

TEST NAME	TEST NO.	TEST DATE	TEST TIME	TEST RESULT	TEST STATUS	TEST COMMENTS	TEST OPERATOR	TEST LOCATION	TEST EQUIPMENT	TEST MATERIALS	TEST NOTES
TEST 1	1	1978	10:00	PASS	OK						
TEST 2	2	1978	10:05	FAIL	ERR						
TEST 3	3	1978	10:10	PASS	OK						
TEST 4	4	1978	10:15	PASS	OK						
TEST 5	5	1978	10:20	PASS	OK						
TEST 6	6	1978	10:25	PASS	OK						
TEST 7	7	1978	10:30	PASS	OK						
TEST 8	8	1978	10:35	PASS	OK						
TEST 9	9	1978	10:40	PASS	OK						
TEST 10	10	1978	10:45	PASS	OK						
TEST 11	11	1978	10:50	PASS	OK						
TEST 12	12	1978	10:55	PASS	OK						

1234567890

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DRLPH-A-D
PRODUCT NAME: LPA/LPS DIAGNOSTIC TEST
DATE: JANUARY 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.

1. ABSTRACT

THIS DIAGNOSTIC TESTS AND EXERCISES THE "LPS11".
 WHEN LOADED WILL TYPE OUT THE PROGRAM TITLE.
 A MESSAGE IS THEN PRINTED GIVING THE LETTER DESIGNATORS
 TO BE TYPED TO RUN ANY ONE OF THE FIVE (5) SEPERATE TESTS OF WHICH
 THIS PROGRAM IS COMPRISED. THE PROGRAM THEN TYPES A 'CR .' AND
 THEN WAITS IN A KEYBOARD MONITOR MODE FOR A LETTER TO BE TYPED.
 ALTHOUGH THESE TESTS MAY BE RUN IN ANY ORDER IT IS IMPERATIVE
 THAT THE 'LOGIC' TESTS ARE RUN FIRST AND PROVED FULLY OPERATIONAL.

THE PROGRAM IS SET UP TO GIVE THE OPERATOR AS MUCH CONTROL
 OVER THE PROGRAM AS POSSIBLE VIA THE TELETYPE. TYPING A 'I C'
 (OBTAINED VIA TYPING THE 'CNTR' AND 'C' KEYS SIMULTANEOUSLY)
 WHILE RUNNING ANY TEST WILL ENABLE THE PROGRAM TO RETURN TO
 THE KEYBOARD MONITOR AND AWAIT A NEW LETTER DESIGNATOR TO BE
 TYPED. TYPING A 'I A' WHILE IN MONITOR MODE WILL ENABLE THE LETTER
 DESIGNATORS TO BE RETYPED. IF RUNNING ON A NON-SWITCH REGISTER CPU,
 TYPING A 'CTRL G' WILL ALLOW THE CHANGING OF A SOFTWARE SWITCH REGISTER.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZLPC-C".
 IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE LPS11
 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 "DZLPC". YOU SHOULD RUN "MD-11-DRLPA" BEFORE
 RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 15.

2. REQUIREMENTS (EQUIPMENT)

- A. PDP-11 COMPUTER WITH 16K OF MEMORY
- B. TELETYPE (OR EQUIVALENT)
- C. LPS11 OPTION BOX WITH:
 - LPSAD12 SIMPLE A TO D CONTROL AND
 - LPSADNP DMA A TO D CONTROL AND/OR
 - LPS5H DUAL SAMPLE AND HOLD CONTROL.

3. LOADING PROCEDURE

- A. USE STANDARD PROCEDURE FOR LOADING BINARY TAPES.

4. STARTING PROCEDURE

THE STARTING ADDRESS OF THIS DIAGNOSTIC IS 200

5. CONSOLE SWITCH SETTINGS

- A. ALL SWITCHES SHOULD BE DOWN (0) WHEN THE PROGRAM IS STARTED.
- B. REFER TO THE INDIVIDUAL TEST DESCRIPTIONS FOR APPLICABLE CONSOLE SWITCH SETTINGS
- C. REFER TO 14. FOR SOFTWARE SWITCH REGISTER OPERATION.

* TYPE 'CARRIAGE RETURN' (CR) TO TERMINATE ALL INPUT DATA.

6. A TO D LOGIC TEST

- A. THE LOGIC TEST CONSISTS OF 26 SUBTESTS WHICH CHECK OUT THE 'LPS A TO D' LOGIC. EACH TEST IS LOOPED '2048' TIMES TO TEST LOGIC RELIABILITY.

B. STARTING SEQUENCE

- 1. TYPE 'A' TO RUN THE 'A TO D LOGIC' TEST.
- 2. THE PROGRAM WILL THEN EXECUTE THE 'A TO D LOGIC' TEST.

C. CONTROL SWITCHES

1. ↑C (CONTROL C)

TYPING A ↑C AT ANY TIME WILL ENABLE THE PROGRAM TO EXIT THE 'A TO D LOGIC' TEST AND RETURN TO THE MONITOR.

2. CONSOLE SWITCH

FUNCTION

CONSOLE SW11=0	NORMAL RUN (2048 PASSES/TEST)
CONSOLE SW11=1	SUPPRESS SUBPROGRAM ITERATIONS
CONSOLE SW13=0	PRINT ERROR MESSAGE
CONSOLE SW13=1	INHIBIT ERROR MESSAGES
CONSOLE SW14=1	RUN SCOPE MODE
CONSOLE SW14=0	INHIBIT SCOPE MODE
CONSOLE SW15=0	CONTINUE AFTER ERROR
CONSOLE SW15=1	HALT ON ERROR

D. LOGIC ERRORS

ON ENCOUNTERING AN ERROR (DATA SWITCHES DOWN) THE ERROR ADDRESS AND THE CONTENTS OF THE A/D STATUS REGISTER AND A/D BUFFER AND DMA STATUS REGISTER ARE TYPED OUT.

E. RESTRICTIONS

DO 'NOT' CONNECT EXTERNAL SYNC FOR THE LOGIC TEST.

F. TEST TIME

IT TAKES APPROXIMATELY 5 SEC. TO RUN THE LOGIC TEST AND TYPE "END PASS". FURTHER ITERATIONS TAKE APPROXIMATELY 5 MINUTES.

7. A TO D CALIBRATION TEST

A. THE 'A/D CALIBRATION' TEST IS DESIGNED TO ACCEPT AN INPUT FROM THE TELETYPE TO INDICATE THE TYPE OF SYNC (EXTERNAL, INTERNAL OR LPS CLOCK TO BE USED AND THEN TAKES CONTINUOUS CONVERSIONS USING THE 'CH.' SELECTED VIA THE CONSOLE SWITCHES. THESE SETTINGS MAY BE CHANGED AT ANY TIME.

B. STARTING SEQUENCE

1. TYPE 'C' TO RUN THE A/D CALIBRATION TEST.
2. THE TEST HEADER PLUS A REQUEST FOR A SYNC TYPE WILL THEN BE TYPED.
3. TYPE IN THE DESIRED SYNC, 'I' FOR INTERNAL, 'E' FOR EXTERNAL 'C' FOR LPS CLOCK (IF INSTALLED) FOLLOWED BY 'CR'.
4. THE TEST WILL START.

C. CONTROL SWITCHES1. ↑A (CONTROL A)

TYPING ↑A WILL ENABLE A NEW SYNC TYPE TO BE ENTERED.

2. ↑C (CONTROL C)

TYPING ↑C WILL CAUSE THE PROGRAM TO EXIT THE CALIBRATION TEST AND RETURN TO THE MONITOR.

3. CONSOLE SWITCH FUNCTION

CONSOLE SW '0-5'	CHANNEL SELECT
CONSOLE SW '10=0'	DISPLAY VALUE IN LED DISPLAY
CONSOLE SW '10=1'	PRINT CONVERSION VALUE

D. CALIBRATION ERRORS

THE PRINTOUT 'ERROR BIT SET' WILL OCCUR WHILE RUNNING WITH EXTERNAL SYNC IF THE SYNC FREQUENCY IS TOO FAST.

8. A TO D REPEATABILITY TEST

A. THIS TEST REQUESTS A CH.(S) AND A COUNT SPREAD OF '1-4' AND MODE OF OPERATION TO BE TYPED IN BY THE OPERATOR. A SERIES OF '512' CONVERSIONS ARE THEN TAKEN ON THE INPUT CH.(S) CONVERSIONS ARE THEN AVERAGED OUT AND IF THE COUNT SPREAD IS FOUND TO BE GREATER THAN REQUEST, THE RESULTS OF THE CONVERSIONS ARE TYPED OUT. A SINGLE CHANNEL OR A SERIES OF CHANNELS MAY BE TESTED VIA TYPING EITHER 'N(CR)' TO SELECT A SINGLE CHANNEL OR 'N,N(CR)' TO TEST A SERIES OF CHANNELS.

B. STARTING SEQUENCE

1. TYPE 'D' TO RUN THE 'REPEATIBILITY' TEST.
2. A REQUEST IS THEN MADE FOR CH.(S) TO BE TESTED AND COUNT SPREAD (RANGE IN WHICH ALL 512 COUNTS MUST FALL FOR THE CH. TO BE CONSIDERED ACCEPTABLE).
3. A REQUEST IS THEN MADE FOR MODE OF OPERATION (NPR OR BR) TYPE 'I' FOR PROGRAM SAMPLING OR TYPE 'B' FOR BURST NPR OR 'D' FOR DUAL MODE SAMPLING IF INSTALLED.
4. IF THE CHANNEL IS FOUND TO BE WITHIN THE SELECTED COUNT SPREAD, THE PROGRAM WILL EITHER CONTINUE TO THE NEXT CHANNEL IF SELECTED OR RETEST THE CURRENT CHANNEL.

C. CONTROL SWITCHES

1. ↑A (CONTROL A)

TYPING A ↑A WHILE THE PROGRAM IS RUNNING WILL ENABLE A NEW CH.(S) AND COUNT SPREAD TO BE SELECTED.

2. ↑C (CONTROL)

TYPING ↑C WILL CAUSE THE PROGRAM TO EXIT THE 'REPEATIBILITY' TEST AND RETURN TO THE MONITOR.

3. CONSOLE SWITCHES

FUNCTION

CONSOLE SW 10=0	PRINT ERRORS ONLY
CONSOLE SW 11=1	PRINT OUT ALL CONVERSIONS
CONSOLE SW 12=0	PRINT ERRORS
CONSOLE SW 13=1	INHIBIT ERROR PRINTOUTS
CONSOLE SW 14=0	DO NOT HALT UPON REPEATIBILITY REPORT
CONSOLE SW 15=1	HALT UPON REPEATIBILITY REPORT

D. REPEATIBILITY ERRORS

ON ENCOUNTERING AN ERROR (CONSOLE SWITCHES DOWN) THE ERROR DATA IS TYPED OUT.

1. ERROR FORMAT

CH.		HI		AV		LO							
A		B		C		D							
LO	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	HI	
E	F	G	H	I	J	K	L	M	N	O	P	Q	

WHERE:

- A=CHANNEL BEING TESTED
- B=THE HIGHEST READING OF THE '512' CONVERSIONS
- C=THE AVERAGE READING OF THE '512' CONVERSIONS
- D=THE LOWEST READING OF THE '512' CONVERSIONS
- E=NUMBER OF COUNTS 'OUT OF RANGE' LOWER THAN 5 COUNTS
- F-J=NUMBER OF COUNTS IN EACH PART LOWER THAN AVERAGE.
- K=NUMBER OF COUNTS AT AVERAGE OF THE '512'
- L-P=NUMBER OF COUNTS IN EACH PART HIGHER THAN AVERAGE.
- Q=NUMBER OF COUNTS 'OUT OF RANGE' HIGHER THAN 5 COUNTS

E. RESTRICTIONS

- 1. IF A SECOND CH. IS ENTERED, IT MUST BE LARGER THAN THE FIRST CH. OR THE INPUT WILL NOT BE ACCEPTED.

9. A TO D RECOVERY TEST

A. THE "RECOVERY TEST" IS DESIGNED TO DETERMINE THE RECOVERY CAPABILITY OF THE 'LPS A TO D'. THE TEST REQUESTS FOR TWO (2) CH. INPUTS TO BE TYPED IN. THE TEST THEN TAKES A SERIES OF SIXTEEN (16) CONVERSIONS (8 ON EACH CH.) AND THEN TYPES OUT THE 'B' CONVERSION VALUES IN THE ORDER THEY WERE TAKEN ON THE SECOND CH.

B. STARTING SEQUENCE

1. TYPE 'E' TO RUN THE RECOVERY TEST.
2. A REQUEST IS THEN MADE FOR THE CH.S TO BE TESTED.
3. TYPE 'N,N (CR)' WHERE 'N' IS ANY CH.
4. THE PROGRAM WILL THEN TAKE CONTINUOUS CONVERSIONS TYPING OUT THE CONVERSION VALUES FOR THE SECOND CH.

EXAMPLE:

CH. A XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX

WHERE:

A=TO THE SECOND CH.
X=TO THE 'B' CONVERSIONS TAKEN ON THAT CH.

