,$RESET
1786 011334 000137 011350 JMP $EOP ;DONE

```

```

1787 011340 012700 177777
1788 011344 000137 002414
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800 011350
1801 011350 000004
1802 011352 005037 001102
1803 011356 005037 001166
1804 011362 005237 001210
1805 011366 042737 100000 001210
1806 011374 005327
1807 011376 000001
1808 011400 003022
1809 011402 012737
1810 011404 000001
1811 011406 011376
1812 011410 104401 011455
1813 011414 013746 001210
1814 011420 104405
1815 011422 104401 011452
1816 011426 013700 000042
1817 011432 001405
1818 011434 000005
1819 011436 004710
1820 011440 000240
1821 011442 000240
1822 011444 000240
1823 011446
1824 011446 000137
1825 011450 011340
1826 011452 377 377 000
1827 011455 015 042412 042116
1828 011462 050040 051501 020123
1829 011470 000043
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
    
```

```

BYPAS1: MOV     #-1,RO
        JMP     RBEG2           ;TEST ANOTHER UNIT
        .SBTTL

.SBTTL  END OF PASS ROUTINE

;*****
; *INCREMENT THE PASS NUMBER ($PASS)
; *INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
; *TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
; *IF THERES A MONITOR GO TO IT
; *IF THERE ISN'T JUMP TO BYPAS1

$EOP:
        CLR     $STNM           ;; ZERO THE TEST NUMBER
        CLR     $TIMES          ;; ZERO THE NUMBER OF ITERATIONS
        INC     $PASS           ;; INCREMENT THE PASS NUMBER
        BIC     #100000,$PASS   ;; DON'T ALLOW A NEG. NUMBER
        DEC     (PC)+           ;; LOOP?
$EOPCT: .WORD    1
        BGT     $DOAGN          ;; YES
        MOV     (PC)+,$(PC)+    ;; RESTORE COUNTER
$ENDCT: .WORD    1
        $EOPCT
        TYPE    $SENDMG         ;; TYPE "END PASS #"
        MOV     $PASS,-(SP)     ;; SAVE $PASS FOR TYPEOUT
        TYPDS   $SENDMG         ;; GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE    $SENDMG         ;; TYPE A NULL CHARACTER
$GET42: MOV     #42,RO          ;; GET MONITOR ADDRESS
        BEQ     $DOAGN          ;; BRANCH IF NO MONITOR
        RESET   $DOAGN          ;; CLEAR THE WORLD
$ENDAD: JSR     PC,(RO)         ;; GO TO MONITOR
        NOP
        NOP
        NOP                     ;; FOR
        NOP                     ;; ACT11
$DOAGN: JMP     @(PC)+           ;; RETURN
$RTNAD: .WORD   BYPAS1
$ENULL: .BYTE   -1,-1,0        ;; NULL CHARACTER STRING
$ENDMG: .ASCIZ  <15><12>/END PASS #/
    
```

```

.SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

;*****
; *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
; *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
; *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
; *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
; *REPLACED WITH SPACES.
; *CALL:
    
```

1841				;* MOV NUM,-(SP)	:: PUT THE BINARY NUMBER ON THE STACK
1842				;* TYPDS	:: GO TO THE ROUTINE
1843					
1844	011472			\$TYPDS:	
1845	011472	010046		MOV R0,-(SP)	:: PUSH R0 ON STACK
1846	011474	010146		MOV R1,-(SP)	:: PUSH R1 ON STACK
1847	011476	010246		MOV R2,-(SP)	:: PUSH R2 ON STACK
1848	011500	010346		MOV R3,-(SP)	:: PUSH R3 ON STACK
1849	011502	010546		MOV R5,-(SP)	:: PUSH R5 ON STACK
1850	011504	012746	020200	MOV #20200,-(SP)	:: SET BLANK SWITCH AND SIGN
1851	011510	016605	000020	MOV 20(SP),R5	:: GET THE INPUT NUMBER
1852	011514	100004		BPL 1\$	:: BR IF INPUT IS POS.
1853	011516	005405		NEG R5	:: MAKE THE BINARY NUMBER POS.
1854	011520	112766	000055 000001	MOVB #'-,1(SP)	:: MAKE THE ASCII NUMBER NEG.
1855	011526	005000		1\$: CLR R0	:: ZERO THE CONSTANTS INDEX
1856	011530	012703	011706	MOV #SDBLK,R3	:: SETUP THE OUTPUT POINTER
1857	011534	112723	000040	MOVB #' ,(R3)+	:: SET THE FIRST CHARACTER TO A BLANK
1858	011540	005002		2\$: CLR R2	:: CLEAR THE BCD NUMBER
1859	011542	016001	011676	MOV \$DTBL(R0),R1	:: GET THE CONSTANT
1860	011546	16C105		3\$: SUB R1,R5	:: FORM THIS BCD DIGIT
1861	011550	002402		4\$: BLT 4\$	:: BR IF DONE
1862	011552	005202		INC R2	:: INCREASE THE BCD DIGIT BY 1
1863	011554	000774		BR 3\$	
1864	011556	060105		4\$: ADD R1,R5	:: ADD BACK THE CONSTANT
1865	011560	005702		TST R2	:: CHECK IF BCD DIGIT=0
1866	011562	001002		BNE 5\$	:: FALL THROUGH IF 0
1867	011564	105716		TSTB (SP)	:: STILL DOING LEADING 0'S?
1868	011566	100407		BMI 7\$	:: BR IF YES
1869	011570	106316		5\$: ASLB (SP)	:: MSD?
1870	011572	103003		BCC 6\$	:: BR IF NO
1871	011574	116663	000001 177777	MOVB 1(SP),-1(R3)	:: YES--SET THE SIGN
1872	011602	052702	000060	BIS #'0,R2	:: MAKE THE BCD DIGIT ASCII
1873	011606	052702	000040	7\$: BIS #' ,R2	:: MAKE IT A SPACE IF NOT ALREADY A DIGIT
1874	011612	110223		MOVB R2,(R3)+	:: PUT THIS CHARACTER IN THE OUTPUT BUFFER
1875	011614	005720		TST (R0)+	:: JUST INCREMENTING
1876	011616	020027	000010	CMP R0,#10	:: CHECK THE TABLE INDEX
1877	011622	002746		BLT 2\$	:: GO DO THE NEXT DIGIT
1878	011624	003002		BGT 8\$	:: GO TO EXIT
1879	011626	010502		MOV R5,R2	:: GET THE LSD
1880	011630	000764		BR 6\$	:: GO CHANGE TO ASCII
1881	011632	105726		8\$: TSTB (SP)+	:: WAS THE LSD THE FIRST NON-ZERO?
1882	011634	100003		9\$: BPL 9\$	:: BR IF NO
1883	011636	116663	177777 177776	MOV -1(SP),-2(R3)	:: YES--SET THE SIGN FOR TYPING
1884	011644	105013		9\$: CLRB (R3)	:: SET THE TERMINATOR
1885	011646	012605		MOV (SP)+,R5	:: POP STACK INTO R5
1886	011650	012603		MOV (SP)+,R3	:: POP STACK INTO R3
1887	011652	012602		MOV (SP)+,R2	:: POP STACK INTO R2
1888	011654	012601		MOV (SP)+,R1	:: POP STACK INTO R1
1889	011656	012600		MOV (SP)+,R0	:: POP STACK INTO R0
1890	011660	104401	011706	TYPE \$DBLK	:: NOW TYPE THE NUMBER
1891	011664	016666	000002 000004	MOV 2(SP),4(SP)	:: ADJUST THE STACK
1892	011672	012616		MOV (SP)+,(SP)	
1893	011674	000002		RTI	:: RETURN TO USER
1894	011676	023420		\$L TBL: 10000.	

1895 011700 001750  
1896 011702 000144  
1897 011704 000012  
1898 011706 000004  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912 011716 012437 016244  
1913 011722 012437 016312  
1914 011728 012437 016314  
1915 011732 012437 016316  
1916 011736 010046  
1917 011740 010146  
1918 011742 010246  
1919 011744 010346  
1920 011746 113737 016313 013006  
1921 011754 005037 013004  
1922 011760 113737 016313 013005  
1923 011766 005237 013004  
1924 011772 017737 004246 016244  
1925 012000 005037 016320  
1926 012004 005737 016316  
1927 012010 100003  
1928 012012 012703 000002  
1929 012016 000407  
1930 012020 012703 000121  
1931 012024 105737 016316  
1932 012030 100002  
1933 012032 012703 000400  
1934 012036 012702 001000  
1935 012042 005037 001564  
1936  
1937  
1938 012056 005001  
1939 012060 012700 001000  
1940  
1941  
1942 012074 060237 001564  
1943  
1944  
1945 012110 013737 013004 001564  
1946  
1947  
1948 012126

1000.  
100.  
10.  
\$DBLK: .BLKW 4  
.SBTTL SUCCESSIVE APPROX. SUBROUTINE  
:S A R SUBROUTINE  
JSR R4,SAR  
AXIS POINTER  
CH. IN HIGH BYTE / DUMMY CH. IN LOW BYTE  
EDGE VALUE  
RESULT/MODE  
RETURN WITH R5 = DIFFERENCE  
:  
BIT15 = 0 RMS  
BIT15 = 1 PEAK  
BIT7 = 1 50-50  
SAR: MOV (R4)+,ADAC ;SAVE DAC ADDRESS POINTER  
MOV (R4)+,SARCHN ;SAVE CHANNEL  
MOV (R4)+,EDGE ;SAVE VALUE EDGE  
MOV (R4)+,CONS ;SAVE RMS/PEAK SW  
MOV R0,-(SP) ;SAVE GPR  
MOV R1,-(SP)  
MOV R2,-(SP)  
MOV R3,-(SP)  
MOVB SARCHN+1,CHANL  
CLR FCHANL  
MOVB SARCHN+1,FCHANL+1  
INC FCHANL ;SET START BIT  
MOV @ADAC,ADAC ;GET BUS ADDRESS OF ACTIVE DAC  
CLR FINS ;CLEAR 2 PASS COUNTER FLAG  
TST CONS ;TEST IF RMS  
BPL 10\$ ;BR IF RMS  
MOV #2,R3 ;LOAD A = .4% OF 512  
BR 15\$  
10\$: MOV #121,R3 ;LOAD A = 16% OF 512  
TSTB CONS ;TEST FOR 50-50  
BPL 15\$  
15\$: MOV #400,R3 ;LOAD 50-50, A = 50% COUNT  
MOV #1000,R2 ;SET UP MSB  
CLR \$TMDAT  
;\* MOV \$TMDAT,@ADAC ;/ PUT DATA FROM \$TMDAT TO DEVICE REG ADAC  
14\$: CLR R1 ;SET HIGH = 0  
MOV #1000,R0 ;SET UP 512 CONVERSIONS  
;\* MOV @ADAC,\$TMDAT ;/READ DEVICE REG ADAC,PUT DATA IN \$TMDAT.  
ADD R2,\$TMDAT ;NEXT BIT OF ACTIVE DAC  
;\* MOV \$TMDAT,@ADAC ;/ PUT DATA FROM \$TMDAT TO DEVICE REG ADAC  
6\$: MOV FCHANL,\$TMDAT ;START CONVERSION  
;\* MOV \$TMDAT,@ADCS ;/ PUT DATA FROM \$TMDAT TO DEVICE REG ADCS  
12\$:

```

1949
1950
1951 012136 023737 001564 016314 ;*      MOV      @ADDBR,$TMDAT ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
1952 012144 002401                BLT      $TMDAT,EDGE
1953 012146 005201                INC      R1                ;YES, BR
1954 012150 105737 016312 1$:      TSTB    SARCHN            ;NO, INC HIGH FOR RESULT > OR = EDGE VALUE
1955 012154 001417                BEQ      3$                ;TEST IF DUMMY CH. IS SET
1956 012156 005037 001564                CLR      $TMDAT           ;BR IF NOT
1957 012162 113737 016312 001565        MOVB    SARCHN,$TMDAT+1   ;LOAD MUX
1958 012170 005237 001564                INC      $TMDAT
1959
1960                ;*      MOV      $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1961 012204                4$:
1962 012204                13$:
1963
1964                ;*      MOV      @ADDBR,$TMDAT ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
1965 012214 005300                3$:      DEC      R0                ;DONE 512 TIMES ?
1966 012216 001334                BNE     6$                ;NO
1967 012220 020103                CMP     R1,R3             ;IS HIGH > OR = A ?
1968 012222 002412                BLT     2$                ;NO, LEAVE BIT ON
1969
1970                ;*      MOV      @ADAC,$TMDAT ;/READ DEVICE REG ADAC,PUT DATA IN $TMDAT.
1971 012234 160237 001564                SUB     R2,$TMDAT        ;YES, TAKE IT OUT
1972
1973                ;*      MOV      $TMDAT,@ADAC ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADAC
1974 012250 022702 000001                2$:      CMP     #1,R2             ;TEST FOR Z = 1, S.A.R. DONE ?
1975 012254 001402                BEQ     21$              ;BR IF DONE
1976 012256 006202                ASR    R2                ;NO SET Z /2
1977 012260 000676                BR     14$              ;TRY AGAIN
1978 012262                21$:
1979
1980                ;*      MOV      @ADAC,DACSAV ;/READ DEVICE REG ADAC,PUT DATA IN DACSAV.
1981 012272 005737 016320                TST    FINS              ;FIRST OR SECOND TIME ?
1982 012276 001025                BNE    45$              ;SECOND, BR
1983
1984                ;*      MOV      @ADAC,$TMDAT ;/READ DEVICE REG ADAC,PUT DATA IN $TMDAT.
1985 012310 013705 001564                MOV     $TMDAT,R5        ;READ VALUE
1986 012314 005237 016320                INC     FINS              ;SET FLAG
1987 012320 005737 016316                TST    CONS              ;TEST IF PEAK/RMS
1988 012324 100003                BPL    41$              ;BR IF RMS
1989 012326 012703 000776                MOV     #776,R3          ;LOAD 99.6% OF 512
1990 012332 000405                BR     42$
1991 012334 012703 000657                41$:    MOV     #657,R3          ;SET A = 84% OF 512
1992 012340 105737 016316                43$:    TSTB   CONS
1993 012344 100411                BMI    5$
1994 012346 000137 012036                42$:    JMP     15$              ;DO MORE TESTING
1995 012352                45$:
1996
1997                ;*      MOV      @ADAC,$TMDAT ;/READ DEVICE REG ADAC,PUT DATA IN $TMDAT.
1998 012362 163705 001564                SUB     $TMDAT,R5        ;SUBTRACT DIFF
1999 012366 005405                NEG     R5
2000 012370 005037 001564                5$:     CLR     $TMDAT           ;RETURN WITH 2 X NOISE IN R5
2001
2002                ;*      MOV      $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS

```



```

2003 012404 012603      MOV      (SP)+,R3
2004 012406 012602      MOV      (SP)+,R2
2005 012410 012601      MOV      (SP)+,R1
2006 012412 012600      MOV      (SP)+,R0
2007 012414 000204      RTS      R4          ;EXIT
2008
2009                      .SBTTL  MISC. SUBROUTINES
2010                      ;CONVERT R2 INTO DECIMAL DIGITS
2011
2012 012416 012737 177774 012522 DECpRT: MOV      #-4,DIGCNT      ;LOAD COUNT
2013 012424 010146      MOV      R1,-(SP)      ;SAVE R1
2014 012426 012701 012536      MOV      #DIGT1-1,R1   ;LOAD POINTER
2015 012432 012737 012526 012524      MOV      #DECPNT+2,DECPNT ;LOAD POINTER
2016 012440 012737 177777 012520 1$:  MOV      #-1,DIGIT      ;LOAD #
2017 012446 005237 012520 2$:  INC      DIGIT          ;UPDATE IT
2018 012452 167702 000046      SUB      @DECPNT,R2     ;SUB MAGNITUDE
2019 012456 100373      BPL      2$            ;BR IF +
2020 012460 067702 000040      ADD      @DECPNT,R2     ;RESTORE
2021 012464 052737 000060 012520      BIS      #60,DIGIT      ;MAKE A #
2022 012472 113721 012520      MOV      DIGIT,(R1)+    ;LOAD INTO STRING
2023 012476 005237 012522      INC      DIGCNT         ;UPDATE COUNT
2024 012502 001002      BNE      3$            ;BR IF NOT DONE
2025 012504 012601      MOV      (SP)+,R1      ;RESTORE R1
2026 012506 000207      RTS      PC            ;EXIT
2027 012510 062737 000002 012524 3$:  ADD      #2,DECPNT     ;UPDATE POINTER
2028 012516 000750      BR      1$            ;BR BACK
2029
2030 012520 000000      DIGIT:  0
2031 012522 000000      DIGCNT: 0
2032 012524 012526      DECPNT: .+2
2033 012526 001750      1000.
2034 012530 000144      100.
2035 012532 000012      10.
2036 012534 000001      1.
2037
2038 012536 000      DIGT0:  .BYTE  0
2039 012537 000      DIGT1:  .BYTE  0
2040 012540 000      DIGT2:  .BYTE  0
2041 012541 000      DIGT3:  .BYTE  0
2042
2043 012542 013546      BYTWO:  MOV      @R5+,-(SP) ;SAVE ON STACK
2044 012544 104403      TYPOS   .BYTE  4,1
2045 012546 004      015665      TYPE    #MULTSP
2046 012550 104401      MOV      @R5+,-(SP)     ;SAVE ON STACK
2047 012554 013546      TYPOS   .BYTE  4,1
2048 012556 104403      TYPOS   .BYTE  4,1
2049 012560 004      001      RTS      R5          ;EXIT
2050 012562 000205
2051
2052 012564 012500      TYPSTG: MOV      (R5)+,R0      ;GET ADDRESS POINTER
2053 012566 012501      MOV      (R5)+,R1      ;LOAD # OF LOC.
2054 012570 012046      1$:  MOV      (R0)+,-(SP)    ;SAVE ON STACK
2055 012572 104403      TYPOS   .BYTE  4,1
2056 012574 004      001

```

```

2057 012576 104401 020045          TYPE      SP3MSG
2058 012602 005301          DEC        R1          ;DONE ?
2059 012604 001371          BNE        1$
2060 012606 000205          2$:      RTS        R5          ;EXIT
2061                                     ;SUBROUTINE TO GET AVERAGE OF 32 CONVERSIONS
2062
2063 012610 012537 012776          CONVRT:  MOV        (R5,+10$
2064 012614 013737 012776 013006      MOV        10$,CHANL
2065 012622 000337 012776          SWAB      10$
2066 012626 012737 000040 013000      MOV        #32,11$
2067 012634 005037 013002          CLR        ADEND
2068 012640 013737 012776 001564      MOV        10$, $TMDAT
2069
2070                                     ;*      MOV        $TMDAT, $ADCS          ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2071 012656          2$:
2072
2073                                     ;*      MOV        $ADCS, $TMDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2074 012666 005237 001564          INC        $TMDAT
2075
2076                                     ;*      MOV        $TMDAT, $ADCS          ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2077 012702          1$:
2078
2079                                     ;*      MOV        $ADCS, $TMDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2080 012712 105737 001564          TSTB      $TMDAT
2081 012716 100371          BPL        1$
2082
2083                                     ;*      MOV        $ADDBR, $TMDAT          ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2084 012730 063737 001564 013002      ADD        $TMDAT, ADEND
2085 012736 005337 013000          DEC        11$
2086 012742 001345          BNE        2$
2087 012744 006237 013002          ASR        ADEND
2088 012750 006237 013002          ASR        ADEND
2089 012754 006237 013002          ASR        ADEND
2090 012760 006237 013002          ASR        ADEND
2091 012764 006237 013002          ASR        ADEND
2092 012770 005537 013002          ADC        ADEND          ;ROUND UP
2093 012774 000205          RTS        R5
2094
2095 012776 000000          10$:      0
2096 013000 000000          11$:      0
2097 013002 000000          ADEND:    0
2098 013004 000000          FCHANL:   0
2099 013006 000000          CHANL:    0
2100 013010 000000          SPREAD:   0
2101
2102 013012 010046          COMPAR:   MOV        R0, -(SP)
2103 013014 010146          MOV        R1, -(SP)
2104 013016 013700 001124          MOV        $GDDAT, R0          ;LOAD R0
2105 013022 013701 001126          MOV        $BDDAT, R1          ;LOAD R1
2106 013026 160100          SUB        R1, R0          ;SUBTRACT
2107 013030 100001          BPL        8$
2108 013032 005400          NEG        R0
2109 013034 020037 013010          8$:      CMP        R0, SPREAD          ;MAGNITUDE OF DIFF. IN R0
2110 013040 003405          BLE        10$

```