C. CONTROL SWITCHES

1. ↑A (CONTROL A)

TYPING A '↑A' WILL ENABLE A NEW SET OF CH.S BE ENTERED.

2. ↑C (CONTROL C)

TYPING A '↑C' WILL ENABLE THE PROGRAM TO RETURN TO THE MONITOR.

3. CONSOLE SWITCHES FUNCTION

CONSOLE SW 10 = 0	PRINT RECOVERY REPORT
CONSOLE SW 10 = 1	INHIBIT PRINTING OF RECOVERY REPORT

D. RESTRICTIONS

DO NOT EXECUTE THIS TEST IF THE WIDE BAND JUMPERS HAVE BEEN REMOVED.

10. DUAL MODE DYNAMIC LOGIC TEST

A. THE 'DUAL MODE' TEST IS DESIGNED TO TEST THE DYNAMIC DUAL SAMPLE AND HOLD LOGIC OF THE LPS. A SAMPLE IS TAKEN FROM A CHANNEL PAIR (IE. CHANNEL 0 AND 10) NOT IN DUAL MODE. ANOTHER SAMPLE IS TAKEN FROM A CHANNEL PAIR USING 'DUAL MODE'. THE RESULTING CONVERSIONS ARE THEN COMPARED FOR VALUE AND SIMILARITY.

B. STARTING SEQUENCE

1. TYPE 'F' TO RUN THE 'DUAL MODE LOGIC TEST'.
2. THE PROGRAM WILL THEN EXECUTE THE DUAL MODE TEST ON CHANNEL PAIRS 0-3.

C. CONTROL SWITCHES

1. ↑C (CONTROL C)

TYPING ↑C WILL CAUSE THE PROGRAM TO EXIT TO THE MONITOR.

2. ↑A (CONTROL A)

TYPING ↑A WILL CAUSE THE RESTARTING OF THE DUAL MODE LOGIC TEST.

3. CONSOLE SWITCHES

FUNCTION

CONSOLE SW13=0	PRINT ERROR MESSAGES
CONSOLE SW13=1	INHIBIT ERROR MESSAGES
CONSOLE SW15=0	CONTINUE AFTER ERROR
CONSOLE SW15=1	HALT ON ERROR

11. MANUAL STARTING ADDRESSES

EXTERNAL STARTING ADDRESSES OF THE BASIC TEST HAVE BEEN PROVIDED FOR THE USER TO MANUAL START THESE TESTS.

STARTING ADDRESS -----	BASIC TEST -----
204	A TO D LOGIC TEST
210	DMA LOGIC TEST
214	DUAL SAMPLE LOGIC TEST
220	A TO D CALIBRATION TEST

12. MISC. INFORMATION

THE PROGRAM CAN BE CHAINED UNDER XXDP/ACT-11
IF THE PROGRAM WAS LOADED FROM ACT-11 OR DDP, THE A TO D LOGIC TEST AND (IF INSTALLED) DMA LOGIC TEST WILL BE RUN.
THIS PROGRAM DOES NOT HAVE THE "APT" HOOKS.

13. OPERATOR VARIABLE LOCATIONS

LOCATION 1000 CONTAINS THE LPS STARTING DEVICE ADDRESS <170400>
LOCATION 1002 CONTAINS THE LPS STARTING VECTOR <340>
LOCATION 1004 CONTAINS THE LPS A TO D BR LEVEL <6><300>
LOCATION 1010 CONTAINS THE + OR - 37 COUNT SPREAD FOR THE DUAL SAMPLE TEST.
LOCATION 1012 CONTAINS THE TTY FILLER CHARACTER COUNT
LOCATION 1014 CONTAINS THE TTY FILLER CHARACTER
LOCATION 170 CONTAINS THE SOFTWARE SWITCH REGISTER VALUE
LOCATION 172 CONTAINS THE SOFTWARE DISPLAY REGISTER VALUE.

NOTE: IF LOCATIONS 1000 OR 1002 ARE CHANGED, THE TEST MUST BE RE-INITIALIZED AT 200.

14. SOFTWARE SWITCH REGISTER OPERATION

THE PROGRAM SUPPORTS NON-SWITCH REGISTER CPU TYPES.
THIS IS ACCOMPLISHED BY TYPING A "CTRL G". THE RESPONSE WILL REPORT THE VALUE OF THE OLD SOFTWARE SWITCH REGISTER AND WAIT FOR A NEW VALUE.
THE OPERATOR NOW INPUTS THE NEW VALUE AND TERMINATES IT WITH A "CR".
IF THE OPERATOR TYPES A "CR" WITH NO INPUT, THE SWITCH REGISTER IS SET TO 0.
UPON TERMINATING, THE PROGRAM WILL RESUME RUNNING THE APPROPATE TEST.

15. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY

SEQ 0010

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-DALPA. WHEN THE USER RUNS DALPA HE CAN GENERALLY TELL WHICH OPTION DIAGNOSTIC (LEVEL 3) TO RUN NEXT. M8254 AND M8200-YC ERRORS MAY "LOOK" ALIKE AND DALPA 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 DALPA 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 0011

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	LPA/DR11-K DIAG.
	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 JUMP+ROM READ TEST

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

.REM !

THIS IS A LIST OF TESTS DELETED FROM THIS DIAGNOSTIC.
THESE TESTS COULD NOT BE DONE THROUGH THE LPA-11
TEST THAT A TO D INTERRUPT AT LEVEL INDICATED-1
TEST THAT A TO D DOES NOT INTERRUPT AT LEVEL INDICATED
TEST THAT THE A TO D DOES NOT INTERRUPT AT LEVEL INDICATED +1
ALL DMA TEST DELETED

!

```

100000 BIT15=100000
040000 BIT14=40000
020000 BIT13=20000
010000 BIT12=10000
004000 BIT11=4000
002000 BIT10=2000
001000 BIT9=1000
000400 BIT8=400
000200 BIT7=200
000100 BIT6=100
000040 BIT5=40
000020 BIT4=20
000010 BIT3=10
000004 BIT2=4
000002 BIT1=2
000001 BIT0=1
    
```

; SWITCH REGISTER DEFINITIONS AND FUNCTIONS:

```

100000 SW15=100000           ;=1, HALT ON ERROR
040000 SW14=40000          ;=1, LOOP ON CURRENT TEST
020000 SW13=20000         ;=1, SUPPRESS ERROR TYPEOUT
010000 SW12=10000
004000 SW11=4000          ;=1, SUPPRESS 'SUBPROGRAM' ITERATIONS
002000 SW10=2000         ;=1, FORCE TYPEOUT (REPEATIBILITY)
001000 SW09=1000
000400 SW08=400
000200 SW07=200
000100 SW06=100
000040 SW05=40
000020 SW04=20
000010 SW03=10
000004 SW02=4
000002 SW01=2
000001 SW00=1
    
```

; REGISTER DEFINITIONS

```

000000 R0=%0
000001 R1=%1
000002 R2=%2
000003 R3=%3
000004 R4=%4
    
```



```

84      000005      RS=%5
85      000006      SP=%6
86      000007      PC=%7
87
88      ;LOAD TRAP CATCHER INTO LOC'S 0-1000
89      .=0
90      000000      000000      HALT
91      000002      000000      HALT
92      .=24
93      000024      011232      PWRFAL      ;POWER FAIL HANDLER
94      000026      000340      340
95      .=60
96      000060      010130      XTTYIN      ;TELEPRINTER KEYBOARD ROUTINE
97      000062      000340      340
98      .=30
99      000030      011334      EMTSRV      ;EMT TRAP, EMT DISPATCH SERVICE
100     000032      000340      340
101     000034      012664      LOGERR      ;TRAP TRAP, LOGIC ERROR TRAP
102     000036      000340      340
103     .=46
104     000046      002022      LOGICAL
105     .=52
106     000052      000000      0
107     .=170
108     000170      000000      SCFISW: 0      ;SOFTWARE SWITCH VALUE
109     000172      000000      SOFTDI: 0      ;SOFTWARE DISPLAY VALUE
110     000174      000137      001546      JMP      MONITR ;PROGRAM 'RESTART' ADDRESS
111     000200      000137      001154      JMP      INIT   ;INITIALIZATION ADDRESS
112     000204      000137      002040      JMP      ADTEST ;MANUAL START OF A TO D LOGIC
113     000210      000240      NOP
114     000212      000240      NOP
115     000214      000137      006772      JMP      DSH*ST ;MANUAL START OF DUAL SAMPLE
116     000220      000137      005246      JMP      MANCAL ;MANUAL START OF THE A TO D CALIBRATION
117
118     ;TRAP EQUIVALENCE TABLE:
119
120     104400      ERROR=TRAP      ;LOGIC TEST ERROR ROUTINE
121     104000      PRINT=EMT      ;MESSAGE PRINTER ROUTINE
122     104001      DECCT=EMT+1      ;OCTAL CONVERSIN ROUTINE
123     104002      SCOPE=EMT+2      ;LOGIC TEST SCOPE SUBROUTINE (4000)
124     104003      SCOPE1=EMT+3      ;LOGIC TEST SCOPE SUBROUTINE (10)
125     104004      CMPUTE=EMT+4      ;A/D AVERAGING ROUTINE
126     104005      CATORIZ=EMT+5      ;ROUTINE TO CALCULATE THE COUNT SPREAD
127     104006      BINDEC=EMT+6      ;BINARY TO DECIMAL CONVERSION ROUTINE
128     104007      SPACE=EMT+7      ;TYPE 'N' SPACES
129     104010      PRTOCT=EMT+10      ;OCTAL PRINT ROUTINE
130     104011      TTYIN=EMT+11      ;TELETYPE INPUT ROUTINE
131     104012      PRTAVG=EMT+12      ;GAIN AVERAGE PRINT ROUTINE
132     104013      TSTTKS=EMT+13      ;SUBROUTINE TO TEST FOR KEYBOARD FLAG
133     104014      SCOPED=EMT+14      ;SCOPE SUBROUTINE (1)
134
135     .=1000
136     001000      170400      LPSADD: 170400 ;LPS STARTING DEVICE ADDRESS
137     001002      000340      LPSVCT: 340   ;LPS STARTING DEVICE VECTOR

```

```

138 001004 000300      ADBRL: 300      ;LPS A TO D BR LEVEL
139 001006 000001      PDPDLY: 1      ;CHANGED BY THE PROGRAM
140 001010 000037      AMASK: 37      ;+ OR - 37 COUNTS FOR DUAL SAMPLE LOGIC TEST
141 001012 000002      FILLS: 2       ;NUMBER OF FILLER CHARS AFTER LF
142 001014 000000      FILCHR: 0      ;FILLER CHARACTER
143
144 001016 177776      PSW: 177776
145 001020 177560      TKS: 177560
146 001022 177562      TKB: 177562
147 001024 177564      TPS: 177564
148 001026 177566      TPB: 177566
149 001030 177570      SWR: 177570    ;OR 170 IF NO SWR CPU TYPE
150 001032 177570      DISPLA: 177570 ;OR 172 IF NO SWR CPU TYPE
151 001034 000000      PASSCT: 0
152 001036 004000      ICOUNT: 4000
153
154                      ;ADDRESS OF KMC-11 OF LPA-11  THE ADDR FOR KMADO MAY BE
155                      ;                          CHANGED BY THE USER TO REFLECT
156                      ;                          A DIFFERENT KMC-11 ADDR. THE
157                      ;                          REST OF THE ADDRESSES WILL
158                      ;                          BE CHANGED BY THE PROGRAM.
159
160
161 001040      LPCI:
162 001040 170460      KMADO: .WORD 170460      ;BASE KMC ADDR. MAY BE PATCHED BY USER.
163
164 001042      LPMR:
165 001042 170461      KMAD1: .WORD 170460+1      ;>DO NOT <;KMC-CSR ADDR
166 001044      LPCO:
167 001044 170462      KMAD2: .WORD 170460+2      ;>PATCH <;
168 001046      LPSO:
169 001046 170463      KMAD3: .WORD 170460+3      ;>THIS AREA <
170 001050      LPADL:
171 001050 170464      KMAD4: .WORD 170460+4      ;
172 001052      LPADH:
173 001052 170465      KMAD5: .WORD 170460+5      ;>DO NOT <
174 001054      LPMS1:
175 001054 170466      KMAD6: .WORD 170460+6      ;>PATCH <
176 001056      LPMS2:
177 001056 170467      KMAD7: .WORD 170460+7      ;>THIS AREA <
178
179 001060 000300      VECTOR: .WORD AVECT1&777    ;BASE VECTOR OF KMC
180 001062 000304      VECTPS: .WORD 4+AVECT1&777 ;VECTR ADDR.+2
181
182 001064 000004      VERSN: .WORD 4            ;CURRENT VERSION NUMBER OF MICROCODE.
183
184 001066 000000      .DVLS: .WORD 0            ;/DEVICE LIST OF I/O ADDR. DEFINED
185 001070 000020      .BLKW 16.              ;/BY INIT.
186
187 001130 170400      ADCS: 170400          ;A TO D STATUS/CONTROL REGISTER
188 001132 170401      ADCS1: 170401         ;A TO D STATUS REGISTER <HIGH BYTE>
189 001134 170402      ADDBR: 170402        ;A TO D CONVERTED VALUE <READ>
190
191 001136 170404      CSR: 170404          ;A TO D LED DISPLAY LIGHTS <WRITE>
                        ;CLOCK STATUS REGISTER

```

```

192 001140 170406 CSB: 170406 ;CLOCK PRESET BUFFER
193
194 001142 170436 UADMR: 170436 ;A TO D DIRECT MEMORY ACCESS ADDRESS UNUSED
195
196 001144 000340 ADINT: 340 ;A TO D INTERRUPT VECTOR
197 001146 000342 ADINT1: 342 ;
198 000300 AVECT1= 300 ;
199 001150 000300 $VECT1: .WORD AVECT1
200 001152 000000 DMK2: 0 ;TEMP STORAGE
201
202
203
204 ;THIS ROUTINE IS EXECUTED ON LOADING THE PROGRAM
205 ; OR RESTARTING AT LOCATION 174
206
207 001154 INIT:
208 001154 013706 014774 MOV STACK,SP ;INIT STACK POINTER=1000
209 001160 012777 000340 177630 MOV #340,SPSW
210
211 ;THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
212 ;
213
214 001166 010046 MOV RO, -(SP)
215 001170 010146 MOV R1, -(SP)
216 001172 C 3700 MOV KMADO,RO ;GET KMC-11 ADDRESS.
217 001176 012701 001042 MOV #KMAD1,R1 ;GET ADDR. OF ADDR. LIST.
218
219 001202 005200 64$: INC RO ;UPDATE ADDR.
220 001204 010021 MOV RO,(1)+ ;WRITE ADDR.
221 001206 020127 001060 CMP R1,#KMAD7+2 ;DONE ALL ADDRESSES?
222 001212 001373 BNE 64$ ;NO - DO NEXT ADDR.
223 001214 005037 001066 CLR .DVLS ;CLR ADDR. LIST.
224 001220 012601 MOV (SP)+,R1
225 001222 012600 MOV (SP)+,RO
226
227 001224 004737 021526 JSR PC,$RESET ;CLEAR THE WORLD
228 001230 013737 001000 001130 MOV LPSADD,ADCS ;HOUSEKEEP THE ADDRESS AND VECTOR
229 001236 013737 001000 001132 MOV LPSADD,ADCS1
230 001244 013737 001000 001134 MOV LPSADD,ADDBR
231 001252 013737 001000 001136 MOV LPSADD,CSR
232 001260 013737 001000 001140 MOV LPSADD,CSB
233 001266 005237 001132 INC ADCS1
234 001272 062737 000002 001134 ADD #2,ADDBR
235 001300 062737 000004 001136 ADD #4,CSR
236 001306 062737 000006 001140 ADD #6,CSB
237 001314 013737 001002 001144 MOV LPSVCT,ADINT
238 001322 013737 001002 001146 MOV LPSVCT,ADINT1
239 001330 062737 000002 001146 ADD #2,ADINT1
240
241 ;NOW TEST IF A SWITCH REGISTE EXISTS
242
243 001336 012737 001352 000004 MOV #15,SWR ;LOAD BUS TRAP
244 001344 005777 177460 TST SWR ;TEST IF REAL SWITCH REGISTER
245 001350 000407 BR CPUTYP ;BR IF NO TRAP

```

E02

```

246 001352 022626          1S:  CMP      (SP)+,(SP)+      ;POP STACK
247 001354 012737 000170 001030  MOV      #170,SWR      ;CHANGE POINTER TO LOC. 170
248 001362 012737 000172 001032  MOV      #172,DISPLA   ;
249                                     ; DETERMINE CPU TYPE
250                                     RO=3 IF 11/45
251                                     RO=2 IF 11/40
252                                     RO=1 IF 11/20
253                                     RO=0 IF 11/05
254                                     PIRQ=177772
255                                     PS=177776
256 001370 012737 000002 000006  CPUTYP: MOV      #RTI,@#6
257 001376 012737 000006 000004  MOV      #6,@#4
258 001404 012700 000003          MOV      #3,RO          ;LOAD RO
259 001410 000261          SEC              ;SET C BIT
260 001412 005737 177772          TST      @#PIRQ       ;TEST PIRQ
261 001416 005600          SBC      RO
262 001420 000261          SEC
263 001422 105737 177777          TSTB     @#PS+1
264 001426 005600          SBC      RO
265 001430 005037 177700          CLR      @#177700
266 001434 005037 000006          CLR      @#6
267 001440 006300          ASL      RO
268 001442 016037 011426 011436  MOV      CPDLAY(RO),CPTIME ;GET TIME DELAY
269 001450 010037 001006          MOV      RO,PDPDLY    ;LOAD CPU DELAY
270 001454 001002          BNE      1S
271 001456 005237 001006          INC      PDPDLY      ;UPDATE IT
272                                     ; TRAP CATCHER
273 001462 012702 000242          1S:  MOV      #242,R2          ;LOAD R2
274 001466 012701 000240          MOV      #240,R1          ;LOAD R1
275 001472 010221          SS:  MOV      R2,(R1)+      ;LOAD .+2
276 001474 005021          CLR      (R1)+          ;LOAD HALT
277 001476 010102          MOV      R1,R2          ;LOAD R2
278 001500 005722          TST      (R2)+          ;BUMP R2
279 001502 020227 001002          CMP      R2,#1002      ;TEST FOR LAST
280 001506 001371          BNE      SS            ;BR UNTIL DONE
281 001510 005737 000042          TST      @#42
282 001514 001402          BEQ      .+6
283 001516 000137 001576          JMP      INIT2
284 001522 012777 000100 177270  MOV      #100,@TKS     ;ENABLE TTY INTERRUPTS
285 001530 005037 014772          CLR      PRINT1
286 001534 104000          PRINT          ;CALL MESSAGE PRINTER VIA 'EMT'
287 001536 013374          TITLE         ;TYPE PROGRAM HEADER.
288 001540 104000          INITA: PRINT          ;PRINT THE TEST CALL LETTERS.
289 001542 013515          MESS4         ;GO AND AWAIT COMMAND.
290 001544 000414          BR           ;C' OR A RESTART AT LOCATION '200'.
291                                     ; MONITOR SUBROUTINE. ENTER VIA 'C' OR A RESTART AT LOCATION '200'.
292 001546 004737 021526  MONITR: JSR      PC,$RESET ;INITIALIZE ON ENTRY
293 001552 013706 014774          MOV      STACK,SP      ;RESET STACK POINTER
294 001556 012777 000340 177232  MOV      #340,@PSW
295 001564 012777 000100 177226  MOV      #100,@TKS
296 001572 104000          PRINT          ;ENABLE TTY INTERRUPTS
297 001574 013762          CNTRLC        ;CALL MESSAGE PRINTER
298 001576 012737 011232 000024  MOV      #PWRFAL,@#24  ;TYPE 'C'
299 001604 012737 000006 000004  MOV      #6,@#4        ;SET UP POWER FAIL
                          ;SET UP BUSS ERROR

```

300	001612	005037	000006		CLR	2#6	
301	001616	013777	001146	177320	MOV	ADINT1,ADINT	
302	001624	005077	177316		CLR	ADINT1	
303	001630	012737	001540	014776	MOV	#INITA,AVECTR	;SET UP 'A' VECTOR ADDRESS.
304	001636	004537	013244		JSR	R5,LEDS	
305	001642	000006			B		
306	001644	005737	000042		TST	2#42	
307	001650	001402			BEQ	.+6	
308	001652	000137	002002		JMP	WHAT	
309	001656	012777	000100	177134	MOV	#100,ATKS	;ENABLE KEYBOARD INTERRUPT
310	001664	104000			PRINT		
311	001666	014022			DOT		;PRINT '.' TO INDICATE MONITOR READY
312	001670	104011			TTYIN		;WAIT FOR TTY ENTRY
313	001672	042737	000040	010546	BIC	#BITS,INBUF	;CLEAR LOWER CASE BIT
314	001700	122737	000101	010546	CMPB	#'A,INBUF	;TEST FOR 'A'
315	001706	001002			BNE	.+6	;NOT 'A'
316	001710	000137	002040		JMP	ADTEST	;YES RUN 'A TO D LOGIC' TEST
317	001714	122737	000103	010546	CMPB	#'C,INBUF	;TEST FOR 'C'
318	001722	001002			BNE	.+6	;NOT 'C'
319	001724	000137	005204		JMP	CALBRT	;YES RUN 'A TO D CALIBRATION' TEST
320	001730	122737	000104	010546	CMPB	#'D,INBUF	;TEST FOR 'D'
321	001736	001002			BNE	.+6	;NOT 'D'
322	001740	000137	005540		JMP	REPTST	;YES RUN 'A TO D REPEATABILITY' TEST
323	001744	122737	000105	010546	CMPB	#'E,INBUF	;TEST FOR 'E'
324	001752	001002			BNE	.+6	;NOT 'E'
325	001754	000137	006356		JMP	RECVRY	;YES RUN 'A TO D RECOVERY TEST
326	001760	122737	000106	010546	CMPB	#'F,INBUF	;TEST FOR 'F'
327	001766	001002			BNE	.+6	;NOT 'F'
328	001770	000137	006772		JMP	DSHTST	;YES RUN 'DUAL MODE' TEST
329	001774	104000			PRINT		;ILLEGAL ENTRY
330	001776	014025			QMARK		;TYPE '?'
331	002000	000676			BR	INIT2	;WAIT AGAIN
332	002002	005037	002036		CLR	NODMA	;CLEAR DMA POINTER
333	002006	000137	002044		JMP	ADTST1	;RUN A TO D TEST
334	002012				WHATA:		
335	002012	004737	021526		WHATB:	JSR	PC,\$RESET
336	002016	013700	000042		MOV	2#42,R0	
337	002022	004710			LOGICAL:	JSR	PC,(0)
338	002024	000240			NOP		
339	002026	000240			NOP		
340	002030	000240			NOP		
341	002032	000137	002002		JMP	WHAT	
342	002036	000000			NODMA:	0	

```

343 ;*****
344 ; A TO D LOGIC TEST
345 ;*****
346
347 002040 104000 ADTEST: PRINT
348 002042 014030 MESS
349 002044 004737 021526 ADTST1: JSR PC,$RESET
350 002050 005037 001034 CLR PASSCT
351 002054 012737 002162 013160 ADTSTD: MOV #ADT1+2,RETURN
352 002062 013777 001034 176742 MOV PASSCT,$DISPLA
353 002070 013706 014774 MOV STACK,$P
354 002074 005077 176716 CLR @PSW
355 002100 052777 000100 176712 BIS #BIT6,@TKS
356 002106 005037 001066 CLR .DVLS
357
358 ;TEST FOR NO BUSS ERRORS
359
360
361 ;* MOV $ZERO,@ADCS ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
362
363 ;* MOV $ZERO,@ADDBR ;/ PUT DATA FROM $ZERO TO DEVICE REG ADDBR
364 002132 012737 000002 015072 MOV #BIT1,$TMDAT
365
366 ;* MOV $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
367 002150 005737 020554 TST $AERR
368 002154 001401 BEQ .+4
369 002156 104400 ERROR
370
371 ;DOES (BIT 1) SET AND CLEAR
372
373 002160 104003 ADT1: SCOPE1
374 002162 012737 000002 015072 MOV #BIT1,$TMDAT ;SET BIT 1
375
376 ;* MOV $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
377
378 ;* MOV @ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
379 002210 022737 000002 015074 CMP #BIT1,$BDDAT
380 002216 001401 BEQ .+4
381 002220 104400 ERROR ;ERROR, ADCS NOT = 2
382
383 ;* MOV $ZERO,@ADCS ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
384
385 ;* MOV @ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
386 002242 005737 015074 TST $BDDAT
387 002246 001401 BEQ .+4
388 002250 104400 ERROR ;ERROR, ADCS NOT = 0
389
390 ;DOES (BIT 2) SET AND CLEAR
391
392 002252 104002 ADT2: SCOPE
393 002254 012737 000004 015072 MOV #BIT2,$TMDAT ;SET BIT 2
394
395 ;* MOV $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
396

```



```

397          ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
398 002302 022737 000004 015074      CMP      @BIT2,$BDDAT
399 002310 001401      BEQ      .+4
400 002312 104400      ERROR      ;ERROR, ADCS NOT = 4
401
402          ;*      MOV      $ZERO,@ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
403
404          ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
405 002334 005737 015074      TST      $BDDAT
406 002340 001401      BEQ      .+4
407 002342 104400      ERROR      ;ERROR, ADCS NOT = 0
408
409          ;DOES BURST (BIT 3) SET AND CLEAR
410
411 002344 104002      ADT3:  SCOPE
412 002346 012737 000010 015072      MOV      @BIT3,$TMDAT      ;SET BIT 3
413
414          ;*      MOV      $TMDAT,@ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
415
416          ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
417 002374 022737 000010 015074      CMP      @BIT3,$BDDAT
418 002402 001401      BEQ      .+4
419 002404 104400      ERROR      ;ERROR, ADCS NOT = 10
420
421          ;*      MOV      $ZERO,@ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
422
423          ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
424 002426 005737 015074      TST      $BDDAT
425 002432 001401      BEQ      .+4
426 002434 104400      ERROR      ;ERROR, ADCS NOT = 0