```

2111 013042 012601
2112 013044 012600
2113 013046 062716 000002
2114 013052 000207
2115
2116 013054 012601
2117 013056 012600
2118 013060 000207
2119
2120
2121 013062 051105 047522 020122
2122 013070 047117 040440 042057
2123 013076 041440 040510 047116
2124 013104 046105 000
2125 013107 126 020103 047514
2126 013114 044507 020103 044523
2127 013122 047107 046101 044040
2128 013130 043511 020110 052517
2129 013136 050124 052125 052040
2130 013144 047517 046040 053517
2131 013152 000
2132 013153 126 020103 052123
2133 013160 052101 051525 051040
2134 013166 043505 051511 042524
2135 013174 020122 047111 042440
2136 013202 051122 051117 000
2137 013207 105 052130 051105
2138 013214 040516 020114 020101
2139 013222 047524 042040 051440
2140 013230 040524 052122 043040
2141 013236 044501 042514 000104
2142 013244 027501 020104 044504
2143 013252 043106 051105 047105
2144 013260 044524 046101 046040
2145 013266 047111 040505 044522
2146 013274 054524 042440 051122
2147 013302 051117 000
2148 013305 116 044517 042523
2149 013312 046040 053105 046105
2150 013320 042440 041530 042505
2151 013326 042504 020104 044514
2152 013334 044515 000124
2153 013340 041526 046040 043517
2154 013346 041511 051440 043511
2155 013354 040516 020114 047514
2156 013362 020127 052517 050124
2157 013370 052125 052040 047517
2158 013376 044040 043511 000110
2159 013404 027504 020101 044504
2160 013412 043106 051105 047105
2161 013420 044524 046101 046040
2162 013426 047111 040505 044522
2163 013434 054524 042440 051122
2164 013442 051117 000

```

```

9$:  MOV (SP)+,R1
     MOV (SP)+,R0
     ADD #2,(SP)
     RTS PC

10$: MOV (SP)+,R1
     MOV (SP)+,R0
     RTS PC
     .SBTTL ASCII MESSAGES AND ERROR POINTERS

EM1: .ASCIZ \ERROR ON A/D CHANNEL\

EM2: .ASCIZ \VC LOGIC SIGNAL HIGH OUTPUT TOO LOW\

EM3: .ASCIZ \VC STATUS REGISTER IN ERROR\

EM4: .ASCIZ \EXTERNAL A TO D START FAILED\

EM5: .ASCIZ \A/D DIFFERENTIAL LINEARITY ERROR\

EM6: .ASCIZ \NOISE LEVEL EXCEEDED LIMIT\

EM7: .ASCIZ \VC LOGIC SIGNAL LOW OUTPUT TOO HIGH\

EM10: .ASCIZ \D/A DIFFERENTIAL LINEARITY ERROR\

```



2219	014074	042104	020040	040440						
2220	014102	042057	041440	027110						
2221	014110	027040	020064	042514						
2222	014116	020126	047440	052125						
2223	014124	052520	000124							
2224	014130	051105	050122	020103	DH10:	.ASCIZ	\ERRPC	ARADD	STEP	NOMINAL SPREAD ACTUAL\
2225	014136	020040	051101	042101						
2226	014144	020104	020040	051440						
2227	014152	042524	020120	020040						
2228	014160	047516	044515	040516						
2229	014166	020114	050123	042522						
2230	014174	042101	020040	041501						
2231	014202	052524	046101	000						
2232	014207	105	051122	041520	DH11:	.ASCIZ	/ERRPC	ARADD	NOMINAL SPREAD	ACTUAL/
2233	014214	020040	040440	040522						
2234	014222	042104	020040	047040						
2235	014230	046517	047111	046101						
2236	014236	051440	051120	040505						
2237	014244	020104	040440	052103						
2238	014252	040525	000114							
2239	014256	005015	020111	044502	BIASST:	.ASCIZ	<15><12>/I	BIAS	=	/
2240	014264	051501	030440	000040						
2241	014272	024040			BIASDN:	.ASCII	/ (/			
2242	014274	027062			BASBT1:	.ASCII	/2./			
2243	014276	020060	044515	051103	BASBT2:	.ASCII	/0	MICROAMP	LIMIT	= /
2244	014304	040517	050115	046040						
2245	014312	046511	052111	036440						
2246	014320	040								
2247	014321	065	024460	005015	BASBT3:	.ASCIZ	/50)/<15><12>			
2248	014326	000								
2249	014327	015	005012	027504	DIFMSG:	.ASCII	<15><12><12>\D/A	DIFFERENTIAL	LINEARITY\	
2250	014334	020101	044504	043106						
2251	014342	051105	047105	044524						
2252	014350	046101	046040	047111						
2253	014356	040505	044522	054524						
2254	014364	005015	052123	050105	.ASCII	<15><12>\STEP	0/1	1/2	3/4	7/10 17/20 37/40\
2255	014372	030040	030457	020040						
2256	014400	020040	027461	020062						
2257	014406	020040	031440	032057						
2258	014414	020040	020040	027467						
2259	014422	030061	020040	030440						
2260	014430	027467	030062	020040						
2261	014436	033463	032057	060						
2262	014443	040	033467	030457	.ASCII	\	77/100	177/200	377/400	777/1000\
2263	014450	030060	030440	033467						
2264	014456	031057	030060	031440						
2265	014464	033467	032057	030060						
2266	014472	033440	033467	030457						
2267	014500	030060	060							
2268	014503	015	020012	020130	.ASCIZ	<15><12>\	X	\		
2269	014510	020040	000							
2270	014513	015	020012	020131	DIYMSG:	.ASCIZ	<15><12>/	Y	/	
2271	014520	020040	000							
2272	014523	015	024012	047516	DIEMSG:	.ASCIZ	<15><12>/	(NOMINAL	STEP	HEIGHT = 62)/<15><12>

2273	014530	044515	040516	020114	
2274	014536	052123	050105	044040	
2275	014544	044505	044107	020124	
2276	014552	020075	031066	006451	
2277	014560	000012			
2278	014562	005015	027501	020104	ADSMG: .ASCII <15><12>\A/D INTER-CHANNEL SETTling (1 LSB = 62)\
2279	014570	047111	042524	026522	
2280	014576	044103	047101	042516	
2281	014604	020114	042523	052124	
2282	014612	044514	043516	024040	
2283	014620	020061	051514	020102	
2284	014626	020075	031066	051	
2285	014633	015	050012	051517	.ASCIZ <15><12>/POSITIVE = /
2286	014640	052111	053111	020105	
2287	014646	020075	000		
2288	014651	040	047040	043505	ADEMSG: .ASCIZ / NEGATIVE = /
2289	014656	052101	053111	020105	
2290	014664	020075	000		
2291	014667	015	047012	044517	NOIMSG: .ASCII <15><12>\NOISE: RMS (1/4 LSB LIMIT = 31) PEAK (2 LSB LIMIT = 144)\
2292	014674	042523	020072	020040	
2293	014702	051040	051515	024040	
2294	014710	027461	020064	051514	
2295	014716	020102	044514	044515	
2296	014724	020124	020075	030463	
2297	014732	020051	050040	040505	
2298	014740	020113	031050	046040	
2299	014746	041123	046040	046511	
2300	014754	052111	036440	030440	
2301	014762	032064	051		
2302	014765	015	040412	042057	.ASCIZ <15><12>\A/D BIPOLAR \
2303	014772	041040	050111	046117	
2304	015000	051101	020040	020040	
2305	015006	000040			
2306	015010	005015	027501	020104	NOIMG1: .ASCIZ <15><12>\A/D UNIPOLAR \
2307	015016	047125	050111	046117	
2308	015024	051101	020040	020040	
2309	015032	000			
2310	015033	015	054012	042040	NOIMG2: .ASCIZ <15><12>\X D/A \
2311	015040	040457	020040	020040	
2312	015046	020040	020040	020040	
2313	015054	000040			
2314	015056	005015	020131	027504	NOIMG3: .ASCIZ <15><12>\Y D/A \
2315	015064	020101	020040	020040	
2316	015072	020040	020040	020040	
2317	015100	000			
2318	015101	015	005012	044514	LINMSG: .ASCII <15><12><12>/LINEARITY (1 LSB = 62)/<15><12>
2319	015106	042516	051101	052111	
2320	015114	020131	030450	046040	
2321	015122	041123	036440	033040	
2322	015130	024462	005015		
2323	015134	020040	020040	020040	.ASCII / 0 200 400 600 1000 1200 1400 1600/
2324	015142	020040	020040	020040	
2325	015150	020040	020040	020060	
2326	015156	020040	020040	031040	

2327	015164	030060	020040	020040	
2328	015172	030064	020060	020040	
2329	015200	033040	030060	020040	
2330	015206	020040	030061	030060	
2331	015214	020040	030440	030062	
2332	015222	020060	020040	032061	
2333	015230	030060	020040	030440	
2334	015236	030066	060		
2335	015241	015	054412	042040	.ASCIZ <15><12>\Y D/A VS. A/D \
2336	015246	040457	053040	027123	
2337	015254	040440	042057	020040	
2338	015262	000040			
2339	015264	005015	020130	027504	LINMG1: .ASCIZ <15><12>\X D/A VS. A/D \
2340	015272	020101	051526	020056	
2341	015300	027501	020104	020040	
2342	015306	000			
2343	015307	015	005012	027501	ADDIFM: .ASCIZ <15><12><12>\A/D DIFFERENTIAL LINEARITY:\<15><12>
2344	015314	020104	044504	043106	
2345	015322	051105	047105	044524	
2346	015330	046101	046040	047111	
2347	015336	040505	044522	054524	
2348	015344	006472	000012		
2349	015350	052123	052101	026505	ADDIF: .ASCIZ /STATE-WIDTH/<15><12>
2350	015356	044527	052104	006510	
2351	015364	000012			
2352	015366	034050	020051	045523	SKPMSG: .ASCIZ /(8) SKIPPED STATE(S)/<15><12>
2353	015374	050111	042520	020104	
2354	015402	052123	052101	024105	
2355	015410	024523	005015	000	
2356	015415	050	024470	037040	GRT2MG: .ASCIZ /(8) > 2 LSB STATE(S)/<15><12>
2357	015422	031040	046040	041123	
2358	015430	051440	040524	042524	
2359	015436	051450	006451	000012	
2360	015444	034050	020051	020074	NARMSG: .ASCIZ \ (8) < 1/2 LSB NARROW STATE(S)\<15><12>
2361	015452	027461	020062	051514	
2362	015460	020102	040516	051122	
2363	015466	053517	051440	040524	
2364	015474	042524	051450	006451	
2365	015502	000012			
2366	015504	034050	020051	020076	WIDMSG: .ASCIZ \ (8) > 1 1/2 LSB WIDE STATE(S)\<15><12>
2367	015512	020061	027461	020062	
2368	015520	051514	020102	044527	
2369	015526	042504	051440	040524	
2370	015534	042524	051450	006451	
2371	015542	000012			
2372	015544	054130	054056	020045	PERHLF: .ASCIZ \XX.X% OF STATES WITHIN 1/2 LSB (SPEC = 95%)\<15><12>
2373	015552	043117	051440	040524	
2374	015560	042524	020123	044527	
2375	015566	044124	047111	030440	
2376	015574	031057	046040	041123	
2377	015602	020040	051450	042520	
2378	015610	020103	020075	032471	
2379	015616	024445	005015	000	
2380	015623	130	027130	022530	PERQRT: .ASCIZ \XX.X% OF STATES WITHIN 1/4 LSB \<15><12>

```

2381 015630 047440 020106 052123
2382 015636 052101 051505 053440
2383 015644 052111 044510 020116
2384 015652 027461 020064 051514
2385 015660 020102 005015 000
2386 015665 040 020040 020040
2387 015672 020040 020040 020040
2388 015700 020040 020040 020040
2389 015706 020040 020040 020040
2390 015714 000040
2391
2392 015716 001116 001556 013006 DT1: .EVEN
2393 015724 001124 013010 001126 $ERRPC,ARBADD,CHANL,$GDDAT,SPREAD,$BDDAT,0
2394 015732 000000
2395 015734 001116 001556 013006 DT2: $ERRPC,ARBADD,CHANL,$GDDAT,$BDDAT,0
2396 015742 001124 001126 000000
2397 015750 001116 001556 001124 DT3: $ERRPC,ARBADD,$GDDAT,$BDDAT,0
2398 015756 001126 000000
2399 015762 001116 001556 001126 DT5: $ERRPC,ARBADD,$BDDAT,$GDDAT,0
2400 015770 001124 000000
2401 015774 001116 001556 016314 DT10: $ERRPC,ARBADD,EDGE,$GDDAT,SPREAD,$BDDAT,0
2402 016002 001124 013010 001126
2403 016010 000000
2404 016012 001116 001556 001124 DT11: $ERRPC,ARBADD,$GDDAT,SPREAD,$BDDAT,0
2405 016020 013010 001126 000000
2406 016026 000004 000203 000402 NUMBF2: 4,203,402,601,1000,1177,1376,1575,1774,BIT15
2407 016034 000601 001000 001177
2408 016042 001376 001575 001774
2409 016050 100000
2410 016052 000000 000000 000000 RSLT0: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2411 016060 000000 000000 000000
2412 016066 000000 000000 000000
2413 016074 000000 000000 000000
2414 016102 000000 000000 000000
2415 016110 000000 000000 000000 RSLT1: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2416 016116 000000 000000 000000
2417 016124 000000 000000 000000
2418 016132 000000 000000 000000
2419 016140 000000 000000 000000
2420 016146 000000 000000 000000 RSLT2: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2421 016154 000000 000000 000000
2422 016162 000000 000000 000000
2423 016170 000000 000000 000000
2424 016176 000000 000000 000000
2425 016204 000000 000000 000000 RSLT3: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2426 016212 000000 000000 000000
2427 016220 000000 000000 000000
2428 016226 000000 000000 000000
2429 016234 000000 000000 000000
2430 016242 000000
2431
2432 016244 170412 .SBTTL DIFFERENTIAL LINEARITY SUBROUTINE
2433 016246 000000 ADAC: 170412 ;BUS ADDRESS OF DAC
2434 016250 000031 DACSAV: 0
RMSMAX: 25. ;1/4 LSB

```



```

2435 016252 000310 PEKMAX: 200. ;2 LSB
2436 016254 000000 CH7RMS: 0
2437 016256 000000 CH7PEK: 0
2438 016260 000000 CH47RM: 0
2439 016262 000000 CH47PK: 0
2440 016264 000000 XDACRM: 0
2441 016266 000000 XDACPK: 0
2442 016270 000000 YDACRM: 0
2443 016272 000000 YDACPK: 0
2444 016274 000000 OFSTBX: 0
2445 016276 000000 OFSTUX: 0
2446 016300 000000 OFSLBX: 0
2447 016302 000000 OFSLUX: 0
2448 016304 000000 ADPMUX: 0
2449 016306 000000 ADNMUX: 0
2450 016310 000000 BIASC1: 0
2451 016312 000 006 SARCHN: .BYTE 0,6 ;CHANNEL # IN HIGH BYTE
2452 016314 001000 EDGE: 1000 ;CONVERTED VALUE EDGE
2453 016316 000000 CONS: 0 ;0 = RMS BIT 15 = 1 PEAK
2454 016320 000000 FINS: 0 ;2 PASS COUNTER
2455 016322 000031 LOLIM1: 31
2456 016324 000046 LOLIM2: 46
2457 016326 000113 HILIM1: 113
2458 016330 000076 HILIM2: 76
2459 016332 000000 QRTOK: 0
2460 016334 000000 NARROW: 0
2461 016336 000000 WIDE: 0
2462 016340 000062 AMEAN: 62
2463 016342 000000 TEMP: 0
2464
2465 016344 012537 020142 DIFLIN: MOV (R5)+,VCREG1 ;LOAD COARSE REGISTER ADDRESS
2466 016350 012537 020144 MOV (R5)+,VCREG2 ;LOAD FINE ADDRESS
2467 016354 012537 020146 MOV (R5)+,DIFCHN ;LOAD CHANNEL
2468 016360 010546 MOV R5,-(SP)
2469 016362 005037 001564 CLR $TMDAT
2470
2471 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
2472 016376 017737 001540 020142 MOV @VCREG1,VCREG1
2473 016404 017737 001534 020144 MOV @VCREG2,VCREG2
2474
2475 ;* MOV $TMDAT,@VCREG1 ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG1
2476
2477 ;* MOV $TMDAT,@VCREG2 ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG2
2478 016432 012701 025726 MOV #ADBUFF,%1
2479 016436 005000 CLR %0
2480 016440 005021 1$: CLR (1)+ ;CLEAR MEMORY
2481 016442 022701 032322 CMP #LAST,%1
2482 016446 001374 BNE 1$
2483
2484 016450 004737 017570 CONVR: JSR PC,CONVT
2485 016454 001410 BEQ 2$ ;BRANCH IF RESULT = 0
2486 016456 020500 CMP R5,R0 ;COMPARE RESULT AND POINT
2487 016460 100406 BMI 2$ ;BRANCH FOR RESULT < POINT
2488 016462 105265 025726 INCB ADBUFF(%5) ;INCREMENT CONTENTS OF Z
    
```

```

2489 016466 103003          BCC      2$
2490 016470 112765 000377 025726  MOVB   #377,ADBUFF(R5)
2491 016476          2$:
2492
2493          ;*      MOV      @VCREG2,$TMDAT ;/READ DEVICE REG VCREG2,PUT DATA IN $TMDAT.
2494 016506 062737 000001 001564  ADD    #1,$TMDAT ;UPDATE FINE REG.
2495          ;*
2496          ;*      MOV      $TMDAT,@VCREG2 ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG2
2497          ;*
2498          ;*      MOV      @VCREG2,$BDDAT ;/READ DEVICE REG VCREG2,PUT DATA IN $BDDAT.
2499 016534 005737 001126  TST    $BDDAT
2500 016540 001343          BNE    CONVR ;BRANCH UNTIL FINISH
2501
2502          ;*      MOV      @VCREG1,$TMDAT ;/READ DEVICE REG VCREG1.PUT DATA IN $TMDAT.
2503 016552 022737 001774 001564  CMP    #1774,$TMDAT ;TEST COARSE REG.
2504 016560 001436          BEQ    READ
2505
2506          ;*      MOV      @VCREG1,$TMDAT ;/READ DEVICE REG VCREG1.PUT DATA IN $TMDAT.
2507 016572 062737 000014 001564  ADD    #14,$TMDAT ;UPDATE COARSE REG.
2508
2509          ;*      MOV      $TMDAT,@VCREG1 ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG1
2510 016610 005037 001564  CLR    $TMDAT ;CLEAR FINE REG.
2511
2512          ;*      MOV      $TMDAT,@VCREG2 ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG2
2513 016624 004737 017570  JSR    PC,CONVT
2514 016630 062705 000004          ADD    #4,R5 ;(4+RESULT) IN R5
2515 016634 010500          MOV    %5,%0 ;LOAD POINT
2516 016636 062705 025726  ADD    #ADBUFF,%5
2517 016642 012704 000010  MOV    #10,%4
2518 016646 105025          1$:  CLRB   (R5)+ ;CLEAR 8. MEM BYTE LOCATIONS
2519 016650 005304          DEC    %4
2520 016652 001375          BNE    1$
2521 016654 000675          BR     CONVR
2522 016656 032777 010000 162254  READ:  BIT    #BIT12,@SWR
2523 016664 001402          BEQ    13$
2524 016666 104401 015307          TYPE  ADDIFM
2525 016672 005000          13$:  CLR    R0
2526 016674 012702 001776  MOV    #1776,%2
2527 016700 012703 025727  MOV    #ADBUFF+1,%3
2528 016704 012737 000001 016342  MOV    #1,TEMP
2529 016712 005037 016332  CLR    QRTOK
2530 016716 005037 017172  CLR    DIFERR
2531 016722 005037 016334  CLR    NARROW
2532 016726 005037 016336  CLR    WIDE
2533 016732 005037 017166  CLR    20$
2534 016736 005037 017170  CLR    SKIPST
2535 016742 005037 017174  CLR    EXCESS
2536 016746 111300          2$:  MOVB  (R3),R0
2537 016750 020027 000144  CMP    R0,#144 ;TEST IF STATE WIDTH > 2LSB
2538 016754 003024          BGT   1$ ;YES, BR
2539 016756 105260 031720  INCB  ABUFF4(R0) ;UPDATE STATE-WIDTH BUFFER
2540 016762 020037 016326  CMP    R0,HILIM1 ;TEST IF STATE WIDTH > 1 1/2 LSB
2541 016766 003021          BGT   3$ ;YES, BR
2542 016770 020037 016330  CMP    R0,HILIM2 ;TEST IF STATE WIDTH > 1 1/4 LSB
    