```

```

427
428
429
430 ; DOES SCHMITT TRIGGER (BIT 4) SET AND CLEAR
431 002436 104003 ADT4: SCOPE1
432 002440 012737 000020 015072 MOV #BIT4,$TMDAT ; SET BIT 4
433
434 ;* MOV $TMDAT,$ADCS ; / PUT DATA FROM $TMDAT TO DEVICE REG ADCS
435
436 ;* MOV $ADCS,$BDDAT ; / READ DEVICE REG ADCS, PUT DATA IN $BDDAT.
437 002466 022737 000020 015072 CMP #BIT4,$TMDAT
438 002474 001401 BEQ .+4
439 002476 104400 ERROR ; ERROR ADCS NOT = 20
440
441 ;* MOV $ZERO,$ADCS ; / PUT DATA FROM $ZERO TO DEVICE REG ADCS
442
443 ;* MOV $ADCS,$BDDAT ; / READ DEVICE REG ADCS, PUT DATA IN $BDDAT.
444 002520 005737 015074 TST $BDDAT
445 002524 001401 BEQ .+4
446 002526 104400 ERROR ; ERROR ADCS NOT = 0
447
448 ; DOES CLOCK OVERFLOW (BIT 5) SET AND CLEAR
449
450 002530 104002 ADT5: SCOPE
451 002532 012737 000040 015072 MOV #BIT5,$TMDAT ; SET BIT 5
452
453 ;* MOV $TMDAT,$ADCS ; / PUT DATA FROM $TMDAT TO DEVICE REG ADCS
454
455 ;* MOV $ADCS,$BDDAT ; / READ DEVICE REG ADCS, PUT DATA IN $BDDAT.
456 002560 022737 000040 015074 CMP #BIT5,$BDDAT
457 002566 001401 BEQ .+4
458 002570 104400 ERROR ; ERROR ADCS NOT = 40
459
460 ;* MOV $ZERO,$ADCS ; / PUT DATA FROM $ZERO TO DEVICE REG ADCS
461
462 ;* MOV $ADCS,$BDDAT ; / READ DEVICE REG ADCS, PUT DATA IN $BDDAT.
463 002612 005737 015074 TST $BDDAT
464 002616 105037 015072 CLRB $TMDAT
465
466 ;* MOV $TMDAT,$ADCS ; / PUT DATA FROM $TMDAT TO DEVICE REG ADCS
467
468 ;* MOV $ADCS,$BDDAT ; / READ DEVICE REG ADCS, PUT DATA IN $BDDAT.
469 002642 005737 015074 TST $BDDAT
470 002646 001401 BEQ .+4
471 002650 104400 ERROR ; ERROR ADCS NOT = 0
472
473 ; DOES INTERRUPT ENABLE (BIT 6) SET AND CLEAR
474
475
476 002652 104002 ADT6: SCOPE
477 002654 012737 000100 015072 MOV #BIT6,$TMDAT ; SET BIT 6
478
479 ;* MOV $TMDAT,$ADCS ; / PUT DATA FROM $TMDAT TO DEVICE REG ADCS
480

```

```

481                                     ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
482 002702 022737 000100 015074      CMP      #BIT6,$BDDAT
483 002710 001401                      BEQ      .+4
484 002712 104400                      ERROR                      ;ERROR ADCS NOT = 100
485
486                                     ;*      MOV      $ZERO,@ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
487
488                                     ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
489 002734 005737 015074      TST      $BDDAT
490 002740 001401                      BEQ      .+4
491 002742 104400                      ERROR                      ;ERROR ADCS NOT = 100
492
493                                     ;DOES MUX CHANNEL (BIT 8) SET AND CLEAR
494
495 002744 104002 000400 015072      ADT7:   SCOPE
496 002746 012737                      MOV      #BIT8,$TMDAT      ;SET BIT 8
497
498                                     ;*      MOV      $TMDAT,@ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
499
500                                     ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
501 002774 022737 000400 015074      CMP      #BIT8,$BDDAT
502 003002 001401                      BEQ      .+4
503 003004 104400                      ERROR                      ;ERROR ADCS NOT = 400
504
505                                     ;*      MOV      $ZERO,@ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
506
507                                     ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
508 003026 005737 015074      TST      $BDDAT
509 003032 001401                      BEQ      .+4
510 003034 104400                      ERROR                      ;ERROR ADCS NOT = 0

```

```

511
512           ;DOES MUX CHANNEL (BIT 9) SET AND CLEAR
513
514 003036 104002 ADT10: SCOPE
515 003040 012737 001000 015072 MOV      #BIT9,$TMDAT           ;SET BIT 9
516
517           ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
518           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
519           ;*      CMP      #BIT9,$BDDAT
520 003066 022737 001000 015074 BEQ      .+4
521 003074 001401
522 003076 104400 ERROR           ;ERROR ADCS NOT = 10000
523
524           ;*      MOV      $ZERO,$ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
525           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
526           ;*      TST      $BDDAT
527 003120 005737 015074 BEQ      .+4
528 003124 001401
529 003126 104400 ERROR           ;ERROR ADCS NOT = 0
530
531           ;DOES MUX CHANNEL (BIT 10) SET AND CLEAR
532
533 003130 104002 ADT11: SCOPE
534 003132 012737 002000 015072 MOV      #BIT10,$TMDAT        ;SET BIT 10
535
536           ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
537           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
538           ;*      CMP      #BIT10,$BDDAT
539 003160 022737 002000 015074 BEQ      .+4
540 003166 001401
541 003170 104400 ERROR           ;ERROR ADCS NOT 2000
542
543           ;*      MOV      $ZERO,$ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
544           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
545           ;*      TST      $BDDAT
546 003212 005737 015074 BEQ      .+4
547 003216 001401
548 003220 104400 ERROR           ;ERROR ADCS NOT = 0
549
550           ;DOES MUX CHANNEL (BIT 11) SET AND CLEAR
551
552 003222 104002 ADT12: SCOPE
553 003224 012737 004000 015072 MOV      #BIT11,$TMDAT        ;SET BIT 11
554
555           ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
556           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
557           ;*      CMP      #BIT11,$BDDAT
558 003252 022737 004000 015074 BEQ      .+4
559 003260 001401
560 003262 104400 ERROR           ;ERROR ADCST NOT = 4000
561
562           ;*      MOV      $ZERO,$ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
563           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
564

```

```

565 003304 005737 015074      TST      $BDDAT
566 003310 001401              BEQ      .+4
567 003312 104400              ERROR                      ;ERROR ADCST NOT = 0
568
569                               ;DOES MUX CHANNEL (BIT 12) SET AND CLEAR
570
571 003314 104002 010000 015072 ADT13:  SCOPE
572 003316 012737              MOV      #BIT12,$TMDAT
573
574                               ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
575                               ;*
576                               ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
577 003344 022737 010000 015074      CMP      #BIT12,$BDDAT
578 003352 001401              BEQ      .+4
579 003354 104400              ERROR                      ;ERROR ADCS NOT = 10000
580
581                               ;*      MOV      $ZERO,$ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
582                               ;*
583                               ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
584 003376 005737 015074      TST      $BDDAT
585 003402 001401              BEQ      .+4
586 003404 104400              ERROR                      ;ERROR ADCS NOT = 0
587

```

```

588                                     ;DOES MUX CHANNEL (BIT 13) SET AND CLEAR
589
590 003406 104002 ADT14: SCOPE
591 003410 012737 020000 015072 MOV      #BIT13,$TMDAT          ;SET BIT 13
592
593                                     ;*
594 MOV      $TMDAT,$ADCS          ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
595                                     ;*
596 003436 022737 020000 015074 MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
597 003444 001401 CMP      #BIT13,$BDDAT
598 003446 104400 BEQ      .+4
599                                     ;ERROR ADCS NOT = 20000
600                                     ;*
601 MOV      $ZERO,$ADCS          ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
602                                     ;*
603 003470 005737 015074 MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
604 003474 001401 TST      $BDDAT
605 003477 104400 BEQ      .+4
606                                     ;ERROR ADCS NOT = 0
607
608                                     ;DOES DUAL MODE (BIT14) SET AND CLEAR
609 003500 104002 ADT15: SCOPE
610 003502 012737 040000 015072 MOV      #BIT14,$TMDAT          ;SET BIT 14
611
612                                     ;*
613 MOV      $TMDAT,$ADCS          ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
614                                     ;*
615 003530 022737 040000 015074 MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
616 003536 001401 CMP      #BIT14,$BDDAT
617 003540 104400 BEQ      .+4
618                                     ;ERROR, ADCS NOT 40000
619                                     ;*
620 MOV      $ZERO,$ADCS          ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
621                                     ;*
622 003562 005737 015074 MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
623 003566 001401 TST      $BDDAT
624 003570 104400 BEQ      .+4
625                                     ;ERROR, ADCS NOT = 0
626
627                                     ;DOES ERROR (BIT 15) SET AND CLEAR
628 003572 104002 ADT16: SCOPE
629 003574 012737 100000 015072 MOV      #BIT15,$TMDAT
630
631                                     ;*
632 MOV      $TMDAT,$ADCS          ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
633                                     ;*
634 003622 022737 100000 015074 MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
635 003630 001401 CMP      #BIT15,$BDDAT
636 003632 104400 BEQ      .+4
637                                     ;ERROR , ERROR BIT FAILED TO SET
638                                     ;*
639 MOV      $ZERO,$ADCS          ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
640                                     ;*
641 003654 005737 015074 MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA .N $BDDAT.
641 TST      $BDDAT

```



```

642 003660 001401          BEQ      .+4
643 003662 104400          ERROR      ;ERROR, ERROR BIT FAILED TO CLEAR
644
645          ;DOES DONE (BIT 7) SET AND CLEAR
646
647 003664 104003          ADT20:   SCOPE1
648
649          ;*      MOV      @ADDBR,$BDDAT  ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
650          MOV      $BDDAT,R0
651 003676 013700 015074          CLR      KSTOR4
652 003702 005037 015014          MOV      #BIT0,$TMDAT
653 003706 012737 000001 015072
654          ;*      MOV      $TMDAT,@ADCS  ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
655
656          ;*      MOV      @ADCS,$BDDAT  ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
657 003734 105737 015074          ADT20A: TSTB   $BDDAT
658 003740 100404          BMI     ADT20B
659 003742 005237 015014          INC     KSTOR4
660 003746 001372          BNE     ADT20A
661 003750 104400          ERROR      ;ERROR, ADCS NOT = 200
662 003752          ADT20B:
663
664          ;*      MOV      @ADDBR,$BDDAT  ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
665 003762 013700 015074          MOV      $BDDAT,R0
666
667          ;*      MOV      @ADCS,$BDDAT  ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
668 003776 105737 015074          TSTB   $BDDAT
669 004002 100001          BPL     .+4
670 004004 104400          ERROR      ;ERROR, DONE FAILED TO CLEAR
671
672          ;*      MOV      @ADCS,$BDDAT  ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
673 004016 032737 000001 015074          BIT     #BIT0,$BDDAT
674 004024 001401          BEQ      .+4
675 004026 104400          ERROR      ;ERROR, AD TO FAILED TO cLEAR
    
```

```

676
677 ;DOES ERROR (BIT 15) SET
678
679 004030 104003 ADT21: SCOPE1
680
681 ;* MOV QADDBR,$BDDAT ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
682 004042 013700 015074 MOV $BDDAT,RO
683 004046 012737 000001 015072 MOV #BIT0,$TMDAT
684
685 ;* MOV $TMDAT,QADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
686 004064 105737 015074 1$: TSTB $BDDAT ;WAIT FOR FLAG
687 004070 100375 BPL 1$
688 004072 013737 011436 015032 MOV CPTIME,DELAY1
689 004100 006237 015032 ASR DELAY1
690 004104 006237 015032 ASR DELAY1
691 004110 006237 015032 ASR DELAY1
692 004114 012737 000001 015072 MOV #BIT0,$TMDAT
693
694 ;* MOV $TMDAT,QADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
695 004132 005337 015032 2$: DEC DELAY1
696 004136 100375 BPL 2$
697
698 ;* MOV QADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
699 004150 005737 015074 TST $BDDAT
700 004154 100401 BMI .+4 ;ERROR ADCS NOT = 100000
701 004156 104400 ERROR
702
703 ;* MOV $ZERO,QADCS ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
704 004170 013737 011436 015032 3$: MOV CPTIME,DELAY1
705 004176 005337 015032 DEC DELAY1
706 004202 100375 BPL 3$
707 004204 000240 NOP
708 004206 000240 NOP
709 004210 000240 NOP
710
711 ;* MOV QADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
712 004222 005737 015074 TST $BDDAT
713 004226 100001 BMI .+4 ;ERROR ADCS NOT = 0
714 004230 104400 ERROR
715
716 ;TEST THAT NO EXTERNAL CONVERSIONS INPL
717
718 004232 104003 ADT22: SCOPE1
719
720 ;* MOV $ZERO,QADCS ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
721
722 ;* MOV QADDBR,$BDDAT ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
723 004254 013700 015074 MOV $BDDAT,RO
724
725 ;* MOV QADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
726 004270 052737 000020 015072 BIS #BIT4,$TMDAT
727
728 ;* MOV $TMDAT,QADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
729 004306 005000 CLR RO

```

```

730 004310 005200      INC      RO
731 004312 001376      BNE      .-2
732
733                    ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
734 004324 105737 015074  TSTB     $BDDAT
735 004330 100001      BPL      .+4
736 004332 104400      ERROR      ;ERROR EXTERNAL CONVERSION
737
738                    ;TEST THAT NO CLOCK OVERFLOWS START CONVERSIONS
739
740 004334 104003      ADT23:   SCOPE1
741
742                    ;*      MOV      $ZERO,@ADCS      ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
743
744                    ;*      MOV      @ADDBR,$BDDAT      ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
745 004356 013700 015074  MOV      $BDDAT,RO
746
747                    ;*      MOV      @ADCS,$TMDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
748 004372 052737 000040 015072  BIS      #BITS,$TMDAT
749
750                    ;*      MOV      $TMDAT,@ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
751 004410 005037 015014      CLR      KSTOR4
752 004414 005237 015014      INC      KSTOR4
753 004420 001375      BNE      1$
754
755                    ;*      MOV      @ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
756 004432 105737 015074  TSTB     $BDDAT
757 004436 100001      BPL      .+4
758 004440 104400      ERROR      ;ERROR OVERFLOW STARTED A CONVERSION

```

;PRE INTERRUPT SETUP

759						
760						
761	004442	042737	177437	001004	BIC	#177437,ADBRL ;MASK TO BITS
762	004450	001001			BNE	.+4
763	004452	000000			HALT	;BR LEVEL INDICATED IS 0
764	004454	022737	000340	001004	CMP	#340,ADBRL ;IS IT BR LEVEL 7
765	004462	001001			BNE	.+4
766	004464	000000			HALT	;BR LEVEL INDICATED IS 7
767						
768	004466	013737	001004	015054	MOV	ADBRL, BRLEV1
769	004474	162737	000040	015054	SUB	#40, BRLEV1
770	004502	013737	001004	015056	MOV	ADBRL, BRLEV2
771	004510	013737	001004	015060	MOV	ADBRL, BRLEV3
772	004516	062737	000040	015060	ADD	#40, BRLEV3

E03

LPS-11 DIAGNOSTIC TEST I MAINDEC-11-DRLPH-A
DRLPH.P11

MACY11 27(654) 15-DEC-77 08:34 PAGE 19

SEQ 0030

```

773
774           ;TEST THAT RESET CLEARS MUX BITS
775
776 004524 104003 ADT27: SCOPE1
777 004526 012737 037400 015072 MOV      #BIT13!BIT12!BIT11!BIT10!BIT9!BIT8,$TMDAT
778
779           ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
780 004544 004737 021526 JSR      PC,$RESET
781
782           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
783 004560 005737 015074 TST      $BDDAT
784 004564 001401 BEQ      .+4
785 004566 104400 ERROR      ;ERROR, RESET FAILED TO CLEAR MUX BITS
786
787           ;TEST THAT RESET CLEARS ST, CLOCK OVERFLOW AND INTERRUPT ENABLE BITS
788
789 004570 104003 ADT30: SCOPE1
790 004572 012737 000160 015072 MOV      #BIT6!BIT5!BIT4,$TMDAT
791
792           ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
793 004610 004737 021526 JSR      PC,$RESET
794
795           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
796 004624 005737 015074 TST      $BDDAT
797 004630 001401 BEQ      .+4
798 004632 104400 ERROR      ;ERROR, RESET FAILED TO CLEAR ENABLES
799
800           ;TEST THAT RESET CLEARS DONE AND ERROR BITS
801
802 004634 104003 ADT31: SCOPE1
803 004636 012737 000001 015072 MOV      #BIT0,$TMDAT
804
805           ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
806
807           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
808 004664 105737 015074 TSTB     $BDDAT
809 004670 100375 BPL      .-4
810 004672 012737 100000 015072 MOV      #BIT15,$TMDAT ;SET ERROR BIT
811
812           ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
813 004710 004737 021526 JSR      PC,$RESET
814
815           ;*      MOV      $ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
816 004724 005737 015074 TST      $BDDAT
817 004730 001401 BEQ      .+4
818 004732 104400 ERROR      ;ERROR, RESET FAILED TO CLEAR ERROR OR DONE FLAG

```

```

819
820 ;TEST THAT RESET CLEARS BURST AND MODE BITS
821
822 004734 104003 ADT32: SCOPE1
823 004736 012737 000016 015072 MOV #BIT3!BIT2!BIT1,$TMDAT ;SET BITS 1-3
824
825 ;* MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
826 004754 004737 021526 JSR PC,$RESET
827
828 ;* MOV $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
829 004770 005737 015074 TST $BDDAT
830 004774 001401 BEQ .+4
831 004776 104400 ERROR ;ERROR, ADCS NOT = 0
832
833
834 ;LED TEST
835
836 TEST15: SCOPE1
837 005002 005737 000042 TST #42
838 005006 001060 BNE TEST16
839 005010 005737 001034 TST PASSCT ;TEST IF FIRST PASS
840 005014 001455 BEQ TEST16 ;BR IF YES
841 005016 005077 173774 CLR $PSW
842 005022 052777 000100 173770 BIS #BIT6,$TKS
843 005030 005037 015046 CLR TEMP1
844 005034 013737 015046 015050 TST15A: MOV TEMP1,TEMP2
845 005042 012737 000006 015052 MOV #6,TEMP3
846 005050 013737 015050 015072 TST15B: MOV TEMP2,$TMDAT
847
848 ;* MOV $TMDAT,$ADDBR ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADDBR
849 005066 013737 001006 015012 MOV PDPDLY,$KSTOR3
850 005074 012700 000000 MOV #0,RO
851 005100 032777 004000 173722 BIT #BIT11,$SWR ;TEST IF NO INTERATIONS
852 005106 001020 BNE TEST16 ;BR IF NO INTERATIONS
853 005110 005300 TST15C: DEC RO
854 005112 001376 BNE TST15C
855 005114 005337 015012 DEC $KSTOR3
856 005120 001373 BNE TST15C
857 005122 105237 015051 INCB TEMP2+1
858 005126 005337 015052 DEC TEMP3
859 005132 001346 BNE TST15B
860 005134 005237 015046 INC TEMP1
861 005140 022737 000040 015046 CMP #40,TEMP1
862 005146 001332 BNE TST15A
863
864 TEST16: SCOPE0 ;LOGICAL END OF A TO d LOGIC TEST
865 005152 004737 021526 JSR PC,$RESET
866 005156 005737 000042 TST #42
867 005162 001402 BEQ .+6
868 005164 000137 002012 JMP WHATA
869 005170 004737 013236 JSR PC,BELL ;REPORT END OF PASS
870 005174 005237 001034 INC PASSCT
871 005200 000137 002054 JMP ADTSTO

```

```

872 ;*****
873 ;CALIBRATION ROUTINE
874 ;*****
875 ;ROUTINE REQUESTS THE TYPE OF 'SYNC' TO BE USED ('I' INTERNAL OR 'E' EXTERNAL).
876 ;OR 'C' LPS CLOCK)
877 ;THE PROGRAM THEN TAKES CONTINUOUS CONVERSIONS USING DATA SW'S 5-0
878 ;TO SELECT THE CH.
879 ;OR SW '10' TO PRINT THE CONVERSION VALUE.
880
881
882 005204 012737 005216 014776 CALBRT: MOV #CALBT1,AVECTR ;SET UP 'A' RESTART ADDRESS
883 005212 104000 PRINT ;
884 005214 014057 MES7 ;TYPE TEST HEADER
885 005216 104000 CALBT1: PRINT ;
886 005220 014136 MES10 ;TEXT 'SYNC I OR E OR C'
887 005222 104011 CALB1A: TTYIN ;WAIT FOR INPUT.
888 005224 042737 000040 010546 BIC #BITS,INBUF ;
889 005232 013737 010546 015000 MOV INBUF,PROC ;SAVE IT IN TEMP STORAGE
890 005240 104000 PRINT ;
891 005242 014020 CRLF ;
892 005244 000402 BR ;
893 005246 005037 015000 MANCAL: CLR .+6 PROC ;CLEAR DMA FLAG
894 005252 005037 015036 CLR USEDMA ;CLEAR CLOCK FLAG
895 005256 005037 015040 CLR USECLK ;CLEAR DUAL MODE
896 005262 005037 015034 CLR USED5H ;
897 005266 012737 000001 015004 MOV #1,COUNT ;SET UP FOR '1' CONVERSION
898 005274 117737 173530 014771 CALBT2: MOVB #SWR,ADWRD2+1 ;GET CH. FROM THE SW REG.
899 005302 042737 140377 014770 BIC #140377,ADWRD2 ;CLR UNWANTED BITS, A/D WORD COMPLETE
900 005310 017737 173514 015010 MOV #SWR,KSTOR2 ;SAVE ORIGINAL SWITCH SETTING.
901 005316 022737 000105 015000 CMP #105,PROC ;TEST SYNC SELECT
902 005324 001004 BNE CALB2 ;BRANCH IF NOT 'E'
903 005326 052737 000020 014770 BIS #20,ADWRD2 ;OTHERWISE ADD 'EXT' SYNC BIT TO A/D.
904 005334 000416 BR CALB2A ;
905 005336 122737 000103 015000 CALB2: CMPB #103,PROC ;BRANCH IF NOT 'C'
906 005344 001007 BNE CALB2C ;OTHERWISE ADD 'CLOCK' SYNC TO A/D
907 005346 052737 000040 014770 BIS #40,ADWRD2 ;
908 005354 012737 177777 015040 MOV #-1,USECLK ;
909 005362 000403 BR CALB2A ;
910 005364 052737 000001 014770 CALB2C: BIS #1,ADWRD2 ;CHECK CLOCK FLAG
911 005372 005737 015040 CALB2A: TST USECLK ;NOT LPS CLOCK
912 005376 001422 BEQ CALB2D ;
913
914 ;* MOV $ZERO,QCSR ;/ PUT DATA FROM $ZERO TO DEVICE REG CSR
915 005410 012737 176000 015072 MOV #-2000,$TMDAT ;SET UP PRESET BUFFER
916
917 ;* MOV $TMDAT,QCSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
918 005426 012737 000403 015072 MOV #403,$TMDAT ;SET UP RATE, START CLOCK
919
920 ;* MOV $TMDAT,QCSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
921 005444 013706 014774 CALB2D: MOV STACK,SP ;TAKE AND STORE THE CONVERSIONS
922 005450 004737 010726 JSR PC,ADCNVT ;SETUP TO PRINT '1' VALUE
923 005454 012737 000001 015052 MOV #1,TEMP3 ;
924 005462 004537 013244 JSR R5,LEDS ;
925 005466 015214 ADBUFF ;