```

2543	016774	003061		BGT	4\$				;YES BR
2544	016776	020037	020302	CMP	RO,V1				;IS IT A SKIPPED STATE?
2545	017002	103416		BLO	10\$				;YES BR
2546	017004	020037	016322	CMP	RO,LOLIM1				;TEST IF STATE WIDTH < 1/2 LSB
2547	017010	002415		BLT	11\$				;YES BR
2548	017012	020037	016324	CMP	RO,LOLIM2				;TEST IF STATE WIDTH < 3/4 LSB
2549	017016	002450		BLT	4\$				;YES BR
2550	017020	005237	016332	INC	QRTOK				;STATE WIDTH BETWEEN 3/4 AND 1 1/4 LSB
2551	017024	000445		BR	4\$				
2552	017026	005237	017174	1\$: INC	EXCESS				;UPDATE > 2 LSB WIDE STATE COUNT
2553	017032	005237	016336	3\$: INC	WIDE				;UPDATE > 1 1/2 LSB WIDE STATE COUNT
2554	017036	000406		BR	12\$				
2555	017040	005237	017170	10\$: INC	SKIPST				;UPDATE SKIPPED STATE COUNT
2556	017044	105700		11\$: TSTB	RO				
2557	017046	100767		BMI	1\$				
2558	017050	005237	016334	INC	NARROW				;UPDATE < 1/2 LSB NARROW STATE COUNT
2559	017054	005237	017172	INC	DIFERR				;UPDATE OUTSIDE + OR - 1/2 LSB COUNT
2560	017060	032777	010000	162052 5\$: BIT	#BIT12, @SWR				;TEST FOR FORCED TYPEOUT
2561	017066	001424		BEQ	4\$				::BR IF NOT
2562	017070	005737	017166	TST	20\$				;TEST IF FIRST TIME ?
2563	017074	001004		BNE	6\$				;BR IF YES
2564	017076	005237	017166	INC	20\$				
2565	017102	104401	015350	TYPE	ADDIF				
2566	017106	013746	016342	6\$: MOV	TEMP,-(SP)				;SAVE CURRENT STATE
2567	017112	104403		TYPOS					
2568	017114	004	001	.BYTE	4,1				
2569	017116	104401	020054	TYPE	TYANXX				
2570	017122	042700	177400	BIC	#177400,RO				
2571	017126	010046		MOV	RO,-(SP)				
2572	017130	104403		TYPOS					
2573	017132	003	001	.BYTE	3,1				
2574	017134	104401	020040	TYPE	ACRLF				
2575	017140	005237	016342	4\$: INC	TEMP				
2576	017144	105723		TSTB	(R3)+				
2577	017146	005302		DEC	%2				
2578	017150	001276		BNE	2\$				
2579	017152	032777	010000	161760 BIT	#BIT12, @SWR				;TEST BIT 12
2580	017160	001006		BNE	SWDIST				;BR IF FORCED PRINTOUT
2581	017162	012605		MOV	(SP)+,R5				
2582	017164	000205		RTS	R5				
2583	017166	000000		20\$: 0					
2584	017170	000000		SKIPST: 0					;SKIPPED STATE COUNT
2585	017172	000000		DIFERR: 0					;OUTSIDE + OR - 1/2 LSB STATE COUNT
2586	017174	000000		EXCESS: 0					; > 2 LSB STATE COUNT
2587									
2588	017176	012703	031720	SWDIST: MOV	#ABUFF4,%3				
2589	017202	005037	016342	CLR	TEMP				
2590	017206	104401	020040	TYPE	ACRLF				
2591	017212	013746	017170	MOV	SKIPST,-(SP)				;SAVE SKIPPED STATES
2592	017216	104403		TYPOS					
2593	017220	003	000	.BYTE	3,0				
2594	017222	104401	015366	TYPE	\$KPMMSG				
2595	017226	013746	017174	MOV	EXCESS,-(SP)				;SAVE EXCESSIVE WIDE STATES
2596	017232	104403		TYPOS					

2597	017234	003	000			.BYTE	3,0	
2598	017236	104401	015415			TYPE	GRT2MG	
2599	017242	013746	016334			MOV	NARROW,-(SP)	
2600	017246	104403				TYPOS		
2601	017250	003	000			.BYTE	3,0	
2602	017252	104401	015444			TYPE	NARMSG	
2603	017256	013746	016336			MOV	WIDE,-(SP)	;SAVE WIDE STATES
2604	017262	104403				TYPOS		
2605	017264	003	000			.BYTE	3,0	
2606	017266	104401	015504			TYPE	WIDMSG	
2607	017272	012700	001776			MOV	#1022,RO	
2608	017276	163700	016336			SUB	WIDE,RO	
2609	017302	163700	016334			SUB	NARROW,RO	
2610	017306	004537	020346			JSR	RS,PRCNT	;CONVERT TO PERCENT
2611	017312	015544				PERHLF		
2612	017314	104401	015544			TYPE	,PERHLF	
2613	017320	013700	016332			MOV	QRTOK,RO	
2614	017324	004537	020346			JSR	RS,PRCNT	
2615	017330	015623				PERQRT		
2616	017332	104401	015623			TYPE	,PERQRT	
2617	017336	104401				TYPE		
2618	017340	020056				STDMSG		
2619	017342	013702	016342	6\$:		MOV	TEMP,R2	
2620	017346	006302				ASL	R2	
2621	017350	004737	012416			JSR	PC,DECPRT	;CONVERT R2 TO DEC.
2622	017354	113737	012537	017702		MOVB	DIGT1,RSV1	
2623	017362	113737	012540	017704		MOVB	DIGT2,RSV2	
2624	017370	113737	012541	017705		MOVB	DIGT3,RSV3	
2625								
2626	017376	111302				MOVB	(R3),R2	
2627	017400	004737	012416			JSR	PC,DECPRT	;CONVERT R2 TO DEC.
2628	017404	113737	012536	017707		MOVB	DIGT0,RSX1	
2629	017412	113737	012537	017710		MOVB	DIGT1,RSX2	
2630	017420	113737	012540	017711		MOVB	DIGT2,RSX3	
2631	017426	113737	012541	017712		MOVB	DIGT3,RSX4	
2632	017434	104401				TYPE		
2633	017436	017700				RSVMSG		
2634	017440	111305				MOVB	(%3),R5	
2635	017442	005305			2\$:	DEC	R5	
2636	017444	100403				BMI	1\$	
2637	017446	104401				TYPE		
2638	017450	020043				TYANX		
2639	017452	000773				BR	2\$	
2640	017454	023737	016322	016342	1\$:	CMP	LOLIM1,TEMP	
2641	017462	001413				BEQ	4\$	
2642	017464	023737	016326	016342		CMP	HILIM1,TEMP	
2643	017472	001412				BEQ	5\$	
2644	017474	023737	016340	016342		CMP	AMEAN,TEMP	
2645	017502	001010				BNE	3\$	
2646	017504	104401	017774			TYPE	,TYMEAN	
2647	017510	000405				BR	3\$	
2648	017512	104401	017716		4\$:	TYPE	,TYLOW	
2649	017516	000402				BR	3\$	
2650	017520	104401	017744		5\$:	TYPE	,TYHIGH	

```

2651 017524 005237 016342      3$:  INC      TEMP
2652 017530 105723          TSTB     (3)+
2653 017532 022737 000145 016342  CMP      #145,TEMP
2654 017540 001300          BNE      6$
2655 017542 104401          TYPE
2656 017544 020040          ACRLF
2657 017546 104401 020020      TYPE     OVERNG
2658 017552 013746 017174      MOV      EXCESS,-(SP)
2659 017556 104403          TYPOS
2660 017560 004      001      .BYTE   4,1
2661 017562 012605      MOV      (SP)+,R5
2662 017564 000205      RTS      R5
2663
2664 017566 000000      12$:  0
2665
2666 017570          CONV:
2667
2668          ;*      MOV      @VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
2669 017600 005237 001564      INC      $TMDAT
2670
2671          ;*      MOV      $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
2672 017614      2$:
2673
2674          ;*      MOV      @VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
2675 017624 105737 001126      TSTB     $BDDAT
2676 017630 100371          BPL      2$
2677 017632 005005          CLR      %5
2678
2679          ;*      MOV      DIFCHN,@ADCS ;/ PUT DATA FROM DIFCHN TO DEVICE REG ADCS
2680 017644      1$:
2681
2682          ;*      MOV      @ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2683 017654 105737 001126      TSTB     $BDDAT ;WAIT FOR DONE
2684 017660 100371          BPL      1$
2685
2686          ;*      MOV      @ADDBR,$TMDAT ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2687 017672 013705 001564      MOV      $TMDAT,R5
2688 017676 000207          RTS      PC
2689
2690 017700 015      012      RSVMSG: .BYTE 15,12
2691 017702 060      056      RSV1:  .BYTE 60,56
2692 017704 060          RSV2:  .BYTE 60
2693 017705 060      055      RSV3:  .BYTE 60,55
2694 017707 060          RSX1:  .BYTE 60
2695 017710 060          RSX2:  .BYTE 60
2696 017711 060          RSX3:  .BYTE 60
2697 017712 060      040      111      RSX4:  .BYTE 60,40,111.0
2698 017715 000
2699 017716 026455 026455 026455  TYLOW:  .ASCIZ "----- (1/2 LSB)"
2700 017724 026455 026455 020040
2701 017732 030450 031057 046040
2702 017740 041123 000051
2703 017744 026455 026455 026455  TYHIGH: .ASCIZ "----- (1 1/2 LSB)"
2704 017752 026455 026455 020040

```

2705 017760 030450 030440 031057  
 2706 017766 046040 041123 000051  
 2707 017774 026455 026455 026455  
 2708 020002 026455 026455 020040  
 2709 020010 030450 046040 041123  
 2710 020016 000051  
 2711 020020 052517 020124 043117  
 2712 020026 051040 047101 042507  
 2713 020034 026440 000040  
 2714 020040 015 012 000  
 2715 020043 052 000  
 2716 020045 040 020040 000  
 2717 020051 000 000 000  
 2718 020054 055 000  
 2719 020056 015 012  
 2720 020060 052123 052101 026505  
 2721 020066 044527 052104 020110  
 2722 020074 044504 052123 044522  
 2723 020102 052502 044524 047117  
 2724 020110 015 012  
 2725 020112 044527 052104 026510  
 2726 020120 052516 041115 051105  
 2727 020126 047440 020106 052123  
 2728 020134 052101 051505 000  
 2729 020142  
 2730 020142 000000  
 2731 020144 000000  
 2732 020146 000000  
 2733  
 2734  
 2735 020150 012701 020266  
 2736 020154 005737 020264  
 2737 020160 001017  
 2738 020162 012702 020306  
 2739 020166 112737 000062 014274  
 2740 020174 112737 000060 014276  
 2741 020202 112737 000065 014321  
 2742 020210 112737 000060 014322  
 2743 020216 000416  
 2744 020220 012702 020324  
 2745 020224 112737 000061 014274  
 2746 020232 112737 000070 014276  
 2747 020240 112737 000064 014321  
 2748 020246 112737 000064 014322  
 2749 020254 012221  
 2750 020256 005711  
 2751 020260 100375  
 2752 020262 000207  
 2753  
 2754 020264 000000  
 2755  
 2756 020266 001754  
 2757 020270 000024  
 2758 020272 000050

TYMEAN: .ASCIZ "----- (1 LSB)"

OVERNG: .ASCIZ /OUT OF RANGE - /

ACRLF: .BYTE 15,12,0  
 TYANX: .BYTE 52,0  
 SP3MSG: .ASCIZ / /  
 TYANXX: .BYTE 0,0,0  
 STDMSG: .BYTE 55,0  
 .ASCII 15,12 /STATE-WIDTH DISTRIBUTION/

.BYTE 15,12  
 .ASCIZ /WIDTH-NUMBER OF STATES/

.EVEN  
 VCREG1: 0  
 VCREG2: 0  
 DIFCHN: 0

.SBTTL PARAMETER ADJUSTMENT ROUTINE  
 WFADJ: MOV #V1754,R1 ;LOAD PARM. POINTER  
 TST WFTEST ;TEST IF OPTION TEST AREA  
 BNE 1\$ ;BR IF IT WAS  
 MOV #VARLT1,R2 ;LOAD "STANDARD" PARM. VALUES  
 MOVB #'2,BASBT1 ;LOAD BIAS MESSAGE  
 MOVB #'0,BASBT2  
 MOVB #'5,BASBT3 ;LOAD BIAS LIMIT  
 MOVB #'0,BASBT3+1  
 BR 2\$  
 1\$: MOV #VARLT2,R2 ;LOAD "OPTION TEST AREA" PARM. VALUES  
 MOVB #'1,BASBT1 ;LOAD BIAS MESSAGE  
 MOVB #'8,BASBT2  
 MOVB #'4,BASBT3 ;LOAD BIAS LIMIT  
 MOVB #'4,BASBT3+1  
 2\$: MOV (R2)+,(R1)+ ;LOAD INTO PARM. LIST  
 TST (R1) ;TEST FOR LAST  
 BPL 2\$ ;BR IF NOT  
 RTS PC ;EXIT

WFTEST: 0

V1754: 1754 ;1760 IF OPTION TEST AREA SELECTED  
 V24: 24 ;20  
 V50: 50 ;40

```

2759 020274 000024 VA24: 24 ;22 " " " "
2760 020276 000062 V62: 62 ;45 " " " "
2761 020300 000077 V77: 77 ;45 " " " "
2762 020302 000001 V1: 1 ;15 " " " "
2763 020304 100000 BIT15 ;TERM.

```

```

2765 020306 001754 VARLT1: 1754
2766 020310 000024 24
2767 020312 000050 50
2768 020314 000024 24
2769 020316 000062 62
2770 020320 000077 77
2771 020322 000001 1

```

```

2773 020324 001760 VARLT2: 1760
2774 020326 000020 20
2775 020330 000040 40
2776 020332 000022 22
2777 020334 000045 45
2778 020336 000045 45
2779 020340 000015 15

```

.SBTTL SUBROUTINE TO CONVERT RO TO DECIMAL PERCENTAGE OF 1022.

```

2783 020342 000000 MSGPNT: 0
2784 020344 000000 PCT: 0
2785 020346 012537 020342 PRCNT: MOV (R5)+,MSGPNT ;GET MESSAGE POINTER
2786 020352 010037 020344 MOV RO,PCT ;
2787 020356 006200 ASR RO
2788 020360 006200 ASR RO
2789 020362 006200 ASR RO
2790 020364 006200 ASR RO
2791 020366 006200 ASR RO
2792 020370 006200 ASR RO
2793 020372 160037 020344 SUB RO,PCT
2794 020376 006200 ASR RO
2795 020400 160037 020344 SUB RO,PCT ;PCT CONTAINS OCTAL %
2796 020404 013702 020344 MOV PCT,R2 ;LOAD R2 WITH PERCENTAGE
2797 020410 004737 012416 JSR PC,DECPRT ;CONVERT TO DEC.
2798 020414 013700 020342 MOV MSGPNT,RO ;LOAD MSG. POINTER
2799 020420 022737 001750 020344 CMP #1000.,PCT ;TEST FOR 100 %
2800 020426 001411 BEQ 1$ ;BR IF 100 %
2801 020430 113720 012537 MOV8 DIGT1,(RO)+ ;LOAD A DIGIT
2802 020434 113720 012540 MOV8 DIGT2,(RO)+
2803 020440 112720 000056 MOV8 #56,(RO)+ ;LOAD "."
2804 020444 113720 012541 MOV8 DIGT3,(RO)+ ;LOAD 1/10 % DIGIT
2805 020450 000205 RTS R5 ;EXIT
2806
2807 020452 112720 000040 1$: MOV8 #40,(RO)+ ;LOAD "SPACE"
2808 020456 112720 000061 MOV8 #'1,(RO)+ ;LOAD '1' 100%
2809 020462 112720 000060 MOV8 #'0,(RO)+
2810 020466 112720 000060 MOV8 #'0,(RO)+
2811 020472 000205 RTS R5 ;EXIT
2812 .SBTTL SCOPE HANDLER ROUTINE

```

```

2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826 020474
2827 020474 104407
2828 020476 104407
2829 020500 032777 040000 160432 1$:
2830 020506 001114
2831
2832 020510 000416
2833
2834 020512 013746 000004
2835 020516 012737 020536 000004
2836 020524 005737 177060
2837 020530 012637 000004
2838 020534 000463
2839 020536 022626 5$:
2840 020540 012637 000004
2841 020544 000423
2842 020546
2843 020546 032777 000400 160364 6$:
2844 020554 001404
2845 020556 127737 160356 001102
2846 020564 001465
2847 020566 105737 001103 2$:
2848 020572 001421
2849 020574 123737 001115 001103
2850 020602 101015
2851 020604 032777 001000 160326
2852 020612 001404
2853 020614 013737 001110 001106 7$:
2854 020622 000446
2855 020624 105037 001103 4$:
2856 020630 005037 001166
2857 020634 000415
2858 020636 032777 004000 160274 3$:
2859 020644 001011
2860 020646 005737 001210
2861 020652 001406
2862 020654 005237 001104
2863 020660 023737 001166 001104
2864 020666 002024
2865 020670 012737 000001 001104 1$:
2866 020676 013737 020754 001166

```

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1      LOOP ON TEST
*SW11=1      INHIBIT ITERATIONS
*SW09=1      LOOP ON ERROR
*SW08=1      LOOP ON TEST IN SWR<7:0>
*CALL
*          SCOPE          ;;SCOPE=IOT

$SCOPE:
          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
          CKSWR
          BIT            #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
          BNE           $OVER              ;;YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR           6$
          MOV           @#ERRVEC,-(SP)      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
          MOV           #5,@#ERRVEC        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
          TST           @#177060          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
          MOV           (SP)+,@#ERRVEC     ;;SET FOR TIMEOUT
          BR            $$VLAD            ;;TIME OUT ON XOR?
          BR            $S               ;;RESTORE THE ERROR VECTOR
          CMP           (SP)+,(SP)+       ;;GO TO THE NEXT TEST
          MOV           (SP)+,@#ERRVEC     ;;CLEAR THE STACK AFTER A TIME OUT
          BR            7$               ;;RESTORE THE ERROR VECTOR
          BR            6$               ;;LOOP ON THE PRESENT TEST
          BIT            #BIT08,@SWR      ;;LOOP ON SPEC. TEST?
          BEQ           2$               ;;BR IF NO
          CMPB         @SWR,$TSTNM        ;;ON THE RIGHT TEST? SWR<7:0>
          BEQ           $OVER            ;;BR IF YES
          TSTB         $ERFLG            ;;HAS AN ERROR OCCURRED?
          BEQ           3$               ;;BR IF NO
          CMPB         $ERMAX,$ERFLG     ;;MAX. ERRORS FOR THIS TEST OCCURRED?
          BHI           3$               ;;BR IF NO
          BIT            #BIT09,@SWR      ;;LOOP ON ERROR?
          BEQ           4$               ;;BR IF NO
          MOV           $LPERR,$LPADR     ;;SET LOOP ADDRESS TO LAST SCOPE
          BR            $OVER
          CLRB         $ERFLG            ;;ZERO THE ERROR FLAG
          CLR          $TIMES            ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
          BR            1$               ;;ESCAPE TO THE NEXT TEST
          BIT            #BIT11,@SWR      ;;INHIBIT ITERATIONS?
          BNE           1$               ;;BR IF YES
          TST          $PASS             ;;IF FIRST PASS OF PROGRAM
          BEQ           1$               ;;INHIBIT ITERATIONS
          INC          $ICNT             ;;INCREMENT ITERATION COUNT
          CMP          $TIMES,$ICNT       ;;CHECK THE NUMBER OF ITERATIONS MADE
          BGE          $OVER            ;;BR IF MORE ITERATION REQUIRED
          MOV          #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
          MOV          $MXCNT,$TIMES     ;;SET NUMBER OF ITERATIONS TO DO