```

```

926 005470 032777 002000 173332 BIT #SW10,@SWR
927 005476 001404 BEQ CALB2B
928 005500 104012 PRTAVG ;PRINT IT
929 005502 104000 PRINT
930 005504 014020 CALF
931 005506 000406 BR CALBT4 ;TEST FOR LOOP
932 005510 013700 015214 CALB2B: MOV ADBUFF,RO ;SET UP A/D BUFFER.
933 005514 010077 173312 MOV RO,@DISPLA ;LOAD DISPLAY REG. (11/45)
934 005520 104013 TSTTKS ;TEST FOR TTY FLAG
935 005522 000240 ARESET: NOP
936 005524 104013 CALBT4: TSTTKS ;TEST FOR KEYBOARD INTERRUPT
937 005526 023777 015010 173274 CMP KSTOR2,@SWR ;TEST IF SWITCH REGISTER HAS CHANGED
938 005534 001716 BEQ CALB2A ;BRANCH AND TAKE NEXT CONVERSION
939 005536 000656 BR CALBT2 ;YES, COMPUTE NEW INPUT
940 *****
941 ;REPEATABILITY TEST
942 *****
943
944 ;THIS ROUTINE TO DESIGNED TO SHOW REPEATIBILITY BY TAKING A SERIES OF
945 ;'512' CONVERSIONS, AVERAGING THEM AND THEN CATORIZING
946 ;THEM IN BINS FROM THE AVERAGE PLUS & MINUS 6 COUNTS. THE ROUTINE
947 ;REQUESTS FOR A CHANNEL OR CHANNELS, A COUNT SPREAD AND A A/D MODE TO BE TYPED
948 ;IN VIA THE OPERATOR. A CONTINUOUS SERIES OF CONVERSIONS ARE THEN TAKEN
949 ;AND COMPARED AGAINST THE INPUT COUNT SPREAD. IF ALL '512' CONVERSIONS
950 ;ARE FOUND TO BE WITHIN THE SPREAD THE NEXT CH. IS EXERCISED OTHERWISE
951 ;THE COUNTS ARE TYPED OUT. SETTING SWITCH '10' TO A '1' WILL FORCE A PRINTOUT
952 ;OF THE CH (S).
953
954 005540 012737 005552 014776 REPTST: MOV #REPT1,AVECTR ;SET UP CNTR 'A' VECTOR ADDRESS
955 005546 104000 PRINT
956 005550 014256 MES13 ;TEXT 'REPEATIBILITY TEST'
957 005552 005037 015062 REPT1: CLR MESPRT
958 005556 005037 015036 CLR USEDMA
959 005562 005037 015034 CLR USED5H
960 005566 005037 015042 CLR OPS1
961 005572 104000 PRINT
962 005574 014306 MES14 ;REQUEST CHANNEL (S)
963 005576 104011 TTYIN ;WAIT FOR INPUT
964 005600 104001 DECOCT ;CONVERT TO OCTAL
965 005602 013737 010716 015006 MOV BCDTAB,KSTOR1 ;SAVE AS INTIAL CH.
966 005610 013737 015006 015010 MOV KSTOR1,KSTOR2 ;ALSO SAVE AS 2ND CH. ENTRY
967 005616 005737 010720 TST BCDTAB+2 ;TEST FOR SECOND ENTRY
968 005622 001407 BEQ REPT2 ;BRANCH IF NO SECOND ENTRY
969 005624 023737 010720 015006 CMP BCDTAB+2,KSTOR1 ;COMPARE ENTRY 1 TO ENTRY 2
970 005632 100747 BMI REPT1 ;BRANCH AND RESTART IF ILLEGAL
971 005634 013737 010720 015010 MOV BCDTAB+2,KSTOR2 ;OTHERWISE SAVE AS SECOND CH.
972 005642 104000 REPT2: PRINT
973 005644 014346 MES16 ;TEXT 'COUNT SPREAD ??'
974 005646 104011 TTYIN ;WAIT FOR ENTRY
975 005650 104001 DECOCT ;DECODE TO OCTAL
976 005652 013737 010716 015012 MOV BCDTAB,KSTOR3 ;SAVE IT
977 005660 104000 PRINT
978 005662 014365 MES18
979 005664 104011 TTYIN ;REQUEST MODE

```


980	005666	122737	000102	010546		CMPB	#'B, INBUF	; TEST FOR B
981	005674	001003				BNE	REPT2B	; NOT
982	005676	005137	015036			COM	USEDMA	
983	005702	000411				BR	REPT2A	
984	005704	122737	000104	010546	REPT2B:	CMPB	#'D, INBUF	; TEST FOR DUAL MODE
985	005712	001005				BNE	REPT2A	; BRANCH IF NOT
986	005714	005137	015034			COM	USEDMA	; SET DUAL MODE
987	005720	052737	000100	015006		BIS	#BIT6, KSTOR1	; BIT
988	005726	013706	014774		REPT2A:	MOV	STACK, SP	
989	005732	013737	015006	015014		MOV	KSTOR1, KSTOR4	; SAVE STARTING CH.
990	005740	104013			REPT3:	TSTTKS		; TEST FOR KEYBOARD FLAG
991	005742	012737	001000	015004		MOV	#1000, COUNT	; SET FOR '512' CONVERSIONS
992	005750	113737	015014	014771		MOVB	KSTOR4, ADWORD2+1	; MOV SELECTED CH. TO HIGH BYTE OF ADWORD
993	005756	042737	100377	014770		BIC	#100377, ADWORD2	; MASK
994	005764	052737	000001	014770		BIS	#1, ADWORD2	
995	005772	004737	010726			JSR	PC, ADCNVT	; TAKE THE CONVERSIONS
996	005776	104004				CMPUTE		; AVERAGE & COMPUTE DISTRIBUTION
997	006000	104005				CATORIZ		
998	006002	013777	015114	173022		MOV	AVERAGE, DISPLA	; LOAD DISPLAY REG. (11/45)
999	006010	032777	002000	173012		BIT	#SW10, JSWR	; TEST DATA SW10
1000	006016	001047				BNE	REPT4	; IF SET, FORCE TYPE OUT
1001	006020	032777	020000	173002	TSTCT4:	BIT	#SW13, JSWR	; TEST FOR INHIBIT TYPEOUT
1002	006026	001135				BNE	REPT7	; BRANCH IF SW SET
1003	006030	022737	000004	015012		CMP	#4, KSTOR3	; WAS 4 TYPED
1004	006036	001005				BNE	TSTCT3	; NO, TEST FOR '3'
1005	006040	022737	001000	015170		CMP	#1000, XSPRD4	; TOTAL COUNTS WITHIN 4 COUNTS
1006	006046	001033				BNE	REPT4	; BRANCH IF NO.
1007	006050	000524				BR	REPT7	; YES, TEST NEXT CH.
1008	006052	022737	000003	015012	TSTCT3:	CMP	#3, KSTOR3	; COUNT = TO 3
1009	006060	001005				BNE	TSTCT2	; NO TEST COUNT 2
1010	006062	022737	001000	015166		CMP	#1000, XSPRD3	
1011	006070	001022				BNE	REPT4	; BRANCH IF COUNT NOT WITHIN 3
1012	006072	000513				BR	REPT7	; YES, TEST NEXT CH.
1013								
1014	006074	022737	000002	015012	TSTCT2:	CMP	#2, KSTOR3	; COUNT = TO 2
1015	006102	001005				BNE	TSTCT1	; NO, TEST COUNT 1
1016	006104	022737	001000	015164		CMP	#1000, XSPRD2	
1017	006112	001011				BNE	REPT4	; BRANCH IF NOT WITHIN 2
1018	006114	000502				BR	REPT7	; YES, TEST NEXT CH.
1019	006116	022737	000001	015012	TSTCT1:	CMP	#1, KSTOR3	; COUNT = TO 1
1020	006124	001004				BNE	REPT4	; NO, REPORT EVEN IF NOT '0'
1021	006126	022737	001000	015162		CMP	#1000, XSPRD1	
1022	006134	001472				BEQ	REPT7	; BRANCH IF TOTAL WITHIN 1 COUNT
1023	006136	104000			REPT4:	PRINT		
1024	006140	014020				CRLF		
1025	006142	005737	015062			TST	MESPRT	; TEST IF HEADER HAS BEEN TYPED
1026	006146	001002				BNE	REPTS	; BRANCH IF YES
1027	006150	104000				PRINT		
1028	006152	014444				MES19		; TEXT 'CH. HIGH AVG. LOW'
1029	006154	104013			REPTS:	TSTTKS		; TEST FOR KEYBOARD INTERRUPT
1030	006156	104000				PRINT		
1031	006160	014020				CRLF		; CARRIAGE RETURN, LINE FEED
1032	006162	013737	015014	006354		MOV	KSTOR4, REPT8A	; MOV. CH.
1033	006170	042737	177700	006354		BIC	#177700, REPT8A	

J03

LPS-11 DIAGNOSTIC TEST I MAINDEC-11-DRLPH-A
dRLPH.P11

MACY11 27(654) 15-DEC-77 08:34 PAGE 24

SEQ 0035

1034	006176	005737	015034		TST	USED5H		; TEST FOR DUAL MODE
1035	006202	001403			BEQ	REPT8		; BRANCH IF NOT
1036	006204	062737	000010	006354	ADD	#10, REPT8A		
1037	006212	104010			REPT8:			
1038	006214	006354			PRTCT			
1039	006216	104007			REPT8A			
1040	006220	104010			SPACE			
1041	006222	015070			PRTCT			; PRINT LOW VALUE
1042	006224	104007			LOW			
1043	006226	104010			SPACE			
1044	006230	015114			PRTCT			; PRINT AVERAGE VALUE
1045	006232	104007			AVRAGE			
1046	006234	104010			SPACE			
1047	006236	015066			PRTCT			; PRINT HIGH VALUE
1048	006240	005737	015062		HIGH			
1049	006244	001002			TST	MESPRT		
1050	006246	104000			BNE	REPT6		
1051	006250	014477			PRINT			
1052	006252	052737	000007	015062	MES20			; PRINT 'COUNT SPREAD' HEADER
1053	006260	022737	001000	015144	REPT6:	#7, MESPRT		; INHIBIT OTHER HEADERS
1054	006266	000240			BIS	#1000, AVGCNT		; TEST IF ALL COUNTS WERE AT AVG.
1055	006270	104000			CMP			; <BEQ REPT7> BRANCH TO NEXT CH. IF YES.
1056	006272	014020			NOP			
1057	006274	012704	015130		PRINT			
1058	006300	012402			CRLF			
1059	006302	104006			MOV	#ORLOW, R4		
1060	006304	022704	015162		REPT6A:	(R4)+, R2		; TYPE OUT COUNT SPREAD
1061	006310	001373			MOV			; TEST FOR DONE
1062	006312	005777	172512		BINDEC			; BRANCH IF NO AND TYPE NEXT COUNT
1063	006316	100001			CMP	#XSPRD1, R4		
1064	006320	000000			BNE	REPT6A		
1065	006322	013704	015014		TST	QSWR		
1066	006326	042704	177700		BPL	REPT7		; REPEATIBILITY ERROR
1067	006332	023704	015010		HALT			
1068	006336	001404			REPT7:	KSTOR4, R4		
1069	006340	005237	015014		MOV	#177700, R4		
1070	006344	000137	005740		BIC	KSTOR2, R4		; TESTED ALL CH.(S)?
1071	006350	000137	005726		CMP	REPT7A		
1072					BEQ	REPT7A		
1073	006354	000000			REPT7A:	KSTOR4		; TEST NEXT CHANNEL
					JMP	REPT3		
					REPT7A:	REPT2A		
					REPT8A:	0		

K03

```

1074
1075
1076
1077
1078
1079
1080
1081 006356 012737 006400 014776
1082 006364 005037 015036
1083 006370 005037 015034
1084 006374 104000
1085 006376 014106
1086 006400 104000
1087 006402 014306
1088 006404 104011
1089 006406 104001
1090 006410 013737 010716 015006
1091 006416 013737 010720 015010
1092 006424 013706 014774
1093 006430 104013
1094 006432 012737 000020 006744
1095 006440 005037 006746
1096 006444 012737 000010 015052
1097 006452 012737 000010 015004
1098 006460 113737 015006 014771
1099 006466 042737 140377 014770
1100 006474 052737 000001 014770
1101 006502 005237 015030
1102 006506 004737 010726
1103 006512 113737 015010 014771
1104 006520 042737 140377 014770
1105 006526 052737 000001 014770
1106 006534 005237 015030
1107 006540 004737 010726

:*****
:RECOVERY TEST
:*****
:THIS TEST IS DESIGNED TO TEST RECOVERY OF THE A/D CONVERTER VIA ACCEPT-
:ING TWO (2) CHANNEL FROM THE TELETYPE AND THEN TAKE A
:SERIES OF EIGHT CONVERSIONS ON EACH CHANNEL AND TYPING OUT THE CON-
:VERSION VALUES OF THE 2ND CHANNEL IN THE ORDER THEY WERE TAKEN.
RECVRY: MOV #RECVY1,AVECTR ;SET UP THE '1A' RETURN ADDRESS
        CLR USEDMA
        CLR USED5H
        PRINT
        MES8 ;TEXT 'RECOVERY TEST'
RECVY1: PRINT ;REQUEST CHANNELS
        MES14
        TTYIN ;WAIT FOR INPUT
        DECOCT ;CONVERT TO OCTAL
        MOV BCDTAB,KSTOR1 ;SAVE 1ST CH.
        MOV BCDTAB+2,KSTOR2 ;SAVE 2ND CH.
RECVY2: MOV STACK,SP
        TSTTKS ;CHECK FOR KEYBOARD FLAG
        MOV #16.,10$ ;LOAD COUNT
        CLR 11$ ;CLEAR POINTER
        MOV #10,TEMP3 ;SET UP TO PRINT 10
        MOV #10,COUNT ;SETUP TO TAKE '8' CONVERSIONS ON 1ST CH.
1$: MOVB KSTOR1,ADWRD2+1 ;LOAD 1ST CH.
   BIC #140377,ADWRD2
   BIS #1,ADWRD2
   INC DELAY ;SET FLAG
   JSR PC,ADCNVT ;TAKE THE CONVERSIONS
   MOVB KSTOR2,ADWRD2+1 ;SET UP 2ND CH.
   BIC #140377,ADWRD2
   BIS #1,ADWRD2
   INC DELAY ;SET FLAG
   JSR PC,ADCNVT ;TAKE 2ND SERIES OF CONVERSIONS