```



```

2867 020704 105237 001102    $SVLAD: INCB  $STSINM      ;; COUNT TEST NUMBERS
2868 020710 113737 001102 001206  MOV  $STSTNM,$STESTN    ;; SET TEST NUMBER IN APT MAILBOX
2869 020716 011637 001106    MOV  (SP), $LPADR      ;; SAVE SCOPE LOOP ADDRESS
2870 020722 011637 001110    MOV  (SP), $LPERR     ;; SAVE ERROR LOOP ADDRESS
2871 020726 005037 001170    CLR  $ESCAPE         ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2872 020732 112737 000001 001115  MOV  #1,$ERMAX       ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2873 020740 013777 001102 160174  $OVER: MOV  $STSTNM,$DISPLAY ;; DISPLAY TEST NUMBER
2874 020746 013716 001106    MOV  $LPADR,(SP)     ;; FUDGE RETURN ADDRESS
2875 020752 000002    RTI                  ;; FIXES PS
2876 020754 000012  $MXCNT: 10.         ;; MAX. NUMBER OF ITERATIONS
2877    .SBTTL  ERROR HANDLER ROUTINE
2878
2879    ;*****
2880    ;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2881    ;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2882    ;AND GO TO $ERRTYP ON ERROR
2883    ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2884    ;$SW15=1      HALT ON ERROR
2885    ;$SW13=1      INHIBIT ERROR TYPEOUTS
2886    ;$SW10=1      BELL ON ERROR
2887    ;$SW09=1      LOOP ON ERROR
2888    ;$CALL
2889    ;*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2890
2891 020756  $ERROR:
2892 020756 104407 001103 7$:  CKSWR      ;; TEST FOR CHANGE IN SOFT-SWR
2893 020760 105237 001103  INC  $ERFLG    ;; SET THE ERROR FLAG
2894 020764 001775 001102 160146  BEQ  7$      ;; DON'T LET THE FLAG GO TO ZERO
2895 020766 013777 001102 160136  MOV  $STSTNM,$DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
2896 020774 032777 002000 160136  BIT  #BIT10,$SWR    ;; BELL ON ERROR?
2897 021002 001402 001172 1$:  BEQ  1$      ;; NO - SKIP
2898 021004 104401 001112  INC  $BELL     ;; RING BELL
2899 021010 005237 001112  INC  $ERTTL    ;; COUNT THE NUMBER OF ERRORS
2900 021014 011637 001116  MOV  (SP), $ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
2901 021020 162737 000002 001116  SUB  #2,$ERRPC
2902 021026 117737 160064 001114  MOVB $ERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
2903 021034 032777 020000 160076  BIT  #BIT13,$SWR    ;; SKIP TYPEOUT IF SET
2904 021042 001004 022034 20$:  BNE  20$     ;; SKIP TYPEOUTS
2905 021044 004737 001177  JSR  PC,$ERRTYP    ;; GO TO USER ERROR ROUTINE
2906 021050 104401 001177  TYPE , $CRLF
2907 021054 122737 000001 001222 20$:  CMPB #APTENV,$ENV  ;; RUNNING IN APT MODE
2908 021062 001007 001114 021076  BNE  2$      ;; NO, SKIP APT ERROR REPORT
2909 021064 113737 001114 021076  MOVB $ITEMB,21$    ;; SET ITEM NUMBER AS ERROR NUMBER
2910 021072 004737 023136  JSR  PC,$ATY4     ;; REPORT FATAL ERROR TO APT
2911 021076 000 21$:  .BYTE 0
2912 021076 000  .BYTE 0
2913 021077 000 22$:  .BYTE 0
2914 021100 000777 160032 22$:  BR  22$      ;; APT ERROR LOOP
2915 021102 005777 160032 2$:  TST  $SWR     ;; HALT ON ERROR
2916 021106 100002 3$:  BPL  3$      ;; SKIP IF CONTINUE
2917 021110 000000 3$:  HALT        ;; HALT ON ERROR!
2918 021112 104407 001000 160016 3$:  CKSWR      ;; TEST FOR CHANGE IN SOFT SWR
2919 021114 032777 001000 160016  BIT  #BIT09,$SWR    ;; LOOP ON ERROR SWITCH SET?
2920 021122 001402 4$:  BEQ  4$      ;; BR IF NO

```

```

2921 021124 013716 001110      MOV      $LPERR, (SP)      ;; FUDGE RETURN FOR LOOPING
2922 021130 005737 001170      4$:     TST      $ESCAPE     ;; CHECK FOR AN ESCAPE ADDRESS
2923 021134 001402                BEQ      5$                ;; BR IF NONE
2924 021136 013716 001170      MOV      $ESCAPE, (SP)    ;; FUDGE RETURN ADDRESS FOR ESCAPE
2925 021142                5$:     CMP      #SENDAD, @#42    ;; ACT-11 AUTO-ACCEPT?
2926 021142 022737 011436 000042  BNE      6$                ;; BRANCH IF NO
2927 021150 001001                HALT                    ;; YES
2928 021152 000000                6$:     RTI                    ;; RETURN
2929 021154                .SBTTL  TTY INPUT ROUTINE
2930 021154 000002
2931
2932
2933 ;:*****
2934 .ENABL  LSB
2935
2936 ;:*****
2937 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2938 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2939 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2940 *WHEN OPERATING IN TTY FLAG MODE.
2941 021156 022737 000176 001140  $CKSWR:  CMP      #SWREG, SWR    ;; IS THE SOFT-SWR SELECTED?
2942 021164 001074                BNE      15$              ;; BRANCH IF NO
2943 021166 105777 157752                TSTB     @STKS            ;; CHAR THERE?
2944 021172 100071                BPL      15$              ;; IF NO, DON'T WAIT AROUND
2945 021174 117746 157746                MOV      @STKB, -(SP)     ;; SAVE THE CHAR
2946 021200 042716 177600                BIC      #↑C177, (SP)    ;; STRIP-OFF THE ASCII
2947 021204 022726 000007                CMP      #7, (SP)+       ;; IS IT A CONTROL G?
2948 021210 001062                BNE      15$              ;; NO RETURN TO USER
2949 021212 123727 001134 000001  CMP      $AUTOB, #1      ;; ARE WE RUNNING IN AUTO-MODE?
2950 021220 001456                BEQ      15$              ;; BRANCH IF YES
2951
2952 021222 104401 021703                $GTSWR:  TYPE     , $CNTLG    ;; ECHO THE CONTROL-G (↑G)
2953 021226 104401 021710                TYPE     , $MSWR        ;; TYPE CURRENT CONTENTS
2954 021232 013746 000176                MOV      $WREG, -(SP)   ;; SAVE SWREG FOR TYPEOUT
2955 021236 104402                TYPOC    ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2956 021240 104401 021721                TYPE     , $MNEW        ;; PROMPT FOR NEW SWR
2957 021244 005046                19$:    CLR      -(SP)      ;; CLEAR COUNTER
2958 021246 005046                CLR      -(SP)          ;; THE NEW SWR
2959 021250 105777 157670                7$:     TSTB     @STKS            ;; CHAR THERE?
2960 021254 100375                BPL      7$              ;; IF NOT TRY AGAIN
2961
2962 021256 117746 157664                MOV      @STKB, -(SP)   ;; PICK UP CHAR
2963 021262 042716 177600                BIC      #↑C177, (SP)   ;; MAKE IT 7-BIT ASCII
2964
2965
2966
2967 021266 021627 000025                9$:     CMP      (SP), #25    ;; IS IT A CONTROL-U?
2968 021272 001005                BNE      10$             ;; BRANCH IF NOT
2969 021274 104401 021676                TYPE     , $CNTLU       ;; YES, ECHO CONTROL-U (↑U)
2970 021300 062706 000006                20$:    ADD      #6, SP      ;; IGNORE PREVIOUS INPUT
2971 021304 000757                BR       19$            ;; LET'S TRY IT AGAIN
2972
2973
2974 021306 021627 000015                10$:    CMP      (SP), #15    ;; IS IT A <CR>?
    
```

```

2975 021312 001022 BNE 16$ ;: BRANCH IF NO
2976 021314 005766 000004 TST 4(SP) ;: YES, IS IT THE FIRST CHAR?
2977 021320 001403 BEQ 11$ ;: BRANCH IF YES
2978 021322 016677 000002 157610 MOV 2(SP), @SWR ;: SAVE NEW SWR
2979 021330 062706 000006 11$: ADD #6, SP ;: CLEAR UP STACK
2980 021334 104401 001177 14$: TYPE $CRLF ;: ECHO <CR> AND <LF>
2981 021340 123727 001135 000001 CMPB $INTAG, #1 ;: RE-ENABLE TTY KBD INTERRUPTS?
2982 021346 001003 BNE 15$ ;: BRANCH IF NOT
2983 021350 012777 000100 157566 MOV #100, @STKS ;: RE-ENABLE TTY KBD INTERRUPTS
2984 021356 000002 15$: RTI ;: RETURN
2985 021360 004737 023050 16$: JSR PC, $TYPEC ;: ECHO CHAR
2986 021364 021627 000060 CMP (SP), #60 ;: CHAR < 0?
2987 021370 002420 BLT 18$ ;: BRANCH IF YES
2988 021372 021627 000067 CMP (SP), #67 ;: CHAR > 7?
2989 021376 003015 BGT 18$ ;: BRANCH IF YES
2990 021400 042726 000060 BIC #60, (SP)+ ;: STRIP-OFF ASCII
2991 021404 005766 000002 TST 2(SP) ;: IS THIS THE FIRST CHAR
2992 021410 001403 BEQ 17$ ;: BRANCH IF YES
2993 021412 006316 ASL (SP) ;: NO, SHIFT PRESENT
2994 021414 006316 ASL (SP) ;: CHAR OVER TO MAKE
2995 021416 006316 ASL (SP) ;: ROOM FOR NEW ONE.
2996 021420 005266 000002 17$: INC 2(SP) ;: KEEP COUNT OF CHAR
2997 021424 056616 177776 BIS -2(SP), (SP) ;: SET IN NEW CHAR
2998 021430 000707 BR 7$ ;: GET THE NEXT ONE
2999 021432 104401 001176 18$: TYPE $QUES ;: TYPE ?<CR><LF>
3000 021436 000720 BR 20$ ;: SIMULATE CONTROL-U
3001 .DSABL LSB
3002
3003
3004 ;: *****
3005 ;: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3006 ;: *CALL:
3007 ;: * RDCHR ;: INPUT A SINGLE CHARACTER FROM THE TTY
3008 ;: * RETURN HERE ;: CHARACTER IS ON THE STACK
3009 ;: * ;: WITH PARITY BIT STRIPPED OFF
3010 ;:
3011
3012 $RDCHR: MOV (SP), -(SP) ;: PUSH DOWN THE PC
3013 021440 011646 000004 000002 MOV 4(SP), 2(SP) ;: SAVE THE PS
3014 021442 016666 157470 1$: TSTB @STKS ;: WAIT FOR
3015 021450 105777 157464 000004 BPL 1$ ;: A CHARACTER
3016 021454 100375 177600 000004 MOVB @STKB, 4(SP) ;: READ THE TTY
3017 021464 042766 000004 000004 BIC #1C<177>, 4(SP) ;: GET RID OF JUNK IF ANY
3018 021472 026627 000004 000023 CMP 4(SP), #23 ;: IS IT A CONTROL-S?
3019 021500 001013 BNE 3$ ;: BRANCH IF NO
3020 021502 105777 157436 2$: TSTB @STKS ;: WAIT FOR A CHARACTER
3021 021506 100375 157432 2$ BPL 2$ ;: LOOP UNTIL ITS THERE
3022 021510 117746 177600 000021 MOVB @STKB, -(SP) ;: GET CHARACTER
3023 021514 042716 000021 BIC #1C177, (SP) ;: MAKE IT 7-BIT ASCII
3024 021520 022627 000021 CMP (SP)+, #21 ;: IS IT A CONTROL-Q?
3025 021524 001366 BNE 2$ ;: IF NOT DISCARD IT
3026 021526 000750 BR 1$ ;: YES, RESUME
3027 021530 026627 000004 000140 3$: CMP 4(SP), #140 ;: IS IT UPPER CASE?
3028 021536 002407 BLT 4$ ;: BRANCH IF YES

```

```

3029 021540 026627 000004 000175      CMP      4(SP),#175      ;; IS IT A SPECIAL CHAR?
3030 021546 003003                    BGT      4$             ;; BRANCH IF YES
3031 021550 042766 000040 000004      BIC      #40,4(SP)      ;; MAKE IT UPPER CASE
3032 021556 000002                    RTI                    ;; GO BACK TO USER
3033                                     4$:
3034                                     ;; *****
3035                                     ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3036                                     ;; CALL:
3037                                     ;; *
3038                                     ;; *   RDLIN                ;; INPUT A STRING FROM THE TTY
3039                                     ;; *   RETURN HERE        ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3040                                     ;; *                       ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
3040 021560 010346                    $RDLIN: MOV      R3, -(SP)      ;; SAVE R3
3041 021562 012703 021666              1$: MOV      #STTYIN,R3      ;; GET ADDRESS
3042 021566 022703 021676              2$: CMP      #STTYIN+8.,R3    ;; BUFFER FULL?
3043 021572 101405                    BLOS     4$             ;; BR IF YES
3044 021574 104410                    RDCHR                    ;; GO READ ONE CHARACTER FROM THE TTY
3045 021576 112613                    MOVB    (SP)+,(R3)      ;; GET CHARACTER
3046 021600 122713 000177              10$: CMPB   #177,(R3)      ;; IS IT A RUBOUT
3047 021604 001003                    BNE     3$             ;; SKIP IF NOT
3048 021606 104401 001176              4$: TYPE   $QUES        ;; TYPE A '?'
3049 021612 000763                    BR      1$             ;; CLEAR THE BUFFER AND LOOP
3050 021614 111337 021664              3$: MOVB   (R3),9$      ;; ECHO THE CHARACTER
3051 021620 104401 021664              TYPE   9$
3052 021624 122723 000015              CMPB   #15,(R3)+      ;; CHECK FOR RETURN
3053 021630 001356                    BNE     2$             ;; LOOP IF NOT RETURN
3054 021632 105063 177777              CLRB   -1(R3)         ;; CLEAR RETURN (THE 15)
3055 021636 104401 001200              TYPE   $LF           ;; TYPE A LINE FEED
3056 021642 012603                    MOV     (SP)+,R3      ;; RESTORE R3
3057 021644 011646                    MOV     (SP),-(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3058 021646 016666 000004 000002      MOV     4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
3059 021654 012766 021666 000004      MOV     #STTYIN,4(SP)
3060 021662 000002                    RTI                    ;; RETURN
3061 021664 000                    9$: .BYTE   0            ;; STORAGE FOR ASCII CHAR. TO TYPE
3062 021665 000                    .BYTE   0            ;; TERMINATOR
3063 021666 000010                    $TTYIN: .BLKB   8.    ;; RESERVE 8 BYTES FOR TTY INPUT
3064 021676 052536 005015 000          $CNTLU: .ASCIZ  /↑U/<15><12>  ;; CONTROL "U"
3065 021703 136 006507 000012          $CNTLG: .ASCIZ  /↑G/<15><12>  ;; CONTROL "G"
3066 021710 005015 053523 020122          $MSWR: .ASCIZ  <15><12>/SWR = /
3067 021716 020075 000
3068 021721 040 047040 053505          $MNEW: .ASCIZ  / NEW = /
3069 021726 036440 000040
3070                                     .SBTTL READ AN OCTAL NUMBER FROM THE TTY
3071
3072                                     ;; *****
3073                                     ;; THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
3074                                     ;; CHANGE IT TO BINARY.
3075                                     ;; CALL:
3076                                     ;; *
3077                                     ;; *   RDOCT                ;; READ AN OCTAL NUMBER
3078                                     ;; *   RETURN HERE        ;; LOW ORDER BITS ARE ON TOP OF THE STACK
3079                                     ;; *                       ;; HIGH ORDER BITS ARE IN $HIOCT
3080 021732 011646                    $RDOCT: MOV     (SP),-(SP)  ;; PROVIDE SPACE FOR THE
3081 021734 016666 000004 000002      MOV     4(SP),2(SP)   ;; INPUT NUMBER
3082 021742 010046                    MOV     R0, -(SP)    ;; PUSH R0 ON STACK

```

```

3083 021744 010146          MOV     R1,-(SP)      ;; PUSH R1 ON STACK
3084 021746 010246          MOV     R2,-(SP)      ;; PUSH R2 ON STACK
3085 021750 104411          1$:   RDLIN          ;; READ AN ASCII LINE
3086 021752 012600          MOV     (SP)+,R0      ;; GET ADDRESS OF 1ST CHARACTER
3087 021754 005001          CLR     R1            ;; CLEAR DATA WORD
3088 021756 005002          CLR     R2
3089 021760 112046          2$:   MOVB     (R0)+,-(SP) ;; PICKUP THIS CHARACTER
3090 021762 001412          BEQ     3$            ;; IF ZERO GET OUT
3091 021764 006301          ASL     R1            ;; *2
3092 021766 006102          ROL     R2
3093 021770 006301          ASL     R1            ;; *4
3094 021772 006102          ROL     R2
3095 021774 006301          ASL     R1            ;; *8
3096 021776 006102          ROL     R2
3097 022000 042716 177770     BIC     #1C7,(SP)     ;; STRIP THE ASCII JUNK
3098 022004 062601          ADD     (SP)+,R1     ;; ADD IN THIS DIGIT
3099 022006 000764          BR      2$           ;; LOOP
3100 022010 005726          3$:   TST     (SP)+     ;; CLEAN TERMINATOR FROM STACK
3101 022012 010166 000012     MOV     R1,12(SP)    ;; SAVE THE RESULT
3102 022016 010237 022032     MOV     R2,$HIOCT
3103 022022 012602          MOV     (SP)+,R2     ;; POP STACK INTO R2
3104 022024 012601          MOV     (SP)+,R1     ;; POP STACK INTO R1
3105 022026 012600          MOV     (SP)+,R0     ;; POP STACK INTO R0
3106 022030 000002          RTI                    ;; RETURN
3107 022032 000000          $HIOCT: .WORD 0      ;; HIGH ORDER BITS GO HERE
3108
3109          .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
3110
3111          ;; *****
3112          ;; THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
3113          ;; ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" ($ERRTB),
3114          ;; AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
3115
3116          $ERRTYP:
3117 022034 104401 001177     TYPE   $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
3118 022040 010046          MOV     R0,-(SP)     ;; SAVE R0
3119 022042 005000          CLR     R0            ;; PICKUP THE ITEM INDEX
3120 022044 153700 001114     BISB   2*$ITEMB,R0
3121 022050 001004          BNE     1$           ;; IF ITEM NUMBER IS ZERO, JUST
3122                                     ;; TYPE THE PC OF THE ERROR
3123 022052 013746 001116     MOV     $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
3124                                     ;; ERROR ADDRESS
3125 022056 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
3126 022060 000426          BR      6$           ;; GET OUT
3127 022062 005300          1$:   DEC     R0            ;; ADJUST THE INDEX SO THAT IT WILL
3128 022064 006300          ASL     R0            ;; WORK FOR THE ERROR TABLE
3129 022066 006300          ASL     R0
3130 022070 006300          ASL     R0
3131 022072 062700 001326     ADD     #$ERRTB,R0   ;; FORM TABLE POINTER
3132 022076 012037 022106     MOV     (R0)+,2$     ;; PICKUP "ERROR MESSAGE" POINTER
3133 022102 001404          BEQ     3$           ;; SKIP TYPEOUT IF NO POINTER
3134 022104 104401          TYPE          ;; TYPE THE "ERROR MESSAGE"
3135 022106 000000          2$:   .WORD 0          ;; "ERROR MESSAGE" POINTER GOES HERE
3136 022110 104401 001177     TYPE   ,CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"

```

```

3137 022114 012037 022124 3$: MOV (RO)+,4$ ;: PICKUP "DATA HEADER" POINTER
3138 022120 001404 BEQ 5$ ;: SKIP TYPEOUT IF 0
3139 022122 104401 TYPE ;: TYPE THE "DATA HEADER"
3140 022124 000000 4$: .WORD 0 ;: "DATA HEADER" POINTER GOES HERE
3141 022126 104401 001177 TYPE $CRLF ;: "CARRIAGE RETURN" & "LINE FEED"
3142 022132 011000 5$: MOV (PO),RO ;: PICKUP "DATA TABLE" POINTER
3143 022134 001004 BNE 7$ ;: GO TYPE THE DATA
3144 022136 012600 6$: MOV (SP)+,RO ;: RESTORE RO
3145 022140 104401 001177 TYPE $CRLF ;: "CARRIAGE RETURN" & "LINE FEED"
3146 022144 000207 RTS PC ;: RETURN
3147 022146 7$:
3148 022146 013046 MOV @ (RO)+,-(SP) ;: SAVE @ (RO)+ FOR TYPEOUT
3149 022150 104402 TYPOC ;: GO TYPE--OCTAL ASCII(ALL DIGITS)
3150 022152 005710 TST (RO) ;: IS THERE ANOTHER NUMBER?
3151 022154 001770 BEQ 6$ ;: BR IF NO
3152 022156 104401 022164 TYPE 8$ ;: TYPE TWO(2) SPACES
3153 022162 000771 BR 7$ ;: LOOP
3154 022164 020040 000 8$: .ASCIZ / / ;: TWO(2) SPACES
3155 022170 .EVEN

```

.SBTTL POWER DOWN AND UP ROUTINES

```

3156
3157
3158
3159
3160 ;:*****
3161 022170 012737 022334 000024 $PWRDN: MOV #SILLUP,@#PWRVEC ;: SET FOR FAST UP
3162 022176 012737 000340 000026 MOV #340,@#PWRVEC+2 ;: PRIO:7
3163 022204 010046 MOV RO,-(SP) ;: PUSH RO ON STACK
3164 022206 010146 MOV R1,-(SP) ;: PUSH R1 ON STACK
3165 022210 010246 MOV R2,-(SP) ;: PUSH R2 ON STACK
3166 022212 010346 MOV R3,-(SP) ;: PUSH R3 ON STACK
3167 022214 010446 MOV R4,-(SP) ;: PUSH R4 ON STACK
3168 022216 010546 MOV R5,-(SP) ;: PUSH R5 ON STACK
3169 022220 017746 156714 MOV @SWR,-(SP) ;: PUSH @SWR ON STACK
3170 022224 010637 022340 MOV SP,$SAVR6 ;: SAVE SP
3171 022230 012737 022242 000024 MOV #SPWRUP,@#PWRVEC ;: SET UP VECTOR
3172 022236 000000 HALT
3173 022240 000776 BR -2 ;: HANG UP
3174
3175 ;:*****
3176
3177 022242 012737 022334 000024 $PWRUP: MOV #SILLUP,@#PWRVEC ;: SET FOR FAST DOWN
3178 022250 013706 022340 MOV $SAVR6,SP ;: GET SP
3179 022254 005037 022340 CLR $SAVR6 ;: WAIT LOOP FOR THE TTY
3180 022260 005237 022340 1$: INC $SAVR6 ;: WAIT FOR THE INC
3181 022264 001375 BNE 1$ ;: OF WORD
3182 022266 012677 156646 MOV (SP)+,@SWR ;: POP STACK INTO @SWR
3183 022272 012605 MOV (SP)+,R5 ;: POP STACK INTO R5
3184 022274 012604 MOV (SP)+,R4 ;: POP STACK INTO R4
3185 022276 012603 MOV (SP)+,R3 ;: POP STACK INTO R3
3186 022300 012602 MOV (SP)+,R2 ;: POP STACK INTO R2
3187 022302 012601 MOV (SP)+,R1 ;: POP STACK INTO R1
3188 022304 012600 MOV (SP)+,RO ;: POP STACK INTO RO
3189 022306 012737 022170 000024 MOV #SPWRDN,@#PWRVEC ;: SET UP THE POWER DOWN VECTOR
3190 022314 012737 000340 000026 MOV #340,@#PWRVEC+2 ;: PRIO:7