```

1108	006544	013701	006746			MOV	11\$,R1		;LOAD A POINTER
1109	006550	012700	015214			MOV	#ADBUFF,R0		;LOAD POINTER
1110	006554	012061	015256			MOV	(R0)+,ADTB0(R1)		
1111	006560	012061	015320			MOV	(R0)+,ADTB1(R1)		
1112	006564	012061	015362			MOV	(R0)+,ADTB2(R1)		
1113	006570	012061	015424			MOV	(R0)+,ADTB3(R1)		
1114	006574	012061	015466			MOV	(R0)+,ADTB4(R1)		
1115	006600	012061	015530			MOV	(R0)+,ADTB5(R1)		
1116	006604	012061	015572			MOV	(R0)+,ADTB6(R1)		
1117	006610	012061	015634			MOV	(R0)+,ADTB7(R1)		
1118	006614	062737	000002	006746		ADD	#2,11\$;UPDATE POINTER
1119	006622	005337	006744			DEC	10\$		
1120	006626	001314				BNE	1\$		
1121									
1122	006630	012700	000010			MOV	#8,R0		;LOAD COUNT
1123	006634	012702	015214			MOV	#ADBUFF,R2		;LOAD POINTER
1124	006640	012701	000000			MOV	#0,R1		
1125	006644	012737	000017	015046	2\$:	MOV	#1\$,TEMP1		;LOAD TEMP
1126	006652	012737	000004	012160		MOV	#4,CMPCNT		;LOAD AVRG. COUNT
1127	006660	016104	006752			MOV	LADTB(R1),R4		;GET POINTER
1128	006664	012746	000340			MOV	#340,-(SP)		;PUSH
1129	006670	004737	012014			JSR	PC,CMPTEA		;CONVERT AND AVERAGE
1130	006674	013722	015114			MOV	AVRAGE,(R2)+		;SAVE AVERAGE
1131	006700	005721				TST	(R1)+		;UPDATE POINTER
1132	006702	005300				DEC	R0		
1133	006704	001357				BNE	2\$;BR IF NOT DONE
1134									
1135	006706	032777	002000	172114		BIT	#SW10,@SWR		;TEST BIT 10
1136	006714	001243				BNE	RECVY2		
1137	006716	104000				PRINT			
1138	006720	014131				MES9			;TEXT 'CH.'
1139	006722	104013				TSTTKS			;CHECK AGAIN FOR KEYBOARD FLAG
1140	006724	013737	015010	006354		MOV	KSTOR2,REPTBA		
1141	006732	104010				PRTCT			
1142	006734	006354				REPTBA			
1143	006736	104007				SPACE			
1144	006740	104012				PRTAVG			;PRINT VALUES OF 2ND CH.
1145	006742	000630				BR RECVY2			;DO IT AGAIN
1146	006744	000000							
1147	006746	000000							
1148	006750	000000							
1149									
1150	006752	015256				LADTB:	ADTB0		
1151	006754	015320					ADTB1		
1152	006756	015362					ADTB2		
1153	006760	015424					ADTB3		
1154	006762	015466					ADTB4		
1155	006764	015530					ADTB5		
1156	006766	015572					ADTB6		
1157	006770	015634					ADTB7		

```

1158
1159
1160
1161
1162
1163 006772 012737 007000 014776 DSHTST: MOV #DSHT0,AVECTR ;SET UP '1A' RETURN
1164 007000 104000 DSHTO: PRINT ;TYPE HEADER
1165 007002 014321 MES15
1166 007004 005037 001034 DSHT1: CLR PASSCT ;CLEAR PASS COUNT
1167 007010 005037 015024 DSHT2: CLR STCHAN ;CLEAR STARTING CHANNEL
1168 007014 004537 013244 DSHT3: JSR RS,LEDS
1169 007020 015024 STCHAN
1170 007022 000240 NOP ;RESET''
1171 007024 052777 000100 171766 BIS #BIT6,ATKS ;ENABLE INTERRUPT
1172 007032 013706 014774 MOV STACK,SP ;LOAD THE STACK POINTER
1173 007036 013777 001034 171766 MOV PASSCT,ADISPLA ;LOAD PASS COUNT
1174 007044 113737 015024 015072 MOVB STCHAN,$TMDAT ;LOAD CHANNEL <0-3>
1175 007052 005077 171740 CLR $PSW
1176
1177
1178 007066 052737 000001 015072 ;* MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1179 BIS #1,$TMDAT
1180
1181 007104 ;* MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1182
1183 ;* MOV $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1184 007114 105737 015074 TSTB $BDDAT
1185 007120 100371 BPL 1$
1186
1187 ;* MOV $ADDBR,AVERM4 ;/READ DEVICE REG ADDBR,PUT DATA IN AVERM4.
1188
1189 ;* MOV $ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
1190 007142 152737 000010 015073 BISB #10,$TMDAT+1
1191
1192 ;* MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1193
1194 ;* MOV $ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
1195 007170 052737 000001 015072 BIS #1,$TMDAT
1196
1197 ;* MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1198 007206 ;*
1199
1200 ;* MOV $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1201 007216 105737 015074 TSTB $BDDAT
1202 007222 100371 BPL 2$
1203
1204 ;* MOV $ADDBR,AVERM3 ;/READ DEVICE REG ADDBR,PUT DATA IN AVERM3.
1205
1206 007234 004537 007634 JSR RS,COMPR ;COMPARE TWO NUMBERS
1207 007240 015104 AVERM4
1208 007242 015106 AVERM3
1209 007244 001014 BNE DSHT6 ;BRANCH IF NOT EQUAL <DIFFERENT INPUT>
1210 007246 032777 020000 171554 BIT #BIT13,$SWR
1211 007254 001004 BNE DSHT4

```

N03

LPS-11 DIAGNOSTIC TEST I MAINDEC-11-DRLPH-A
DRLPH.P11

MACY11 27(654) 15-DEC-77 08:34 PAGE 28

SEQ 0039

1212 007256 104000
1213 007260 014020
1214 007262 104000
1215 007264 014605
1216 007266 005777
1217 007272 100001
1218 007274 000000
1219
1220

171536

DSHT4:

PRINT
CRLF
PRINT
MES22
TST
BPL
HALT

JSWR
DSHT6

;PRINT MESSAGE

;ERROR, CHANNEL PAIRS HAVE SAME INPUT VOLTAGE
;THE FIRST CHANNEL NUMBER OF THE PAIR IS IN THE LEDS

```

1221
1222 ;NOW TEST THAT DUAL SAMPLE REALLY WORKS
1223 ;SET DUAL MODE AND CONVERT A CHANNEL THE FIRST CONVERSION
1224 ;SHOULD NOT EQUAL THE SECOND CONVERSION
1225 ;IF IT DOES THE DUAL SAMPLE AND HOLD IS NOT WORKING CORRECTLY
1226
1227 007276 113737 015024 015073 DSHT6:  MOVB  STCHAN,$TMDAT+1 ;LOAD 1ST MUX CHANNEL
1228
1229 ;*  MOV  $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1230 007314 052737 040000 015072 ;*  BIS  #BIT14,$TMDAT
1231
1232 ;*  MOV  $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1233
1234 ;*  MOV  $ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
1235 007342 052737 000001 015072 ;*  BIS  #1,$TMDAT
1236
1237 ;*  MOV  $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1238 007360 ;S:
1239
1240 ;*  MOV  $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1241 007370 105737 015074 ;*  TSTB $BDDAT ;WAIT FOR DONE
1242 007374 100371 ;*  BPL  IS
1243
1244 ;*  MOV  $ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
1245 007406 005237 015072 ;*  INC  $TMDAT
1246
1247 ;*  MOV  $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1248
1249 ;*  MOV  $ADDBR,$AVERM2 ;/READ DEVICE REG ADDBR,PUT DATA IN AVERM2.
1250 ;*  ;SAVE 1ST CHANNEL CONVERSION
1251 007432 ;S:
1252
1253 ;*  MOV  $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1254 007442 105737 015074 ;*  TSTB $BDDAT ;WAIT FOR DONE
1255 007446 100371 ;*  BPL  2S
1256
1257 ;*  MOV  $ADDBR,$AVERM1 ;/READ DEVICE REG ADDBR,PUT DATA IN AVERM1.
1258
1259 007460 004537 007634 ;*  JSR  R5,COMPR
1260 007464 015110
1261 007466 015112
1262 007470 001015
1263 007472 032777 020000 171330 ;*  BNE  DSHT7 ;BRANCH IF NOT EQUAL
1264 007500 001004 ;*  BIT  #BIT13,$SWR
1265 007502 104000 ;*  BNE  DSHT6A
1266 007504 014020 ;*  PRINT
1267 007506 104000 ;*  CRLF
1268 007510 014663 ;*  PRINT
1269 007512 005777 171312 ;*  MES23 ;PRINT ERROR
1270 007516 100014 ;*  DSHT6A: TST  $SWR
1271 007520 000000 ;*  BPL  DSHT7A
1272 007522 000412 ;*  HALT ;ERROR, FIRST CONVERSION ON CHANNEL X (STCHAN) WAS EQUAL
1273 ;*  BR  DSHT7A ;TO THE SECOND CONVERSION IN DUAL MODE

```

1274											
1275	007524	004537	007634		DSHT7:	JSR	R5, COMP				
1276	007530	015104				AVERM4				; COMPARE TWO NUMBERS	
1277	007532	015110				AVERM2				; TEST IF NOT EQUAL	
1278	007534	001021				BNE	DSHT8				
1279	007536	004537	007634			JSR	R5, COMP			; COMPARE TWO SAMPLES	
1280	007542	015106				AVERM3					
1281	007544	015112				AVERM1					
1282	007546	001014				BNE	DSHT8			; BRANCH IF NOT EQUAL	
1283	007550	005237	015024		DSHT7A:	INC	STCHAN			; UPDATE CHANNEL	
1284	007554	022737	000004	015024		MP	#4, STCHAN			; FINISHED ?	
1285	007562	001402				BEQ	IS			; YES	
1286	007564	000137	007014			JMP	DSHT3				
1287	007570	005037	015024		IS:	CLR	STCHAN			; CLEAR CHANNEL	
1288	007574	104013				TSTTKS				; TEST KEYBOARD	
1289	007576	000414				BR	DSHT8B				
1290											
1291	007600	032777	020000	171222	DSHT8:	BIT	#BIT13, @SWR				
1292	007606	001004				BNE	DSHT8A				
1293	007610	104000				PRINT					
1294	007612	014020				CRLF					
1295	007614	104000				PRINT					
1296	007616	014742				MES24					
1297	007620	005777	171204		DSHT8A:	TST	@SWR				
1298	007624	100001				BPL	DSHT8B				
1299	007626	000000				HALT				; ERROR, PREVIOUS CONVERSION ON CHANNEL NOT EQUAL	
1300	007630	000137	007550		DSHT8B:	JMP	DSHT7A			; CURRENT CONVERSION	
1301											
1302	007634	013537	007724		COMP:	MOV	@(R5)+, CMPRA			; SAVE 1ST NUMBER	
1303	007640	013537	007726			MOV	@(R5)+, CMPRB			; SAVE 2ND NUMBER	
1304	007644	013737	007726	007730		MOV	CMPRA, CMPRC			; SAVE 2ND AGAIN	
1305	007652	163737	007724	007730		SUB	CMPRA, CMPRC			; SUBTRACT 2ND FROM 1ST	
1306	007660	001413				BEQ	CMPR2			; BRANCH IF EQUAL	
1307	007662	100005				BPL	CMPR1			; BRANCH IF + (A-B=+)	
1308	007664	063737	001010	007730		ADD	AMASK, CMPRC			; ADD SPREAD	
1309	007672	100006				BPL	CMPR2			; BRANCH IF (A=B+MASK)	
1310	007674	000410				BR	CMPR3			; BRANCH IF	
1311	007676	163737	001010	007730	CMR1:	SUB	AMASK, CMPRC			; SUBTRACT AMASK	
1312	007704	003401				BLE	CMPR2			; BRANCH IF EQUAL *	
1313	007706	000403				BR	CMPR3			; BRANCH IF NOT-EQUAL *	
1314											
1315	007710	005037	007730		CMR2:	CLR	CMPRC			; A AND B ARE WITHIN AMASK	
1316	007714	000205				RTS	R5				
1317											
1318	007716	05137	007730		CMR3:	COM	CMPRC			; A AND B ARE NOT WITHIN AMASK	
1319	007722	000205				RTS	R5				
1320											
1321	007724	000000			CMR4:	0					
1322	007726	000000			CMR5:	0					
1323	007730	000000			CMR6:	0					


```

1324 ;ROUTINE TO LOOP THUR A SINGLE LOGIC SUBTEST. ENTERED FROM THE 'MONITOR'
1325 ;VIA TYPING 'TNN' WHERE 'NN' IS EQUATED TO THE 'PC' OF A SUBTEST.
1326 ;NOTE THAT 'SW11' MUST BE '0' (DOWN) TO RUN THIS TEST.
1327
1328 007732 012701 010546 TESTX: MOV #INBUF,R1
1329 007738 005037 015006 CLR KSTOR1
1330 007742 042711 177770 TSTA: BIC #177770,(R1) ;MASK TO OCTAL
1331 007746 062137 015006 ADD (R1)+,KSTOR1 ;ADD TO LAST RESULT
1332 007752 005337 015002 DEC CHCNT
1333 007756 001407 BEQ TSTXB
1334 007760 006337 015006 ASL KSTOR1
1335 007764 006337 015006 ASL KSTOR1
1336 007770 006337 015006 ASL KSTOR1
1337 007774 000762 BR TSTA
1338
1339 007776 022737 005204 015006 TSTXB: CMP #CALBRT,KSTOR1 ;IS NO. WITHIN LIMITS OF THE LOGIC TEST
1340 010004 100002 BPL .+6 ;CONTINUE IF YES
1341 010006 000137 001774 JMP INIT3 ;OTHERWISE RETURN TO MONITOR
1342 010012 062737 000002 015006 ADD #2,KSTOR1 ;ADD '2' TO POINT TO INSTRUCTION AFTER SCOPE
1343 010020 005037 013156 XLOOP: CLR SCOPEF ;KEEP COUNT AT ZERO
1344 010024 012737 010020 013160 MOV #XLOOP RETURN ;LOAD SCOPE LOOP RETURN POINTER
1345 010032 000177 004750 JMP @KSTOR1 ;JUMP TO TEST
1346
1347 ;SUBROUTINE TO ISSUE N SPACES
1348 ;N IS ONE PLUS VALUE CONTAINED IN SPACEX
1349 ;SPACEX IS CLEARED WITHIN THE SUBROUTINE, SO THAT A CALL ON
1350 ;SPACE WITHOUT LOADING SPACEX ISSUES ONLY ONE SPACE
1351
1352 010036 105777 170762 XSPACE: TSTB @TPS ;WAIT FOR TTY READY
1353 010042 100375 BPL -4
1354 010044 012777 000240 170754 MOV #240,@TPB ;OUTPUT A SPACE
1355 010052 005337 010066 DEC SPACEX ;DECREMENT COUNT
1356 010056 003367 BGT XSPACE ;LOOP IF NOT DONE
1357 010060 005037 010066 CLR SPACEX ;RESET COUNT TO ZERO
1358 010064 000002 RTI ;RETURN
1359 010066 000000 SPACEX: 0
1360
1361 ;SUBROUTINE TO TYPE OUT AVERAGES FOR THE RECOVERY TEST
1362
1363 010070 012737 015214 010100 XPRTAV: MOV #ADBUFF,AVGTAB
1364 010076 104010 XPTA1: PRTCT
1365 010100 015214 AVGTAB: ADBUFF
1366 010102 062737 000002 010100 ADD #2,AVGTAB
1367 010110 012737 000002 010066 MOV #2,SPACEX
1368 010116 104007 SPACE
1369 010120 005337 015052 DEC TEMP3
1370 010124 001364 BNE XPTA1
1371 010126 000002 RTI