```

```

3191 022322 104401
3192 022324 022342
3193 022326 012716
3194 022330 001612
3195 022332 000002
3196 022334 000000
3197 022336 000776
3198 022340 000000
3199 022342 005015 042522 052123
3200 022350 051101 044524 043516
3201 022356 040440 052106 051105
3202 022364 040440 050040 053517
3203 022372 051105 043040 044501
3204 022400 052514 042522 005015
3205 022406 000012
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233 022410 017646 000000
3234 022414 116637 000001 022633
3235 022422 112637 022635
3236 022426 062716 000002
3237 022432 000406
3238 022434 112737 000001 022633
3239 022442 112737 000006 022635
3240 022450 112737 000005 022632
3241 022456 010346
3242 022460 010446
3243 022462 010546
3244 022464 113704 022635

TYPE
$PWRMG: .WORD PWRMSG
MOV (PC)+,(SP)
$PWRAD: .WORD BEGIN
RTI
$ILLUP: HALT
BR .-2
$SAVR6: 0
PWRMSG: .ASCIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>

;;REPORT THE POWER FAILURE
;;POWER FAIL MESSAGE POINTER
;;RESTART AT BEGIN
;;RESTART ADDRESS

;;THE POWER UP SEQUENCE WAS STARTED
;;BEFORE THE POWER DOWN WAS COMPLETE
;;PUT THE SP HERE

.EVEN
.SBTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS   ;;CALL FOR TYPEOUT
*   .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC   ;;CALL FOR TYPEOUT
*$TYPOS: MOV     @ (SP),-(SP) ;;PICKUP THE MODE
MOV     1(SP), $OFILL ;;LOAD ZERO FILL SWITCH
MOV     (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
ADD     #2,(SP) ;;ADJUST RETURN ADDRESS
BR     $TYPON
*$TYPOC: MOV     #1, $OFILL ;;SET THE ZERO FILL SWITCH
MOV     #6, $OMODE+1 ;;SET FOR SIX(6) DIGITS
*$TYPON: MOV     #5, $OCNT ;;SET THE ITERATION COUNT
MOV     R3,-(SP) ;;SAVE R3
MOV     R4,-(SP) ;;SAVE R4
MOV     R5,-(SP) ;;SAVE R5
MOV     $OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE

```

```

3245 022470 005404          NEG      R4
3246 022472 062704 000006  ADD      #6,R4          ;; SUBTRACT IT FOR MAX. ALLOWED
3247 022476 110437 022634  MOVB    R4,$OMODE     ;; SAVE IT FOR USE
3248 022502 113704 022633  MOVB    $OFILL,R4    ;; GET THE ZERO FILL SWITCH
3249 022506 016605 000012  MOV     12(SP),R5    ;; PICKUP THE INPUT NUMBER
3250 022512 005003          CLR     R3           ;; CLEAR THE OUTPUT WORD
3251 022514 006105          1$:    ROL     R5           ;; ROTATE MSB INTO "C"
3252 022516 000404          BR     3$           ;; GO DO MSB
3253 022520 006105          2$:    ROL     R5           ;; FORM THIS DIGIT
3254 022522 006105          ROL     R5
3255 022524 006105          ROL     R5
3256 022526 010503          MOV     R5,R3
3257 022530 006103          3$:    ROL     R3           ;; GET LSB OF THIS DIGIT
3258 022532 105337 022634  DECB    $OMODE       ;; TYPE THIS DIGIT?
3259 022536 100016          BPL    7$           ;; BR IF NO
3260 022540 042703 177770  BIC     #177770,R3   ;; GET RID OF JUNK
3261 022544 001002          BNE    4$           ;; TEST FOR 0
3262 022546 005704          TST    R4           ;; SUPPRESS THIS 0?
3263 022550 001403          BEQ    5$           ;; BR IF YES
3264 022552 005204          4$:    INC     R4           ;; DON'T SUPPRESS ANYMORE 0'S
3265 022554 052703 000060  BIS     #'0,R3      ;; MAKE THIS DIGIT ASCII
3266 022560 052703 000040  5$:    BIS     #' ,R3   ;; MAKE ASCII IF NOT ALREADY
3267 022564 110337 022630  MOVB    R3,$S       ;; SAVE FOR TYPING
3268 022570 104401 022630  TYPE    $S           ;; GO TYPE THIS DIGIT
3269 022574 105337 022632  7$:    DECB    $OCNT    ;; COUNT BY 1
3270 022600 003347          BGT    2$           ;; BR IF MORE TO DO
3271 022602 002402          BLT    6$           ;; BR IF DONE
3272 022604 005204          INC     R4           ;; INSURE LAST DIGIT ISN'T A BLANK
3273 022606 000744          BR     2$           ;; GO DO THE LAST DIGIT
3274 022610 012605          6$:    MOV     (SP)+,R5  ;; RESTORE R5
3275 022612 012604          MOV     (SP)+,R4     ;; RESTORE R4
3276 022614 012603          MOV     (SP)+,R3     ;; RESTORE R3
3277 022616 016666 000002 000004  MOV     2(SP),4(SP)  ;; SET THE STACK FOR RETURNING
3278 022624 012616          MOV     (SP)+,(SP)
3279 022626 000002          RTI                    ;; RETURN
3280 022630          8$:    .BYTE   0           ;; STORAGE FOR ASCII DIGIT
3281 022631          .BYTE   0           ;; TERMINATOR FOR TYPE ROUTINE
3282 022632          .BYTE   0           ;; OCTAL DIGIT COUNTER
3283 022633          .BYTE   0           ;; ZERO FILL SWITCH
3284 022634 000000          $OMODE: .WORD   0     ;; NUMBER OF DIGITS TO TYPE
3285          .SBTTL  TYPE ROUTINE
3286
3287          ;; *****
3288          ;; *ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3289          ;; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3290          ;; *NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3291          ;; *NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3292          ;; *NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3293          ;; *
3294          ;; *CALL:
3295          ;; *1) USING A TRAP INSTRUCTION
3296          ;; *      TYPE      ,MESADR          ;; MESADR IS FIRST ADDRESS OF AN ASCII STRING
3297          ;; *OR
3298          ;; *      TYPE

```



```

3299          :*      MESADR
3300          :*
3301
3302 022636 105737 001157 $TYPE: TSTB $TFPLG      ;; IS THERE A TERMINAL?
3303 022642 100002          BPL 1$          ;; BR IF YES
3304 022644 000000          HALT          ;; HALT HERE IF NO TERMINAL
3305 022646 000430          BR 3$          ;; LEAVE
3306 022650 010046          1$: MOV RO, -(SP)  ;; SAVE RO
3307 022652 017600 000002  MOV @2(SP), RO  ;; GET ADDRESS OF ASCIZ STRING
3308 022656 122737 000001 001222  CMPB #APTENV, $ENV  ;; RUNNING IN APT MODE
3309 022664 001011          BNE 62$       ;; NO, GO CHECK FOR APT CONSOLE
3310 022666 132737 000100 001223  BITB #APTSPOOL, $ENVM  ;; SPOOL MESSAGE TO APT
3311 022674 001405          BEQ 62$       ;; NO, GO CHECK FOR CONSOLE
3312 022676 010037 022706  MOV RO, 61$  ;; SETUP MESSAGE ADDRESS FOR APT
3313 022702 004737 023126  JSR PC, $ATY3  ;; SPOOL MESSAGE TO APT
3314 022706 000000          .WORD 0      ;; MESSAGE ADDRESS
3315 022710 132737 000040 001223  62$: BITB #APTCSUP, $ENVM  ;; APT CONSOLE SUPPRESSED
3316 022716 001003          BNE 60$       ;; YES, SKIP TYPE OUT
3317 022720 112046          2$: MOVB (RO)+, -(SP)  ;; PUSH CHARACTER TO BE TYPED ONTO STACK
3318 022722 001005          BNE 4$          ;; BR IF IT ISN'T THE TERMINATOR
3319 022724 005726          TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
3320 022726 012600          60$: MOV (SP)+, RO  ;; RESTORE RO
3321 022730 062716 000002  3$: ADD #2, (SP)  ;; ADJUST RETURN PC
3322 022734 000002          RTI          ;; RETURN
3323 022736 122716 000011  4$: CMPB #HT, (SP)  ;; BRANCH IF <HT>
3324 022742 001430          BEQ 8$          ;;
3325 022744 122716 000200  CMPB #CRLF, (SP)  ;; BRANCH IF NOT <CRLF>
3326 022750 001006          BNE 5$          ;;
3327 022752 005726          TST (SP)+      ;; POP <CR><LF> EQUIV
3328 022754 104401          TYPE          ;; TYPE A CR AND LF
3329 022756 001177          $CRLF
3330 022760 105037 023114  CLRB $CHARCNT  ;; CLEAR CHARACTER COUNT - - - -
3331 022764 000755          2$:          ;; GET NEXT CHARACTER
3332 022766 004737 023050  5$: JSR PC, $TYPEC  ;; GO TYPE THIS CHARACTER
3333 022772 123726 001156  6$: CMPB $FILLC, (SP)+  ;; IS IT TIME FOR FILLER CHARS.?
3334 022776 001350          BNE 2$          ;; IF NO GO GET NEXT CHAR.
3335 023000 013746 001154  MOV $NULL, -(SP)  ;; GET # OF FILLER CHARS. NEEDED
3336          ;; AND THE NULL CHAR.
3337 023004 105366 000001  7$: DECB 1(SP)  ;; DOES A NULL NEED TO BE TYPED?
3338 023010 002770          BLT 6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
3339 023012 004737 023050  JSR PC, $TYPEC  ;; GO TYPE A NULL
3340 023016 105337 023114  DECB $CHARCNT  ;; DO NOT COUNT AS A COUNT
3341 023022 000770          BR 7$          ;; LOOP
3342
3343          ;HORIZONTAL TAB PROCESSOR
3344
3345 023024 112716 000040  8$: MOVB #' (SP)  ;; REPLACE TAB WITH SPACE
3346 023030 004737 023050  9$: JSR PC, $TYPEC  ;; TYPE A SPACE
3347 023034 132737 000007 023114  BITB #7, $CHARCNT  ;; BRANCH IF NOT AT
3348 023042 001372          BNE 9$          ;; TAB STOP
3349 023044 005726          TST (SP)+      ;; POP SPACE OFF STACK
3350 023046 000724          BR 2$          ;; GET NEXT CHARACTER
3351 023060 105777 156074  $TYPEC: TSTB @2$PS  ;; WAIT UNTIL PRINTER IS READY
3352 023051 100375          BPL $TYPEC

```

```

3353 023056 116677 000002 156066      MOVB    2(SP),2$TPB      ;; LOAD CHAR TO BE TYPED INTO DATA REG.
3354 023064 122766 000015 000002      CMPB    #CR,2(SP)      ;; IS CHARACTER A CARRIAGE RETURN?
3355 023072 001003          BNE     1$             ;; BRANCH IF NO
3356 023074 105037 023114      CLRB    $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
3357 023100 000406          BR      $TYPEX        ;; EXIT
3358 023102 122766 000012 000002 1$:      CMPB    #LF,2(SP)      ;; IS CHARACTER A LINE FEED?
3359 023110 001402          BEQ     $TYPEX        ;; BRANCH IF YES
3360 023112 105227          INCB   (PC)+         ;; COUNT THE CHARACTER
3361 023114 000000      $CHARCNT: .WORD    0   ;; CHARACTER COUNT STORAGE
3362 023116 000207      $TYPEX:  RTS        PC
3363
3364          .SBTTL  APT COMMUNICATIONS ROUTINE
3365
3366          ..*****
3367 023120 112737 000001 023364  $ATY1:  MOVB    #1,$FFLG      ;; TO REPORT FATAL ERROR
3368 023126 112737 000001 023362  $ATY3:  MOVB    #1,$MFLG      ;; TO TYPE A MESSAGE
3369 023134 000403          BR      $ATYC
3370 023136 112737 000001 023364  $ATY4:  MOVB    #1,$FFLG      ;; TO ONLY REPORT FATAL ERROR
3371 023144          $ATYC:
3372 023144 010046          MOV     R0,-(SP)      ;; PUSH R0 ON STACK
3373 023146 010146          MOV     R1,-(SP)      ;; PUSH R1 ON STACK
3374 023150 105737 023362      TSTB   $MFLG          ;; SHOULD TYPE A MESSAGE?
3375 023154 001450          BEQ     5$           ;; IF NOT: BR
3376 023156 122737 000001 001222      CMPB    #APTENV,$ENV   ;; OPERATING UNDER APT?
3377 023164 001031          BNE     3$           ;; IF NOT: BR
3378 023166 132737 000100 001223      BITB   #APTSPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
3379 023174 001425          BEQ     3$           ;; IF NOT: BR
3380 023176 017600 000004          MOV     @4(SP),R0     ;; GET MESSAGE ADDR.
3381 023202 062766 000002 000004          ADD     #2,4(SP)     ;; BUMP RETURN ADDR.
3382 023210 005737 001202          1$:  TST     $MSGTYPE     ;; SEE IF DONE W/ LAST XMISSION?
3383 023214 001375          BNE     1$           ;; IF NOT: WAIT
3384 023216 010037 001216          MOV     R0,$MSGAD    ;; PUT ADDR IN MAILBOX
3385 023222 105720          2$:  TSTB   (R0)+        ;; FIND END OF MESSAGE
3386 023224 001376          BNE     2$
3387 023226 163700 001216          SUB     $MSGAD,R0     ;; SUB START OF MESSAGE
3388 023232 006200          ASR    R0            ;; GET MESSAGE LNTH IN WORDS
3389 023234 010037 001220          MOV     R0,$MSGGLT   ;; PUT LENGTH IN MAILBOX
3390 023240 012737 000004 001202      MOV     #4,$MSGTYPE   ;; TELL APT TO TAKE MSG.
3391 023246 000413          BR      5$
3392 023250 017637 000004 023274  3$:  MOV     @4(SP),4$     ;; PUT MSG ADDR IN JSR LINKAGE
3393 023256 062766 000002 000004          ADD     #2,4(SP)     ;; BUMP RETURN ADDRESS
3394 023264 013746 177776          MOV     177776,-(SP) ;; PUSH 177776 ON STACK
3395 023270 004737 022636          JSR    PC,$TYPE      ;; CALL TYPE MACRO
3396 023274 000000          4$:  .WORD    0
3397 023276          5$:
3398 023276 105737 023364          10$: TSTB   $FFLG          ;; SHOULD R PORT FATAL ERROR?
3399 023302 001416          BEQ     12$          ;; IF NOT: BR
3400 023304 005737 001222          TST    $ENV          ;; RUNNING UNDER APT?
3401 023310 001413          BEQ     12$          ;; IF NOT: BR
3402 023312 005737 001202          11$: TST     $MSGTYPE     ;; FINISHED LAST MESSAGE?
3403 023316 001375          BNE     11$          ;; IF NOT: WAIT
3404 023320 017637 000004 001204      MOV     @4(SP),$FATAL ;; GET ERROR #
3405 023326 062766 000002 000004          ADD     #2,4(SP)     ;; BUMP RETURN ADDR.
3406 023334 005237 001202          INC    $MSGTYPE     ;; TELL APT TO TAKE ERROR

```

```

3407 023340 105037 023364      12$: CLRB   $FFLG      ;; CLEAR FATAL FLAG
3408 023344 105037 023363      CLRB   $LFLG      ;; CLEAR LOG FLAG
3409 023350 105037 023362      CLRB   $MFLG      ;; CLEAR MESSAGE FLAG
3410 023354 012601          MOV    (SP)+,R1   ;; POP STACK INTO R1
3411 023356 012600          MOV    (SP)+,RO   ;; POP STACK INTO RO
3412 023360 000207          RTS     PC        ;; RETURN
3413 023362      000      $MFLG: .BYTE    0      ;; MESSG. FLAG
3414 023363      000      $LFLG: .BYTE    0      ;; LOG FLAG
3415 023364      000      $FFLG: .BYTE    0      ;; FATAL FLAG
3416      023366          .EVEN
3417      000200      APTSIZE=200
3418      000001      APTENV=001
3419      000100      APTSPool=100
3420      000040      APTCSUP=040
3421      .SBTTL TRAP DECODER
3422
3423 *****
3424 *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3425 *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3426 *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3427 *GO TO THAT ROUTINE.
3428
3429 023366 010046          $TRAP: MOV    RO, -(SP)      ;; SAVE RO
3430 023370 016600 000002      MOV    2(SP),RO      ;; GET TRAP ADDRESS
3431 023374 005740          TST    -(RO)        ;; BACKUP BY 2
3432 023376 111000          MOVB  (RO),RO      ;; GET RIGHT BYTE OF TRAP
3433 023400 006300          ASL   RO            ;; POSITION FOR INDEXING
3434 023402 016000 023422      MOV    $TRPAD(RO),RO ;; INDEX TO TABLE
3435 023406 000200          RTS     RO          ;; GO TO ROUTINE
3436
3437
3438 ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
3439
3440 023410 011646          $TRAP2: MOV   (SP), -(SP)    ;; MOVE THE PC DOWN
3441 023412 016666 000004 000002      MOV   4(SP), 2(SP)      ;; MOVE THE PSW DOWN
3442 023420 000002          RTI                ;; RESTORE THE PSW
3443
3444 .SBTTL TRAP TABLE
3445
3446 *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3447 *BY THE "TRAP" INSTRUCTION.
3448
3449 ;
3450 ROUTINE
3451 -----
3452 $TRPAD: .WORD  $TRAP2
3453 $TYPE   ;; CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
3454 $TYPOC  ;; CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3455 $TYPOS  ;; CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3456 $TYPON  ;; CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3457 $TYPDS  ;; CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
3458 023436 021226          $GTSWR ;; CALL=GTSWR      TRAP+6(104406)  GET SOFT-SWR SETTING
3459
3460 023440 021156          $CKSWR ;; CALL=CKSWR      TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR

```

```

3461 023442 021440 $RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
3462 023444 021560 $RDLIN ;;CALL=RDLIN TPAP+11(104411) TTY TYPEIN STRING ROUTINE
3463 023446 021732 $RDOCT ;;CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481 023450 $LPAI:
3482 023450 013746 000004 MOV 4,-(SP)
3483
3484 023454 000413 BR 31$
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497 023456 012737 023502 000004 MOV #30$,4
3498 023464 005237 170000 INC 170000
3499 023470 104401 023476 TYPE 65$
3500 023474 000401 BR 64$
3501
3502 023500 ;;65$: .ASCIZ <?>##
3503 023500 64$: BR 31$
3504 023502 000401 30$: CMP (SP)+,(SP)+
3505 023504 012637 000004 31$: MOV (SP)+,4
3506 023510 005037 024326 CLR $AERR
3507 023514 004537 024330 JSR RS,$LOAD
3508 023520 000000G .WORD DRLPX2
3509
3510 023522 052777 040000 155736 BIS #BIT14,$KMADD
3511
3512 023530 1$:
3513
3514 023530 010146 MOV R1,-(SP)

```

```

;*
;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
;*NEXT WE WILL INIT BOTH UPROCESSORS
;*THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
;*THE ORDER OF LOAD IS DETERMINED BY THE USER.
;*
;*      CALL=   JSR      RS,$LPAI
;*              .WORD   0                ;ADDR. OF DEVICE ADDRESS.
;* ROUTINES REQUIRED: .LOADLP
;* PROGRAMS REQUIRED: DRLPX2
;*
;*              ;RETURNS WITH $AERR=1 IF SLAVE
;*              ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
;*
;FIELD DOES NOT HAVE A BUS SWITCH TO
;WORRY ABOUT,SO WE WILL UNCONDITIONALLY
;BRANCH AROUND THE NEXT CODE THAT
;WORKS BASED ON A BUS SWITCH.
;CODE LEFT IN HERE FOR IN HOUSE
;PERSONAL WHO MAY PATCH THIS BRANCH
;INSTRUCTION TO A <NOP> OCTAL <240>
;IN ORDER TO RUN PROGRAM WITH A SWITCH.
;NOTE THIS "SWITCH" IS A PIECE OF INHOUSE
;TEST EQUIPMENT ONLY IT CONNECTS
;THE UNIBUS TO THE I/O BUS FOR
;CERTAIN TESTING.
;;TYPE ASCIZ STRING
;;GET OVER THE ASCIZ
;ALL THIS JUNK MUST BE REMOVED!!
;LOAD MICRO-CODE.
;FILE "DRLPX2.OBJ"
;ISSUE KMC+DMC INIT.
;"HANGS" HERE THEN KMC-11 ERROR.

```

```

3515 023532 005001          CLR      R1
3516 023534 005201      25:  INC      R1          ;STALL FOR DMC-UP
3517 023536 001376          BNE     25$
3518 023540 012777 104000 155720  MOV     #BIT15!BIT11,2AKMADO ;SET RUN, AND ENABLE ARBITRATION.
3519 023546 105201      25$:  INCB   R1
3520 023550 001376          BNE     25$
3521
3522 023552 032777 000040 155700  BIT     #BITS,2AKMADO ;SLAVE READY? (READING IPBM SR)
3523 023560 001401          BEQ     35$
3524
3525 023562 104000          ERROR   ;FATAL LPA-11 ERROR SLAVE NOT READY.
3526
3527 023564 012777 000004 155700  35:  MOV     #4,2AKMAD2 ;READ FAST PATH
3528 023572          45:
3529 023572 004537 025240  JSR     R5, $TOUT ;-TOUT-CHECK FOR TIMEOUT
3530
3531 023576 104000          ERROR   ;/TIME-OUT ERROR
3532          ;/WE FAILED TO COMPLETE
3533          ;/CURRENT OPERATION.
3534          ;/CONTINUES IN THIS LOOP
3535          ;/WOULD MAKE US "HANG" HERE
3536
3537 023600 000774          BR      45$
3538
3539          ;/RETURNS HERE-FROM-TIMED OUT.
3540 023602 122777 000377 155662  CMPB   #377,2AKMAD2 ;WAIT TILL KMC DONE COMMAND.
3541 023610 001370          BNE     45$
3542 023612 122777 000377 155656  CMPB   #377,2AKMAD4 ;IF FAST PATH=377 THEN ERROR.
3543 023620 001001          BNE     35$
3544 023622 104000          ERROR   ;IPBM ERROR (SLAVE SIDE)
3545          ;YOU MUST RUN IPBM DIAGNOSTIC.
3546
3547 023624 122777 000004 155644  35$:  CMPB   #4,2AKMAD4 ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
3548 023632 001543          BEQ     55$ ;YES-CONTINUE.
3549 023634 005227 177777  INC     #-1
3550 023640 001140          BNE     55$
3551 023642 005227 177777  INC     #-1
3552 023646 001135          BNE     55$
3553 023650 104401 023656  TYPE   67$ ;:TYPE ASCIZ STRING
3554 023654 000440          BR      66$ ;:GET OVER THE ASCIZ
3555          ;: "W A R N I N G THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
3556 023756          ;:67$: .ASCIZ <200>"W A R N I N G THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
3557 023756 104401 023764  TYPE   69$ ;:TYPE ASCIZ STRING
3558 023762 000430          BR      68$ ;:GET OVER THE ASCIZ
3559          ;:69$: .ASCIZ <200>"MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED."
3560 024044          ;:68$:
3561 024044 104401 024052  TYPE   71$ ;:TYPE ASCIZ STRING
3562 024050 000434          BR      70$ ;:GET OVER THE ASCIZ
3563          ;:71$: .ASCIZ <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
3564 024142          ;:70$:
3565
3566 024142 112737 177777 024274  5$:  MOVB   #0-1,11$ ;DAC CODE FOR SLAVE.
3567 024150 012501          MOV     (5)+,R1 ;GET NEXT DEVICE ADDR.
3568 024152 021127 000000      6$:  CMP     (R1),#0 ;TERM REACHED?

```

3569	024156	001444			BEQ	10\$		
3570	024160	105237	024274		INCB	11\$		
3571	024164	113777	024274	155304	MOVB	11\$, @KMAD4		; FIFO DATA
3572	024172	004737	024276		JSR	PC, 20\$		; ISSUE SEND
3573	024176	112177	155274		MOVB	(R1)+, @KMAD4		; SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
3574	024202	004737	024276		JSR	PC, 20\$		; ISSUE SEND
3575	024206	112177	155264		MOVB	(R1)+, @KMAD4		; SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
3576	024212	004737	024276		JSR	PC, 20\$		
3577								
3578	024216	032777	000002	155242	7\$: BIT	#BIT1, @KMAD0		; WAIT FOR FIFO DATA
3579	024224	001374			BNE	7\$		; =1 NO DATA. =0 DATA.
3580	024226	112777	000002	155236	MOVB	#2, @KMAD2		; READ FIFO.
3581								
3582	024234				8\$: JSR	RS, \$TOUT		; -TOUT-CHECK FOR TIMEOUT
3583	024234	004537	025240					
3584								
3585	024240	104000			ERROR			; /TIME-OUT ERROR
3586								; /WE FAILED TO COMPLETE
3587								; /CURRENT OPERATION.
3588								; /CONTINUES IN THIS LOOP
3589								; /WOULD MAKE US "HANG" HERE
3590								
3591	024242	000774			BR	8\$		
3592								
3593								
3594	024244	122777	000377	155220	CMPB	#377, @KMAD2		; /RETURNS HERE-FROM-TIMED OUT.
3595	024252	001370			BNE	8\$		; WAIT FOR READ.
3596	024254	105777	155216		TSTB	@KMAD4		; WAS A ZERO RETURNED?
3597	024260	001734			BEQ	6\$		; YES GET NEXT ADDR.
3598								; SLAVE WILL RETURN CODE 0 IF
3599	024262	005237	024326		INC	\$AERR		; DEV PRESENT. ELSE
3600								; EXIT \$AERR=1 IF SLAVE GIVES ERROR.
3601	024266	005041			CLR	-(1)		; GET RID OF REFERENCE TO BAD ADDR.
3602	024270	012601			10\$: MOV	(SP)+, R1		
3603	024272	000205			RTS	RS		; RETURN ALL ADDR. CHECKED.
3604								
3605	024274	000000			11\$: .WORD	0		; HOLDS DAC CODE PLUS OFFSET
3606								; TO SLAVES ADDR. TABLE.
3607								
3608	024276	112777	000003	155166	20\$: MOVB	#3, @KMAD2		; ISSUE FIFO WRITE
3609	024304				21\$: JSR	RS, \$TOUT		; -TOUT-CHECK FOR TIMEOUT
3610	024304	004537	025240					
3611								
3612	024310	104000			ERROR			; /TIME-OUT ERROR
3613								; /WE FAILED TO COMPLETE
3614								; /CURRENT OPERATION.
3615								; /CONTINUES IN THIS LOOP
3616								; /WOULD MAKE US "HANG" HERE
3617								
3618	024312	000774			BR	21\$		
3619								
3620								
3621	024314	122777	000377	155150	CMPB	#377, @KMAD2		; /RETURNS HERE-FROM-TIMED OUT.
3622	024322	001370			BNE	21\$		; KMC CODE WILL RETURN A "377"
								; WHEN DONE COMMAND.



```

3677 ;*THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
3678 ;*
3679 ;* CALL = JSR R5,$TLKW
3680 ;* .WORD 0 ;OFFSET OF DEVICE ADDR.
3681 ;* .WORD 0 ;DATA TO BE WRITTEN
3682 ;*
3683 024470 010046 $TLKW: MOV R0,-(SP) ;SAVE R0
3684 024472 012500 MOV (5)+,R0 ;GET DEVICE OFFSET
3685 024474 052700 000340 BIS #340,R0 ;ADD WRITE CODE.
3686 024500 004737 024752 JSR PC,$LPW ;WAIT FOR FAST PATH READY
3687 024504 010037 024576 MOV R0,W1
3688 024510 010077 154762 MOV R0,@KMA4
3689 024514 112777 000005 154750 MOVB #5,@KMA2 ;ISSUE FAST PATH WRITE
3690 024522 004737 024752 JSR PC,$LPW ;WAIT FOR RDY
3691 024526 011537 024600 MOV (5)W2
3692 024532 112577 154740 MOVB (5)+,@KMA4 ;WRITE LOW BYTE DATA.
3693
3694 024536 112777 000005 154726 MOVB #5,@KMA2 ;FP WRITE
3695 024544 004737 024752 JSR PC,$LPW
3696 024550 111537 024602 MOVB (5)W3
3697 024554 112577 154716 MOVB (5)+,@KMA4 ;WRITE HIGH BYTE
3698 024560 112777 000005 154704 MOVB #5,@KMA2
3699 024566 004737 024752 JSR PC,$LPW
3700 024572 012600 MOV (SP)+,R0
3701 024574 000205 RTS R5 ;EXIT DONE.
3702 024576 000000 W1: 0
3703 024600 000000 W2: 0
3704 024602 000000 W3: 0
3705
3706 ;*
3707 ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
3708 ;*
3709 ;* CALL = JSR R5,$TLKR
3710 ;* .WORD 0 ;OFFSET OF DEVICE
3711 ;* .WORD 0 ;RETURNS HERE
3712 ;* *DATA IN WORD $DATR
3713 ;*
3714
3715 024604 010046 $TLKR: MOV R0,-(SP) ;SAVE R0
3716 024606 012500 MOV (5)+,R0 ;GET OFFSET
3717 024610 052700 000300 BIS #300,R0 ;ADD READ CODE
3718 024614 004737 024752 JSR PC,$LPW ;WAIT TILL READY
3719 024620 110077 154652 MOVB R0,@KMA4
3720 024624 112777 000005 154640 MOVB #5,@KMA2 ;ISSUE WRITE FP
3721 024632 004737 024752 JSR PC,$LPW
3722 024636 010037 024746 MOV R0,R01
3723 024642
3724 024642 004537 025240 1$: JSR R5,$TOUT ;--TOUT-CHECK FOR TIMEOUT
3725
3726 024646 104000 ERROR ;/TIME-OUT ERROR
3727 ;/WE FAILED TO COMPLETE
3728 ;/CURRENT OPERATION.
3729 ;/CONTINUES IN THIS LOOP
3730 ;/WOULD MAKE US "HANG" HERE

```



```

3731
3732 024650 000774 BR 1$
3733
3734 ;/RETURNS HERE-FROM-TIMED OUT.
3735 024652 032777 000040 154606 BIT #BITS,@KMAD0 ;FAST PATH GOT DATA?
3736 024660 001370 BNE 1$
3737 024662 112777 000004 154602 MOVB #4,@KMAD2 ;ISSUE FAST PATH READ
3738 024670 004737 024752 JSR PC,$LPW
3739 024674 117737 154576 024750 MOVB @KMAD4,$DATR ;GET LOW BYTE
3740 024702 2$:
3741 024702 004537 025240 JSR R5,$TOUT ;-TOUT-CHECK FOR TIMEOUT
3742
3743 024706 104000 ERROR ;/TIME-OUT ERROR
3744 ;/WE FAILED TO COMPLETE
3745 ;/CURRENT OPERATION.
3746 ;/CONTINUES IN THIS LOOP
3747 ;/WOULD MAKE US "HANG" HERE
3748
3749 024710 000774 BR 2$
3750
3751 ;/RETURNS HERE-FROM-TIMED OUT.
3752 024712 032777 000040 154546 BIT #BITS,@KMAD0 ;FAST PATH READY?
3753 024720 001370 BNE 2$
3754 024722 112777 000004 154542 MOVB #4,@KMAD2 ;ISSUE FAST PATH READ
3755 024730 004737 024752 JSR PC,$LPW
3756 024734 117737 154536 024751 MOVB @KMAD4,$DATR+1 ;SAVE HIGH BYTE
3757 024742 012600 MOV (SP)+,R0
3758 024744 000205 RTS R5
3759 024746 000000 RD1: 0
3760 024750 000000 $DATR: .WORD 0
3761
3762 ; THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
3763 ; AS FAST PATH TO BE READ.
3764
3765 ; CALL = JSR PC,$LPW
3766
3767 ; IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
3768 ; THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
3769
3770
3771 024752 010146 $LPW: MOV R1,-(SP) ;SAVE R1
3772 024754 005001 CLR R1
3773 024756 122777 000377 154506 1$: CMPB #377,@KMAD2 ;FINISHED INSTRUCTION?
3774 024764 001403 BEQ 2$
3775 024766 005201 INC R1 ;TIME OUT?
3776 024770 001372 BNE 1$
3777 024772 000411 BR 10$
3778
3779 024774 032777 000020 154464 2$: BIT #BIT4,@KMAD0 ;FAST PATH READ?
3780 025002 001403 BEQ 3$
3781 025004 005201 INC R1 ;NO - TIME OUT?
3782 025006 001372 BNE 2$
3783 025010 000402 BR 10$ ;YES - REPORT AN ERROR
3784

```

```

3785 025012 012601      3$:  MOV    (SP)+,R1      ;RESTORE R1
3786 025014 000207      RTS    PC              ;EXIT
3787
3788 025016              10$:  TYPE    65$          ;;TYPE ASCIZ STRING
3789 025016 104401 025024  BR     64$          ;;GET OVER THE ASCIZ
3790 025022 000407      ;;65$: .ASCIZ <200>#LPA-11 FAULT#
3791
3792 025042      64$:
3793
3794 025042 000000      11$:  HALT
3795 025044 000776      BR     11$          ;LPA-11 FAULT RUN LPA-11
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810 025046 010046      $OUTLP: MOV   R0,-(SP)    ;SAVE R0
3811 025050 010146      MOV   R1,-(SP)    ;SAVE R1
3812
3813 025052 012700 001514  MOV   #.DVLS,R0    ;PROGRAM DEFINED LIST.
3814 025056 005001      CLR   R1
3815 025060 005710      1$:  TST   (0)          ;TERMINATOR REACHED?
3816 025062 001421      BEQ   10$          ;YES NEXT STEP.
3817 025064 027520 000000  CMP   @5),(0)+     ;MATCH WITH ADDR IN LIST?
3818 025070 001402      BEQ   2$
3819 025072 005201      INC   R1
3820 025074 000771      BR    1$
3821
3822 025076 010137 025114  2$:  MOV   R1,3$        ;SAVE OFFSET, DEVICE KNOWN.
3823 025102 005725      TST   (5)+
3824 025104 013537 025116  MOV   @5)+,4$      ;GET DATA TO BE WRITTEN
3825 025110 004537 024470  JSR   R5,$TLKW     ;DO WRITE
3826 025114 000000      3$:  .WORD 0           ;DEVICE OFFSET
3827 025116 000000      4$:  .WORD 0           ;DATA TO BE WRITTEN.
3828 025120 012601      MOV   (SP)+,R1
3829 025122 012600      MOV   (SP)+,R0
3830 025124 000205      RTS   R5
3831 025126 017520 000000  10$: MOV   @5),(0)+     ;SAVE ADDR.
3832 025132 005010      CLR   (0)
3833 025134 004537 023450  JSR   R5,$LPAI
3834 025140 001514      .WORD .DVLS
3835 025142 000755      BR    2$
3836
3837
3838

```

```

;*
;* THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
;* A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.
;*
;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED
;* BEFORE. IF NOT WE HAVE TO INITIALIZE THE LPA WITH
;* THAT ADDRESS.
;* WHEN THE ADDR. IS KNOWN BY THE LPA, DO THE OUTPUT BY
;* $TLKW
;*

```

```

;*
;* THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE

```

```

3839 ;*TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
3840 ;*
3841 ;*FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN
3842 ;*USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
3843 ;*WITH THE NEW ADDR.
3844 ;*WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH
3845 ;*$TLKR
3846 ;* CALL THROUGH MOVEI DATA,ADDR.
3847 ;* WHICH EQUALS:
3848 ;* JSR RS,$INLP
3849 ;* .WORD XX ADDR OF DEVICE
3850 ;* .WORD YY ADDR TO STORE READ DATA.
3851
3852 025144 010046 $INLP: MOV RO,-(SP) ;SAVE RO
3853 025146 010146 MOV R1,-(SP) ;SAVE R1
3854
3855 025150 012700 001514 MOV #.DVLS,R0 ;PROG DEFINED ADDR. LIST.
3856 025154 005001 CLR R1
3857 025156 005710 1$: TST (0) ;EOL REACHED?
3858 025160 001420 BEQ 10$ ;YES - DEFINE NEW ADDR.
3859
3860 025162 027520 000000 CMP @ (5),(0)+ ;ADDR. MATCH?
3861 025166 001402 BEQ 2$
3862 025170 005201 INC R1
3863 025172 000771 BR 1$
3864
3865 025174 010137 025206 2$: MOV R1,3$ ;SAVE LIST OFFSET
3866 025200 005725 TST (5)+
3867 025202 004537 024604 JSR RS,$TLKR ;GO READ DEVICE
3868 025206 000000 $OFS=.
3869 025206 000000 3$: .WORD 0 ;OFFSET OF DEVICE
3870
3871 025210 013735 024750 MOV $DATR,@ (5)+ ;STORE DATA.
3872 025214 012601 MOV (SP)+,R1 ;RESTORE R1
3873 025216 012600 MOV (SP)+,R0 ;RESTORE R2
3874 025220 000205 RTS RS ;EXIT
3875
3876 025222 017520 000000 10$: MOV @ (5),(0)+
3877 025226 005010 CLR (0)
3878 025230 004537 023450 JSR RS,$LPAI
3879 025234 001514 .WORD .DVLS
3880 025236 000756 BR 2$
3881 ;*
3882 ;*$STOUT ROUTINE USED TO WATCH IF
3883 ;* WE'RE IN A LOOP TOO-LONG
3884 ;* CALL= JSR RS,$STOUT
3885 ;* ERROR X ;RETURNS HERE ON TIMEOUT
3886 ;* BR
3887 ;* ;RETURNS HERE NO ERROR
3888 ;*
3889
3890 025240 020537 025274 $STOUT: CMP RS,$SAD ;SAME ADDR?
3891 025244 001405 BEQ 1$
3892 025246 010537 025274 MOV RS,$SAD ;NO-SAVE THIS ADDR.
    
```

3893	025252	005037	025276		CLR	\$CNT		; CLR CNT AT ADDR.
3894	025256	000403			BR	2\$		
3895	025260	005237	025276	1\$:	INC	\$CNT		; OVERFLOW?
3896	025264	100402			BMI	3\$		; YES-ERROR RETURN
3897	025266	062705	000004	2\$:	ADD	#4,R5		; NO-NON ERROR RETURN
3898	025272	000205		3\$:	RTS	RS		; RETURN.
3899								
3900	025274	000000		\$SAD:	.WORD	0		; CONTAINS LOOP ADDR.
3901	025276	000000		\$CNT:	.WORD	0		; # OF TIMES AT ADDR.
3902								
3903								
3904								
3905								
3906								
3907								
3908								
3909								
3910								
3911								
3912	025300	000005		\$RESET:	RESET			; RESET THE WORLD.
3913								
3914								
3915	025312	005737	024326	*:	MOV	2\$ 1\$		; /READ DEVICE REG 2\$, PUT DATA IN 1\$.
3916	025316	001004			TST	\$AERR		; IF NO ERROR, LOOP
3917	025320	062737	000002 025334		BNE	10\$		; THERE WAS AN ERROR.
3918					ADD	#2,2\$		; UPDATE DEVICE ADDR.
3919								; YOU SEE, WE HAVE TO PROTECT OUR SELF!
3920								; IF 2\$ CONTAINED A VALID ADDR, WE
3921								; MUST KEEP TRYING UNTIL WE GENERATE
3922	025326	000764			BR	\$RESET		; AN INVALID ADDR.
3923	025330			10\$:				
3924	025330	000207			RTS	PC		
3925	025332	000000		1\$:	.WORD	0		; JUNK LOC.
3926	025334	160000		2\$:	.WORD	160000		; DUMB ADDR. FORCES INIT OF DMC/KMC.
3927								
3928								
3929								
3930								
3931								
3932								
3933								
3934								
3935								
3936								
3937								
3938								
3939								
3940								
3941	025336			SDELAY:				
3942	025336	005737	025420		TST	RTCCSR		; CLOCK PRESENT?
3943	025342	100016			BPL	10\$		
3944	025344	012737	000002 025410		MOV	#2 TIME		
3945	025352	052777	000115 000040		BIS	#115, @RTCCSR		; START CLOCK
3946	025360	005037	177776		CLR	PS		

```

; * THIS ROUTINE REPLACES WHAT THE USER WOULD ORDINARILY
; * USE FOR A RESET. FIRST, WE DO A RESET INSTRUCTION.
; * THEN WE CLR "DVLST" WHICH FORCES US TO RESET BOTH THE
; * KMC AND DMC AS SOON AS A DEVICE IS REFERENCED.
; *
; * CALL=JSR PC, $RESET ; REPLACES "RESET INSTRUCTION"
; * ; RETURNS HERE.
; *
; * $RESET: RESET ; RESET THE WORLD.
; *
; * MOV 2$ 1$ ; /READ DEVICE REG 2$, PUT DATA IN 1$.
; * TST $AERR ; IF NO ERROR, LOOP
; * BNE 10$ ; THERE WAS AN ERROR.
; * ADD #2,2$ ; UPDATE DEVICE ADDR.
; * YOU SEE, WE HAVE TO PROTECT OUR SELF!
; * IF 2$ CONTAINED A VALID ADDR, WE
; * MUST KEEP TRYING UNTIL WE GENERATE
; * AN INVALID ADDR.
; *
; * BR $RESET
; *
; * 10$:
; * RTS PC
; * 1$: .WORD 0 ; JUNK LOC.
; * 2$: .WORD 160000 ; DUMB ADDR. FORCES INIT OF DMC/KMC.
; *
; * SDELAY- ROUTINE TO GIVE A MINOR DELAY.
; * IS NOT TIME DEPENDENT CODE SENCE
; * NOT USED TO GET SPECIFIC TIME BUT
; * JUST A LITTLE DELAY.
; *
; * THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
; * THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
; *
; * CALL= JSR PC, SDELAY
; *
; * SDELAY:
; * TST RTCCSR ; CLOCK PRESENT?
; * BPL 10$
; * MOV #2 TIME
; * BIS #115, @RTCCSR ; START CLOCK
; * CLR PS
    
```

3947	025364	005737	025410	1\$:	TST	TIME	
3948	025370	001375			BNE	1\$	
3949	025372	005077	000022		CLR	RTCCSR	;STOP CLOCK
3950							
3951	025376	000207			RTS	PC	
3952	025400	105237	025410	10\$:	INCB	TIME	
3953	025404	001375			BNE	10\$	
3954	025406	000207			RTS	PC	
3955							
3956	025410	000000		TIME:	.WORD	0	
3957							
3958	025412	005337	025410	CLKINT:	DEC	TIME	
3959	025416	000002			RTI		
3960	025420	000000		RTCCSR:	.WORD	0	;CLOCK CSR IF USED.
3961							
3962							
3963							
3964							
3965							
3966							
3967							
3968							
3969							
3970							
3971							
3972	025422			\$UTK:			
3973	025422	005037	001514		CLR	.DVLS	
3974	025426			21\$:			
3975	025426	104401	025434		TYPE	65\$	::TYPE ASCIZ STRING
3976	025432	000405			BR	64\$	::GET OVER THE ASCIZ
3977							
3978	025446						
3979	025446	105777	153472				
3980	025452	100375			TSTB	2\$TKS	
3981	025454	117737	153466	025576	BPL	1\$	
3982	025462	104401	025576		MOVW	2\$TKB, 20\$	;GET INPUT
3983	025466	142737	000240	025576	TYPE	20\$	;ECHO, NEXT MESSAGE.
3984	025474	104412			BICB	#240, 20\$	;STRIP PARITY, LC
3985	025476	012637	025574		RDOCT		;GET ADDR.
3986	025502	123727	025576	000104	MOV	(SP)+, 14\$	
3987	025510	001411			CMPB	20\$, #1D	;DEPOSIT?
3988					BEG	10\$	
3989	025512	004537	025144				
3990	025516	025574		2\$:	JSR	R5, \$INLP	;GET DATA
3991	025520	025532			.WORD	14\$	
3992					.WORD	5\$	
3993	025522	013746	025532				
3994	025526	104402			MOV	5\$, -(SP)	::SAVE 5\$ FOR TYPEOUT
3995	025530	000736			TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
3996	025532	000000			BR	21\$	::LOOP.
3997				5\$:	.WORD	0	
3998	025534						
3999	025534	104401	025542	10\$:			
4000	025540	000404			TYPE	67\$	::TYPE ASCIZ STRING
					BR	66\$	::GET OVER THE ASCIZ

```

; *THIS MACRO ALLOWS THE OPERATOR TO TALK TO
; *ANY DEVICE ON THE I/O BUS
; *USER MUST START AT THIS ADDR.
; *HE MUST SAY EITHER "E" FOR EXAMINE, OR "D" FOR DEPOSIT.
; *"E" IS DEFAULT.
; *NEXT, HE MUST SUPPLY AN ADDR.
; *NOTE IF ADDR. IS NOT FOUND ON I/O BUS, A HALT
; *WILL OCCUR.

```

```

4001
4002 025552
4003 025552 104412
4004 025554 012637 025572
4005
4006 025560 004537 025046
4007 025564 025574
4008 025566 025572
4009 025570 000716
4010
4011 025572 000000
4012 025574 000000
4013 025576 100001 042504 044526
4014 025604 042503 040440 042104
4015 025612 036522 000040
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040 025616 012537 025626
4041 025622 004537 025144
4042 025626 000000
4043 025630 025724
4044 025632 113777 025206 153642
4045 025640 113777 025206 153636
4046 025646 013737 025626 025666
4047 025654 062737 000002 025666
4048 025662 004537 025144
4049 025666 000000
4050 025670 025724
4051 025672 113777 025206 153574
4052 025700 152777 000340 153574
4053 025706 152777 000300 153570
4054 025714 152777 000300 153552

```

```

66$: .ASCIZ <200>#DATA= #
RDOCT
MOV (SP)+,13$
11$: JSR R5,$OUTLP ;OUTPUT ROUTINE.
12$: .WORD 14$ ;DEVICE ADDR.
.WORD 13$ ;DATA
BR 21$
13$: .WORD 0
14$: .WORD 0
20$: .ASCIZ <1><200>#DEVICE ADDR= #

```

.EVEN

```

: THIS ROUTINE LOOKS THROUGH CURENT .OVLS FOR A/D ADDR.
: IF UNFOUND, GENERATES IT. THIS ROUTINE'S WHOLE PURPOSE IS
: TO SET UP THE USER PROGRAM TO LINK TO FILE "DRLPX2" FOR
: SAMPLE TAKEING PURPOSES.
: TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
: A/D CSR IN BSEL 4 AND 5.
: (2) HE MUST CALL THIS ROUTINE:
: JSR R5,$PUTS ;CALL SET UP ROUTINE.
: .WORD ADCSR ;ADDR. OF A/D CSR.
: ;RETURNS HERE ;KMC BSEL 3,6,7 PERMINENTLY SET UP
: ;(UNTILL ONE DOES A RESET)
: (3)THE USER MUST PUT CODE 006 INTO KMC REG 2 TO
: START CONVERSION CAUTION*DO WITH MOVVB INSTR.!
: (4)MONITOR KMC REG 2 FOR CODE 377 (DRLPX2 IS DONE)
: (5)READ KMC REG 4,5 FOR A/D RESULT.
: (6) TO TAKE MORE SAMPLES, SIMPLY PUT A/D CSR INTO
: BSEL 4,5 AND CODE 6 INTO BSEL 2.

```

```

$PUTS: MOV (5)+,1$ ;GET ADDR OF ADDR. OF A/D
JSR R5,$INLP
1$: .WORD 0
.WORD 10$
MOVVB $OFS,@KMA06
MOVVB $OFS,@KMA07
MOV 1$,2$
ADD #2,2$
JSR R5,$INLP
2$: .WORD 0
.WORD 10$
MOVVB $OFS,@KMA03
BISB #340,@KMA06
BISB #300,@KMA07
BISB #300,@KMA03

```

4055	025722	000205			
4056	025724	000000	10\$:	RTS	R5
4057				.WORD	0
4058					
4059	025726	000000	ADBUFF:	0	
4060	025730	001774		.BLKW	1020.
4061	031720	000000	ABUFF4:	0	
4062	031722	000200		.BLKW	200
4063	032322	000000	LAST:	0	
4064		000001		.END	









CSR	001576	460#												
DACSAV	016246	1099#	1113	1114	1148	1162	1163	1202	1209	1234	1241	1468	1469	1487
		1488#	1505	1506	1531	1532	1557	1577	1651	1715	1981	2433#		
DDI#P =	177570	57#	241	518										
DECPNT	012524	2015*	2018	2020	2027*	2032*								
DECPRT	012416	2012#	2621	2627	2797									
DH1	013575	347	2185#											
DH10	014130	389	2224#											
DH11	014207	395	401	407	413	2232#								
DH2	013654	353	2193#											
DH3	013723	359	365	2200#										
DH5	013760	371	2205#											
DH6	014020	377	2211#											
DH7	014061	383	2217#											
DIEMSG	014523	1184	2272#											
DIFCHN	020146	2467*	2680	2732#										
DIFERR	017172	1073	2530*	2559*	2585#									
DIFLIN	016344	1059	2465#											
DIFMSG	014327	1176	2249#											
DIGCNT	012522	2012*	2023*	2031#										
DIGIT	012520	2016*	2017*	2021*	2022	2030#								
DIGTO	012536	2038#	2628											
DIGT1	012537	2014	2039#	2622	2629	2801								
DIGT2	012540	2040#	2623	2630	2802									
DIGT3	012541	2041#	2624	2631	2804									
DISPLA	001142	241#	518*	526*	2873*	2895*								
DISPRE	000174	173#	526											
DIYMSG	014513	1180	2270#											
DRLPX2 =	*****	14#	3508											
DSWR =	177570	56#	240	517										
DT1	015716	348	2392#											
DT10	015774	390	2401#											
DT11	016012	396	402	408	414	2404#								
DT2	015734	354	384	2395#										
DT3	015750	360	366	378	2397#									
DT5	015762	372	2399#											
EDGE	016314	1914*	1951	2401	2452#									
EMTVEC =	000030	145#	501*	502*										
EM1	013062	346	2121#											
EM10	013404	388	2159#											
EM11	013445	394	2165#											
EM12	013506	400	2171#											
EM13	013523	406	2174#											
EM14	013545	412	2178#											
EM14A	013571	1684*	1747*	2183#										
EM14B	013570	1691*	1754*	2182#										
EM2	013107	352	2125#											
EM3	013153	358	2132#											
EM4	013207	364	2137#											
EM5	013244	370	2142#											
EM6	013305	376	2148#											
EM7	013340	382	2153#											
ERRVEC =	000004	138#	515	516*	527*	2834	2835*	2837*	2840*					
EXCESS	017174	1069	2535*	2552*	2586#	2595	2658							

G

FCHANL	013004	1921*	1922*	1923*	1945	2098*									
FINS	016320	1925*	1981	1986*	2454*										
GNS	= ***** U	172	553	601	605	903	3452	3453	3454	3455	3456	3458	3460	3461	
		3462	3463	3501	3555	3559	3563	3791	3977	4001					
GRT2MG	015415	2356*	2598												
GTSWR	= 104406	548	3458*												
HILIM1	016326	2457*	2540	2642											
HILIM2	016330	2458*	2542												
HT	= 000011	48*	3323	3364											
IOTVEC	= 000020	143*	499*	500*											
KMAD0	001466	426*	481	3510*	3518*	3522	3578	3638*	3640*	3642*	3643*	3650*	3655*	3735	
		3752	3779												
KMAD1	001470	429*	482												
KMAD2	001472	431*	3527*	3540	3580*	3594	3608*	3621	3689*	3694*	3698*	3720*	3737*	3754*	
		3773													
KMAD3	001474	433*	4051*	4054*											
KMAD4	001476	435*	3542	3547	3571*	3573*	3575*	3596	3639*	3644*	3647*	3656*	3688*	3692*	
		3697*	3719*	3739	3756										
KMAD5	001500	437*													
KMAD6	001502	439*	3641*	3651	4044*	4052*									
KMAD7	001504	441*	486	4045*	4053*										
LAST	032322	2481	4063*												
LF	= 000012	49*	3358	3364											
LINMG1	015264	1765	2339*												
LINMSG	015101	1761	2318*												
LOLIM1	016322	2455*	2546	2640											
LOLIM2	016324	2456*	2548												
LPAOH	001500	436*													
LPADL	001476	434*													
LPCI	001466	425*													
LPCO	001472	430*													
LPMR	001470	428*													
LPMS1	001502	438*													
LPMS2	001504	440*													
LPSO	001474	432*													
MSGPNT	020342	2783*	2785*	2798											
MULTSP	015665	2046	2386*												
NARMSG	015444	2360*	2602												
NARROW	016334	2460*	2531*	2558*	2599	2609									
NBEXT	001566	455*	578*	1776	1780*	1784*									
NMBEXT	001562	453*	573*	578	596	1784									
NOIMG1	015010	1443	2306*												
NOIMG2	015033	1447	2310*												
NOIMG3	015056	1451	2314*												
NOIMSG	014667	1439	2291*												
NUMBF2	016026	1638	1702	2406*											
OFSLBX	016300	1505*	2446*												
OFSLUX	016302	1531*	2447*												
OFSTBX	016274	1468*	1513	1558	1578	2444*									
OFSTUX	016276	1487*	1539	2445*											
OVERNG	020020	2657	2711*												
PC	=%000007	69*	580*	589*	759*	774*	789*	804*	819*	834*	849*	863*	895*	1119*	
		1168*	1214*	1246*	1472*	1491*	1509*	1516*	1535*	1542*	1561*	1581*	1597*	1622*	
		1679*	1742*	1785*	1806*	1809*	1819*	1824	2026*	2114*	2118*	2484*	2513*	2621*	

		2627*	2688*	2752*	2797*	2905*	2911*	2985*	3146*	3193	3313*	3332*	3339*	3346*
		3360*	3362*	3395*	3412*	3572*	3574*	3576*	3623*	3686*	3690*	3695*	3699*	3718*
		3721*	3738*	3755*	3786*	3924*	3951*	3954*						
		2784*	2786*	2793*	2795*	2796	2799							
PCT	020344	1298	1338	1385	1431	2435*								
PEKMAX	016252	2372*	2611	2612										
PERHLF	015544	2380*	2615	2616										
PERQRT	015623													
PIRQ =	177772	55*												
PIRQVE =	000240	149*												
PRCNT	020346	2610	2614	2785*										
PRO =	000000	72*												
PR1 =	000040	73*												
PR2 =	000100	74*												
PR3 =	000140	75*												
PR4 =	000200	76*												
PR5 =	000240	77*												
PR6 =	000300	78*												
PR7 =	000340	79*												
PS =	177776	52*	53	556*	3946*									
PSW =	177776	53*												
PWRMSG	022342	3192	3199*											
PWRVEC =	000024	144*	505*	506*	3161*	3162*	3171*	3177*	3189*	3190*				
QRTOK	016332	2459*	2529*	2550*	2613									
RBEG	001642	471	474*											
RBEG2	002414	580*	1788											
ROCHR =	104410	3044	3461*											
RDLIN =	104411	3085	3462*											
RDOCT =	104412	3463*	3984	4003										
RD1	024746	3722*	3759*											
READ	016656	2504	2522*											
RESVEC =	000010	139*												
RMSMAX	016250	1278	1318	1362	1408	2434*								
RSLT0	016052	1639	1656	1657	1663*	1664*	1665*	1666*	1670*	1671*	1672*	1673*	1674*	1676*
		1678	1763	2410*										
RSLT1	016110	1703	1720	1721	1727*	1728*	1729*	1730*	1734*	1735*	1736*	1737*	1738*	1740*
		1741	1767	2415*										
RSLT2	016146	1086	1178	2420*										
RSLT3	016204	1135	1182	2425*										
RSVMSG	017700	2633	2690*											
RSV1	017702	2622*	2691*											
RSV2	017704	2623*	2692*											
RSV3	017705	2624*	2693*											
RSX1	017707	2628*	2694*											
RSX2	017710	2629*	2695*											
RSX3	017711	2630*	2696*											
RSX4	017712	2631*	2697*											
RTCCSR	025420	3942	3945*	3949*	3960*									
RO =	%000000	60*	479	481*	484*	485	490*	579*	590	608*	609*	610	613*	615*
		1638*	1641	1642	1652	1656*	1657*	1658*	1659*	1660*	1661*	1664	1702*	1705
		1706	1716	1720*	1721*	1722*	1723*	1724*	1725*	1728	1787*	1816*	1819	1845
		1855*	1859	1875	1876	1889*	1916	1939*	1965*	2006*	2052*	2054	2102	2104*
		2106*	2108*	2109	2112*	2117*	2486	2525*	2536*	2537	2539*	2540	2542	2544
		2546	2548	2556	2570*	2571	2607*	2608*	2609*	2613*	2786	2787*	2788*	2789*
		2790*	2791*	2792*	2793	2794*	2795	2798*	2801*	2802*	2803*	2804*	2807*	2808*

R1	=%000001	2809*	2810*	3082	3086*	3089	3105*	3118	3119*	3120*	3127*	3128*	3129*	3130*
		3131*	3132	3137	3142*	3144*	3148	3150	3163	3188*	3306	3307*	3312	3317
		3320*	3372	3380*	3384	3385	3387*	3388*	3389	3411*	3429	3430*	3431	3432*
		3433*	3434*	3435*	3636	3637*	3648*	3651	3659*	3683	3684*	3685*	3687	3688
		3700*	3715	3716*	3717*	3719	372*	3757*	3810	3813*	3829*	3852	3855*	3873*
		61*	480	482*	486	489*	582*	583*	584*	585	614*	615	616*	617
		1085*	1088	1091	1105	1122*	1123	1134*	1137	1140	1154	1171*	1172	1662*
		1665*	1666	1667	1668	1675*	1676*	1678	1681	1685	1693	1694	1726*	1729*
		1730	1731	1732	1739*	1740*	1741	1744	1748	1756	1757	1846	1859*	1860
		1864	1888*	1917	1938*	1953*	1967	2005*	2013	2014*	2022*	2025*	2053*	2058*
		2103	2105*	2106	2111*	2116*	2116*	2735*	2749*	2750	3083	3087*	3091*	3095*
		3098*	3101	3104*	3164	3187*	3373	3410*	3514	3515*	3516*	3519*	3567*	3568
		3573	3575	3602*	3771	3772*	3775*	3781*	3785*	3811	3814*	3819*	3822	3828*
		3853	3856*	3862*	3865	3872*								
R2	=%000002	62*	558*	560	565*	581*	583	585*	586	587	1086*	1116*	1135*	1165*
		1639*	1651*	1681*	1682*	1683*	1684	1685*	1686*	1687*	1688*	1689*	1690*	1691
		1703*	1715*	1744*	1745*	1746*	1747	1748*	1749*	1750*	1751*	1752*	1753*	1754
		1847	1858*	1862*	1865	1872*	1873*	1874	1879*	1887*	1918	1934*	1942	1971
		1974	1976*	2004*	2018*	2020*	2619*	2620*	2626*	2738*	2744*	2749	2796*	3084
R3	=%000003	3088*	3092*	3094*	3096*	3102	3103*	3165	3186*		1857*	1871*	1874*	1883*
		63*	559*	566*	568	572*	573	1848	1856*		1871*	1874*	1883*	1884*
		1886*	1919	1928*	1930*	1933*	1967	1989*	1991*	2003*	2536	2576	2626	3040
		3041*	3042	3045*	3046	3050	3052	3054*	3056*	3166	3185*	3241	3250*	3256*
		3257*	3260*	3265*	3266*	3267	3276*							
R4	=%000004	64*	1093*	1107*	1142*	1156*	1196*	1203*	1228*	1235*	1270*	1290*	1310*	1330*
		1353*	1376*	1399*	1422*	1462*	1481*	1499*	1525*	1552*	1572*	1645*	1709*	1912
		1913	1914	1915	2007*	3167	3184*	3242	3244*	3245*	3246*	3247	3248*	3262
R5	=%000005	3264*	3272*	3275*	3635	3660*	3665*							
		65*	563*	669*	672*	674*	681*	684*	686*	694*	697*	699*	711*	715*
		718*	720*	723*	726*	733*	743*	745*	753*	755*	770*	785*	800*	815*
		830*	845*	859*	877*	884*	887*	890*	894*	922*	923*	938*	941*	942*
		957*	958*	973*	974*	990*	991*	1007*	1008*	1024*	1025*	1041*	1042*	1052*
		1059*	1091*	1102*	1105*	1140*	1151*	1154*	1177*	1181*	1196*	1228*	1276	1277
		1296	1297	1316	1317	1336	1337	1352*	1359	1376*	1382	1399*	1405	1422*
		1428	1440*	1444*	1448*	1452*	1552*	1572*	1592*	1593*	1603*	1606*	1617*	1618*
		1628*	1631*	1645*	1709*	1762*	1766*	1849	1851*	1853*	1860*	1864*	1879	1885*
		1938*	1942*	1945*	1948*	1951*	1961*	1965*	1971*	1974*	1981*	1985*	1998*	1999*
		2003*	2043	2047	2050*	2052	2053	2060*	2063	2071*	2074*	2077*	2080*	2084*
		2093*	2465	2466	2467	2468	2472*	2476*	2478*	2486	2490*	2494*	2497*	2499*
		2503*	2507*	2510*	2513*	2514*	2518*	2581*	2582*	2610*	2614*	2634*	2635*	2661*
		2662*	2669*	2672*	2675*	2680*	2683*	2687*	2785	2805*	2811*	3168	3183*	3243
		3249*	3251*	3253*	3254*	3255*	3256	3274*	3507*	3529*	3583*	3603*	3610*	3661*
		3701*	3724*	3741*	3758*	3825*	3830*	3833*	3867*	3874*	3878*	3890	3892	3897*
		3898*	3915*	3989*	4006*	4041*	4048*	4055*						
R6	=%000006	66*	493*	494*	495									
R7	=%000007	67*												
SAR	011716	1093	1107	1142	1156	1196	1203	1228	1235	1270	1290	1310	1330	1353
		1376	1399	1422	1462	1481	1499	1525	1552	1572	1645	1709	1912*	
SARCHN	016312	1913*	1920	1922	1954	1957	2451*							
SDELAY	025336	3941*												
SKIPST	017170	1065	2534*	2555*	2584*	2591								
SKPMSG	015366	2352*	2594											
*P	=%000006	68*	479*	480*	489	490	497*	515*	523*	527	557*	596*	905*	909*
		1252*	1256*	1813*	1845*	1846*	1847*	1848*	1849*	1850*	1851	1854*	1867	1869*

























ADC	1661	1673	1725	1737	2092										
ADD	565	615	616	1664	1666	1683	1690	1728	1730	1746	1753	1778	1779	1864	1942
	2020	2027	2084	2113	2494	2507	2514	2516	2970	2979	3098	3131	3236	3246	3321
	3381	3393	3405	3897	3917	4047									
ASL	1122	1171	2620	2993	2994	2995	3091	3093	3095	3128	3129	3130	3433		
ASLB	1869														
ASR	1658	1659	1660	1670	1671	1672	1666	1687	1688	1722	1723	1724	1734	1735	1736
	1749	1750	1751	1976	2087	2088	2089	2090	2091	2787	2788	2789	2790	2791	2792
	2794	3388													
BCC	1870	2489													
BEQ	531	539	545	591	594	675	687	700	736	900	1066	1070	1175	1250	1438
	1760	1777	1817	1955	1975	2485	2504	2523	2561	2641	2643	2800	2844	2846	2848
	2852	2861	2894	2897	2920	2923	2950	2977	2992	3090	3133	3138	3151	3263	3311
	3324	3359	3375	3379	3399	3401	3523	3548	3569	3597	3654	3774	3780	3816	3818
	3858	3861	3891	3987											
BGE	946	978	995	1046	1076	1280	1300	1320	1340	1364	1387	1410	1433	2864	
BGT	1808	1878	2538	2541	2543	2989	3030	3270							
BHI	2850														
BIC	694	733	734	1682	1689	1745	1752	1805	2570	2946	2963	2990	3017	3023	3031
	3097	3260													
BICB	3983														
BIS	681	1872	1873	2021	2997	3265	3266	3510	3640	3642	3650	3685	3717	3945	
BISB	3120	4052	4053	4054											
BIT	899	1174	1249	1437	1759	2522	2560	2579	2829	2843	2851	2858	2896	2903	2919
	3522	3578	3735	3752	3779										
BITB	530	3310	3315	3347	3378										
BLE	927	962	1012	1029	2110										
BLO	2545														
BLOS	3043														
BLT	1861	1877	1952	1958	2547	2549	2987	3028	3271	3338					
BMI	1868	1993	2487	2557	2636	3896									
BNE	487	496	520	537	543	547	564	569	588	611	618	881	1124	1173	1669
	1695	1733	1758	1866	1966	1982	2024	2059	2086	2482	2500	2520	2563	2578	2580
	2645	2654	2737	2830	2859	2904	2909	2927	2942	2948	2968	2975	2982	3019	3025
	3047	3053	3121	3143	3181	3261	3309	3314	3318	3326	3334	3348	3355	3377	3383
	3386	3403	3517	3520	3541	3543	3550	3552	3579	3595	3622	3646	3652	3736	3753
	3776	3782	3916	3948	3953										
BPL	727	730	891	1607	1632	1653	1717	1852	1882	1927	1932	1988	2019	2081	2107
	2676	2684	2751	2916	2944	2960	3015	3021	3259	3303	3352	3666	3943	3980	
BR	471	522	549	552	567	600	604	760	775	790	805	820	835	850	864
	896	902	1120	1125	1169	1185	1215	1217	1247	1260	1473	1492	1510	1517	1536
	1543	1562	1582	1598	1623	1680	1743	1781	1863	1880	1929	1977	1990	2028	2521
	2551	2554	2639	2647	2649	2743	2832	2838	2841	2854	2857	2914	2971	2998	3000
	3026	3049	3099	3126	3153	3173	3197	3237	3252	3273	3305	3331	3341	3350	3357
	3369	3391	3484	3500	3503	3537	3554	3558	3562	3591	3618	3657	3669	3732	3749
	3777	3783	3790	3795	3820	3835	3863	3880	3894	3922	3976	3995	4000	4009	
CLR	469	470	488	494	508	509	529	555	556	559	575	584	1064	1514	1540
	1559	1579	1654	1663	1674	1675	1718	1727	1738	1739	1802	1803	1855	1858	1921
	1925	1935	1938	1956	2000	2067	2469	2479	2480	2510	2525	2529	2530	2531	2532
	2533	2534	2535	2589	2677	2856	2871	2957	2958	3087	3088	3119	3179	3250	3506
	3515	3601	3638	3639	3643	3647	3655	3772	3814	3832	3856	3877	3893	3946	3949
	3973														
CLRB	1884	2518	2855	3054	3330	3356	3407	3408	3409						
CMP	486	495	519	538	546	587	610	617	674	686	699	735	926	945	961

	977	994	1011	1028	1045	1075	1123	1172	1279	1299	1319	1339	1363	1386	1409
	1432	1668	1694	1732	1757	1876	1951	1967	1974	2109	2481	2486	2503	2537	2540
	2542	2544	2546	2548	2640	2642	2644	2653	2799	2839	2863	2926	2941	2947	2967
	2974	2986	2988	3018	3024	3027	3029	3042	3504	3568	3645	3651	3653	3817	3860
	3890														
CMPB	544	2845	2849	2908	2949	2981	3046	3052	3308	3323	3325	3333	3354	3358	3376
	3540	3542	3547	3594	3621	3773	3966								
DEC	572	729	880	1102	1151	1780	1806	1965	2058	2085	2519	2577	2635	3127	3958
DECb	3258	3269	3337	3340											
EMT	44														
HALT	172	570	2917	2928	3172	3196	3304	3667	3794						
INC	484	536	566	612	720	884	1804	1862	1923	1953	1958	1986	2017	2023	2074
	2550	2552	2553	2555	2558	2559	2564	2575	2651	2669	2862	2899	2996	3180	3264
	3272	3406	3498	3516	3549	3551	3599	3644	3656	3775	3781	3819	3862	3895	
INCB	2488	2539	2867	2893	3360	3519	3570	3665	3952						
IOT	45														
JMP	176	177	178	571	592	595	1786	1788	1824	1994					
JSR	563	580	589	669	672	674	681	684	686	694	697	699	711	715	718
	720	723	726	733	743	745	753	755	759	770	774	785	789	800	804
	815	819	830	834	845	849	859	863	877	884	887	890	894	895	922
	923	938	941	942	957	958	973	974	990	991	1007	1008	1024	1025	1041
	1042	1052	1059	1091	1093	1102	1105	1107	1119	1140	1142	1151	1154	1156	1168
	1177	1181	1196	1203	1214	1228	1235	1246	1270	1290	1310	1330	1352	1353	1376
	1399	1422	1440	1444	1448	1452	1462	1472	1481	1491	1499	1509	1516	1525	1535
	1542	1552	1561	1572	1581	1592	1593	1597	1603	1606	1617	1618	1622	1628	1631
	1645	1679	1709	1742	1762	1766	1785	1819	1938	1942	1945	1948	1951	1961	1965
	1971	1974	1981	1985	1998	2003	2071	2074	2077	2080	2084	2472	2476	2478	2484
	2494	2497	2499	2503	2507	2510	2513	2610	2614	2621	2627	2669	2672	2675	2680
	2683	2687	2797	2905	2911	2985	3313	3332	3339	3346	3395	3507	3529	3572	3574
	3576	3583	3610	3686	3690	3695	3699	3718	3721	3724	3738	3741	3755	3825	3833
	3867	3878	3915	3989	4006	4041	4048								
MOV	472	473	479	480	481	482	485	489	490	493	497	499	500	501	502
	503	504	505	506	507	511	512	515	516	517	518	523	525	526	527
	532	557	558	560	573	574	576	577	578	579	581	582	583	585	596
	608	609	613	614	665	666	669	678	691	707	708	715	728	740	749
	750	757	758	768	772	773	773	783	784	787	788	798	799	802	803
	813	814	817	818	828	829	832	833	843	844	847	848	857	858	861
	862	871	872	877	878	879	894	905	909	918	919	922	925	934	935
	938	941	944	953	954	957	960	969	970	973	976	986	987	990	993
	1003	1004	1007	1010	1020	1021	1024	1027	1037	1038	1041	1044	1049	1057	1065
	1069	1073	1074	1084	1085	1086	1088	1091	1099	1105	1113	1114	1116	1117	1118
	1133	1134	1135	1137	1140	1148	1154	1162	1163	1165	1166	1167	1192	1193	1202
	1209	1211	1212	1213	1224	1225	1234	1241	1243	1244	1245	1252	1256	1268	1276
	1277	1278	1288	1296	1297	1298	1308	1316	1317	1318	1328	1336	1337	1338	1348
	1349	1359	1361	1362	1372	1382	1384	1385	1395	1405	1407	1408	1418	1428	1430
	1431	1461	1468	1469	1470	1471	1480	1487	1488	1489	1490	1498	1505	1506	1507
	1508	1515	1524	1531	1532	1533	1534	1541	1549	1557	1560	1569	1577	1580	1588
	1589	1592	1595	1596	1613	1614	1617	1620	1621	1637	1638	1639	1541	1642	1651
	1655	1656	1662	1678	1681	1685	1701	1702	1703	1705	1706	1715	1719	1720	1726
	1741	1744	1748	1775	1782	1783	1784	1787	1809	1813	1816	1845	1846	1847	1848
	1849	1850	1851	1856	1859	1879	1885	1886	1887	1888	1889	1891	1892	1912	1913
	1914	1915	1916	1917	1918	1919	1924	1928	1930	1933	1934	1939	1945	1985	1989
	1991	2003	2004	2005	2006	2012	2013	2014	2015	2016	2025	2043	2047	2052	2053
	2054	2063	2064	2066	2068	2102	2103	2104	2105	2111	2112	2116	2117	2465	2466





	1324	1326	1328	1329	1340	1344	1346	1348	1349	1364	1368	1370	1372	1373	1387
	1391	1393	1395	1396	1410	1414	1416	1418	1419	1433	1438	1457	1459	1461	1462
	1473	1476	1478	1480	1481	1492	1494	1496	1498	1499	1510	1517	1520	1522	1524
	1525	1536	1543	1545	1547	1549	1550	1562	1565	1567	1569	1570	1582	1584	1586
	1588	1589	1609	1611	1613	1614	1633	1635	1637	1638	1695	1697	1699	1701	1702
	1758	1760	1771	1773	1775	1776	1793	1794	1795	1796	1797	1798	1799	1801	1807
	1810	1812	1816	1818	1824	1826	1827	1834	2561	2814	2817	2822	2828	2829	2841
	2843	2844	2845	2847	2848	2849	2858	2860	2868	2870	2875	2876	2877	2879	2882
	2893	2896	2903	2905	2906	2908	2915	2919	2926	2930	2931	2933	2935	2936	2937
	2965	3004	3005	3033	3041	3042	3046	3047	3063	3064	3070	3072	3075	3087	3111
	3126	3142	3159	3169	3170	3175	3182	3183	3191	3193	3195	3199	3210	3287	3308
	3366	3368	3371	3398	3413	3423	3429	3433	3444	3453	3454	3455	3456	3457	3458
	3460	3461	3462	3463	3464	3501	3531	3537	3555	3559	3563	3585	3591	3612	3618
	3726	3732	3743	3749	3791	3962	3977	4001							
. IFF	42	161	164	165	166	183	187	189	194	196	203	216	219	222	250
	261	264	497	537	539	662	663	664	665	675	687	701	704	705	706
	707	736	746	747	748	749	761	765	766	767	768	776	780	781	782
	783	791	795	796	797	798	806	810	811	812	813	821	825	826	827
	828	836	840	841	842	843	851	854	855	856	857	865	868	869	870
	871	896	901	915	916	917	918	928	931	932	933	934	947	950	951
	952	953	963	966	967	968	969	979	983	984	985	986	996	1000	1001
	1002	1003	1013	1017	1018	1019	1020	1030	1034	1035	1036	1037	1046	1054	1055
	1056	1057	1058	1066	1070	1077	1081	1082	1083	1084	1120	1124	1126	1130	1131
	1132	1133	1169	1173	1176	1186	1189	1190	1191	1192	1216	1218	1221	1222	1223
	1224	1247	1251	1261	1265	1266	1267	1268	1281	1285	1286	1287	1288	1301	1305
	1306	1307	1308	1321	1325	1326	1327	1328	1341	1345	1346	1347	1348	1365	1369
	1370	1371	1372	1388	1392	1393	1394	1395	1411	1415	1416	1417	1418	1433	1439
	1458	1459	1460	1461	1474	1477	1478	1479	1480	1493	1495	1496	1497	1498	1510
	1518	1521	1522	1523	1524	1536	1544	1546	1547	1548	1549	1563	1566	1567	1568
	1569	1583	1585	1586	1587	1588	1610	1611	1612	1613	1634	1635	1636	1637	1695
	1698	1699	1700	1701	1758	1761	1772	1773	1774	1775	1776	1794	1798	1802	1807
	1810	1826	1835	2561	2815	2842	2845	2846	2849	2876	2880	2882	2896	2926	2931
	2934	2937	3005	3007	3012	3033	3034	3043	3047	3064	3073	3112	3127	3156	3160
	3176	3191	3211	3288	3367	3424	3430	3531	3537	3585	3591	3612	3618	3726	3732
	3743	3749													
. IFT	554	602	606	904	2857	2906	3007	3012	3091	3107	3108	3502	3556	3560	3564
. IFTF	3792	3978	4002												
	554	602	606	904	2855	2905	2952	3005	3008	3087	3091	3107	3502	3556	3560
	3564	3792	3978	4002											
. IIF	30	35	40	158	159	160	162	165	166	172	260	264	498	501	507
	508	509	511	512	538	1796	1802	1803	1814	1826	1830	2818	2819	2820	2821
	2822	2823	2827	2856	2857	2873	2876	2877	2883	2884	2885	2886	2887	2892	2918
	2926	2931	2934	2955	3056	3064	3070	3124	3149	3364	3452	3453	3454	3455	3456
	3458	3460	3461	3462	3463	3994									
. IRP	474	661	703	745	764	779	794	809	824	839	853	867	914	930	949
	965	982	999	1016	1033	1053	1080	1129	1188	1220	1264	1284	1304	1324	1344
	1368	1391	1414	1457	1476	1494	1520	1545	1565	1584	1609	1633	1697	1771	1845
	1885	2828	3082	3103	3163	3169	3182	3183	3372	3373	3394	3410	3411		
. LIST	14	30	150	165	172	250	252	253	254	261	264	474	513	538	541
	554	563	602	606	661	665	669	672	674	681	684	686	694	697	699
	703	707	711	715	718	720	723	726	733	743	745	749	753	755	764
	768	779	783	794	798	809	813	824	828	839	843	853	857	867	871
	877	884	887	890	894	904	914	918	922	930	934	938	941	949	953
	957	965	969	973	982	986	990	999	1003	1007	1016	1020	1024	1033	1037

	1041	1052	1053	1057	1080	1084	1091	1102	1105	1129	1133	1140	1151	1154	1188
	1192	1196	1220	1224	1228	1264	1268	1284	1288	1304	1308	1324	1328	1344	1348
	1352	1368	1372	1376	1391	1395	1399	1414	1418	1422	1457	1461	1476	1480	1494
	1498	1520	1524	1545	1549	1552	1565	1569	1572	1584	1588	1592	1603	1606	1609
	1613	1617	1628	1631	1633	1637	1645	1697	1701	1709	1771	1775	1802	1818	1938
	1942	1945	1948	1951	1961	1965	1971	1974	1981	1985	1998	2003	2071	2074	2077
	2080	2084	2472	2476	2478	2494	2497	2499	2503	2507	2510	2513	2669	2672	2675
	2680	2683	2687	2822	2926	3033	3444	3452	3453	3454	3455	3456	3457	3458	3459
	3460	3461	3462	3463	3464	3502	3556	3560	3564	3792	3915	3978	4002		
.MACRO	17	18	19	20	22	24	25	26	27	28	29	30	166	213	451
.MCALL	529	3444													
.NLIST	30	150	261	513	541										
	14	30	150	165	172	250	252	253	254	261	264	474	513	538	541
	554	563	602	606	661	665	669	672	674	681	684	686	694	697	699
	703	707	711	715	718	720	723	726	733	743	745	749	753	755	764
	768	779	783	798	809	809	813	824	828	839	843	853	857	867	871
	877	884	887	890	894	904	914	918	922	930	934	938	941	949	953
	957	965	969	973	982	986	990	999	1003	1007	1016	1020	1024	1033	1037
	1041	1052	1053	1057	1080	1084	1091	1102	1105	1129	1133	1140	1151	1154	1188
	1192	1196	1220	1224	1228	1264	1268	1284	1288	1304	1308	1324	1328	1344	1348
	1352	1368	1372	1376	1391	1395	1399	1414	1418	1422	1457	1461	1476	1480	1494
	1498	1520	1524	1545	1549	1552	1565	1569	1572	1584	1588	1592	1603	1606	1609
	1613	1617	1628	1631	1633	1637	1645	1697	1701	1709	1771	1775	1802	1818	1938
	1942	1945	1948	1951	1961	1965	1971	1974	1981	1985	1998	2003	2071	2074	2077
	2080	2084	2472	2476	2478	2494	2497	2499	2503	2507	2510	2513	2669	2672	2675
	2680	2683	2687	2822	2926	3033	3444	3452	3453	3454	3455	3456	3457	3458	3459
	3460	3461	3462	3463	3464	3502	3556	3560	3564	3792	3915	3978	4002		
.PAGE	213	329													
.PSECT	14														
.REM	1	14													
.REPT	172	252													
.SBTTL	40	154	166	175	180	191	213	261	329	466	491	534	541	619	658
	659	660	661	703	745	764	779	794	809	824	839	853	867	914	930
	949	965	982	999	1016	1033	1053	1080	1129	1188	1220	1264	1284	1304	1324
	1344	1368	1391	1414	1457	1476	1494	1520	1545	1565	1584	1609	1633	1697	1771
	1789	1791	1832	1899	2009	2119	2431	2734	2781	2812	2877	2931	3070	3109	3157
	3208	3285	3364	3421	3444										
.TITLE	30														
.WORD	172	173	174	188	207	208	209	210	211	212	221	224	225	226	227
	230	231	232	233	234	235	236	239	240	241	250	252	253	266	267
	268	269	270	271	272	273	277	278	279	292	296	299	302	303	304
	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319
	320	321	322	323	324	426	429	431	433	435	437	439	441	443	444
	446	448	563	598	669	672	674	681	684	686	694	697	699	711	715
	718	720	723	726	733	743	745	753	755	877	884	887	890	894	922
	938	941	957	973	990	1007	1024	1041	1052	1091	1102	1105	1140	1151	1154
	1196	1228	1352	1376	1399	1422	1552	1572	1592	1603	1606	1617	1628	1631	1645
	1709	1807	1810	1825	1938	1942	1945	1948	1951	1961	1965	1971	1974	1981	1985
	1998	2003	2071	2074	2077	2080	2084	2472	2476	2478	2494	2497	2499	2503	2507
	2510	2513	2669	2672	2675	2680	2683	2687	3107	3135	3140	3192	3194	3284	3314
	3361	3396	3451	3508	3605	3625	3760	3826	3827	3834	3869	3879	3900	3901	3915
	3925	3926	3956	3960	3990	3991	3996	4007	4008	4011	4012	4042	4043	4049	4050

G09

MAINDEC-11-DRLPE-A MACY11 27(654)  
DRLPE.P11 CROSS REFERENCE TABLE

14-DEC-77 20:24 PAGE 102

SEQ 0111

000000

ERRORS DETECTED: 0

H09

MAINDEC-11-DRLPE-A  
DRLPE.P11

MACY11 27(654) 14-DEC-77 20:24 PAGE 103

SEQ 0112

\*DRLPE,DRLPE/SOL/CRF=DRLPA.MAC,DRLPE  
RUN-TIME: 26 17 2 SECONDS  
CORE USED: 40K  
EOF1DRLPESEQ

00010000

780223

PDP10 411

2