```

```

1372                                     ;KEYBOARD SERVICE ROUTINE
1373
1374 010130 010046 XTTYIN: MOV      RD, -(SP)
1375 010132 010146      MOV      R1, -(SP)
1376 010134 010246      MOV      R2, -(SP)
1377 010136 010346      MOV      R3, -(SP)
1378 010140 010446      MOV      R4, -(SP)
1379 010142 010546      MOV      R5, -(SP)
1380 010144 012704 010546 NEWIN: MOV      #INBUF, R4      ; SETUP CHARACTER BUFFER
1381 010150 042777 000100 170642 BIC      #BIT6, @TKS
1382 010156 005037 015002      CLR      CHRCNT      ; CLEAR CHARACTER COUNTER
1383 010162 005037 010546      CLR      INBUF
1384 010166 005037 010550      CLR      INBUF+2
1385 010172 005037 010552      CLR      INBUF+4
1386 010176 005037 010554      CLR      INBUF+6
1387 010202 005037 010556      CLR      INBUF+10
1388 010206 005037 010560      CLR      INBUF+12
1389 010212 005037 010562      CLR      INBUF+14
1390 010216 105777 170576 INPUTA: TSTB   @TKS      ; CHARACTER READY?
1391 010222 100375      BPL      INPUTA      ; NO, WAIT IT OUT
1392 010224 017701 170572      MOV      @TKB, R1    ; SAVE CHARACTER
1393 010230 042701 177600      BIC      #177600, R1 ; STRIP PARITY BIT
1394 010234 120127 000060      CMPB    R1, #60      ; IS IT A SPECIAL CHARACTER
1395 010240 100420      BMI      SPCHR      ; YES, TEST IT
1396 010242 122701 000173      CMPB    #173, R1
1397 010246 100415      BMI      SPCHR
1398 010250 010124      MOV      R1, (R4)+   ; SAVE CHARACTER
1399 010252 005237 015002 INPUTB: INC      CHRCNT ; INCREMENT THE CHARACTER COUNT.
1400 010256 022737 000006 015002      CMP      #6, CHRCNT
1401 010264 100524      BMI      SPCHR5
1402 010266 105777 170532 OUTPTA: TSTB   @TPS      ; ECHO CHARACTER
1403 010272 100375      BPL      OUTPTA
1404 010274 110177 170526      MOVB   R1, @TPB
1405 010300 000746      BR      INPUTA      ; WAIT FOR NEXT CHARACTER
1406
1407                                     ; SUBROUTINE TO TEST FOR SPECIAL CHARACTERS : '↑A', '↑C', '↑G', 'CR', ',', ' OR 'RUBOUT'
1408
1409 010302 122701 000001 SPCHR:  CMPB    #1, R1      ; CHAR. = '↑A'
1410 010306 001016      BNE     SPCHR1      ; NO, NOT '↑A'
1411 010310 104000      PRINT   ; ECHO '↑A'
1412 010312 013766      CNTRLA ; RESTORE 'SP'
1413 010314 012605      MOV     (SP)+, R5
1414 010316 012604      MOV     (SP)+, R4
1415 010320 012603      MOV     (SP)+, R3
1416 010322 012602      MOV     (SP)+, R2
1417 010324 012601      MOV     (SP)+, R1
1418 010326 012600      MOV     (SP)+, R0
1419 010330 022626      CMP     (SP)+, (SP)+
1420 010332 052777 000100 170460 BIS      #BIT6, @TKS ; ENABLE KEYBOARD INTR.
1421 010340 000177 004432      JMP     @VECTA      ; YES, EXIT VIA '↑A' VECTOR ADDRESS.
1422 010344 122701 000003 SPCHR1: CMPB    #3, R1      ; CHAR. = '↑C'
1423 010350 001002      BNE     1$         ; NO, NOT '↑C'
1424 010352 000137 001546      JMP     MONITR      ; YES, EXIT TO MONITOR
1425 010356 122701 000177 1$:      CMPB    #177, R1    ; CHAR. = 'RUBOUT'

```

1426	010362	001011		BNE	SPCHR3		; NO TRY ANOTHER CHAR
1427	010364	005737	015002	TST	CHRCNT		; IS RUBOUT LEGAL?
1428	010370	001712		BEQ	INPUTA		; NO, IGNORE IT
1429	010372	005337	015002	DEC	CHRCNT		
1430	010376	012701	000134	MOV	#134,R1		; TYPE '\ ' TO INDICATE RUBOUT
1431	010402	005744		TST	-(R4)		; POP OFF LAST CHARACTER
1432	010404	000730		BR	OUTFTA		; WAIT FOR NEXT CHARACTER
1433	010406	122701	000054	SPCHR3: CMPB	#54,R1		; TEST FOR ' '
1434	010412	001716		BEQ	INPUTB		; LEGAL CHAR., SAVE IT
1435	010414	122701	000015	SPCHR4: CMPB	#15,R1		; =TO 'CARRIAGE RETURN' TO TERMINATE?
1436	010420	001014		BNE	1\$; NO, CONTINUE
1437	010422	104000		PRINT			; YES, TYPE 'CR-LF'
1438	010424	014020		CRLF			
1439	010426	012605		4\$: MOV	(SP)+,R5		
1440	010430	012604		MOV	(SP)+,R4		
1441	010432	012603		MOV	(SP)+,R3		
1442	010434	012602		MOV	(SP)+,R2		
1443	010436	012601		MOV	(SP)+,R1		
1444	010440	012600		MOV	(SP)+,R0		
1445	010442	052777	000100 170350	BIS	#BIT6,ATKS		; ENABLE KEYBOARD INTR.
1446	010450	000002		RTI			; EXIT
1447	010452	122701	000007	1\$: CMPB	#7,R1		; TEST IF CTRL G
1448	010456	001027		BNE	SPCHR5		; BR IF NOT
1449	010460	104000		PRINT			
1450	010462	013771		CNTRLG			
1451	010464	104010		PRTOCT			; PRINT OLD VALUE
1452	010466	000170		SOFTSW			
1453	010470	104000		PRINT			
1454	010472	014007		NEWSWR			; ASK FOR NEW VALUE
1455	010474	104011		TTYIN			
1456	010476	005001		CLR	R1		
1457	010500	012700	010546	MOV	#INBUF,R0		
1458	010504	005710		2\$: TST	(R0)		; TEST FOR TERM
1459	010506	001410		BEQ	3\$; BR IF TERM
1460	010510	012002		MOV	(R0)+,R2		; GET CHAR
1461	010512	042702	177770	BIC	#177770,R2		; MASK OUT
1462	010516	006301		ASL	R1		
1463	010520	006301		ASL	R1		
1464	010522	006301		ASL	R1		
1465	010524	060201		ADD	R2,R1		
1466	010526	000766		BR	2\$		
1467	010530	010137	000170	3\$: MOV	R1,SOFTSW		; SAVE NEW SWITCH VALUE
1468	010534	000734		BR	4\$		

```

1469 010536 104000          SPCHRS: PRINT          ; OTHERWISE TYPE '?'
1470 010540 014025          QMARK
1471 010542 000137 010130  JMP          XTTYIN      ; WAIT FOR NEW ENTRY
1472 010546 000000          INBUF: 0              ; CHARACTER STORAGE BUFFER
1473
1474
1475          010564          ; SUBROUTINE WILL CONVERT 'N' WORDS (SEPARATED VIA COMMA'S)
1476          ; WHICH WERE STORED IN A TABLE VIA 'TTYIN' TO OCTAL AND STORE THEM.
1477
1478
1479 010564 012704 010546  BCDBIN: MOV          #INBUF,R4      ; SETUP ASCII STORAGE TABLE
1480 010570 012703 010716  MOV          #BCDTAB,R3      ; TABLE FOR STORAGE OF CONVERTED WORDS
1481 010574 005037 010720  CLR          BCDTAB+2
1482 010600 005001          BCDNB1: CLR          R1      ; REG. TO STORE RUNNING TOTAL
1483 010602 005002          CLR          R2      ; TEMP. STORAGE FOR 'R1'
1484 010604 005737 015002  BCDNB2: TST          CHRCNT    ; END OF DATA?
1485 010610 003424          BLE          BCDEND      ; YES, EXIT
1486 010612 005337 015002  DEC          CHRCNT      ; DECREMENT CHARACTER COUNTER
1487 010616 122714 000054  CMPB        #54,(R4)     ; IS CHARACTER = TO '?'
1488 010622 001417          BEQ          BCDEND      ; YES, DECODE NEW WORD
1489 010624 121427 000060  CMPB        (R4),#60
1490 010630 002425          BLT          BCDERR      ; TEST FOR LEGAL NO.
1491 010632 021427 000067  CMP          (R4),#67
1492 010636 003022          BGT          BCDERR
1493 010640 042714 177770  BIC          #177770,(R4) ; STRIP NO.
1494 010644 012400          MOV          (R4)+,R0     ; SAVE NO. IN R0.
1495 010646 010102          MOV          R1,R2      ; SAVE CURRENT TOTAL
1496 010650 006301          ASL          R1
1497 010652 006301          ASL          R1
1498 010654 006301          ASL          R1
1499 010656 060001          ADD          R0,R1      ; N+NEW NO.
1500 010660 000751          BR          BCDNB2
1501 010662 020127 000077  BCDEND: CMP          R1,#77
1502 010666 003006          BGT          BCDERR
1503 010670 005724          TST          (R4)+
1504 010672 010123          MOV          R1,(R3)+
1505 010674 005737 015002  TST          CHRCNT
1506 010700 001337          BNE          BCDNB1
1507 010702 000002          RTI
1508 010704 104000          BCDERR: PRINT
1509 010706 014025          QMARK
1510 010710 104011          TTYIN
1511 010712 000137 010564  JMP          BCDBIN
1512 010716 000000          BCDTAB: 0
1513 010720 000000          0
1514 010722 000000          0
1515 010724 000000          0

```

```

1516 ;SUBROUTINE TO TAKE 'N' CONVERSIONS AND STORE THEM IN AN A/D BUFFER. ROUTINE
1517 ;IS ENTERED WITH 'N' IN COUNT AND THE CH TO BE CONVERTED IN 'ADWORD'.
1518
1519 010726 005077 170064 ADCNVT: CLR @PSW
1520 010732 013737 015004 015046 MOV COUNT,TEMP1 ;SET UP # OF CONVERSIONS TO BE TAKEN
1521 010740 012704 015214 MOV #ADBUF,R4 ;LOAD BUFFER ADDRESS
1522 010744 005037 015026 CLR FIRST ;CLEAR TEMP
1523 010750 005737 015030 TST DELAY ;TEST DELAY
1524 010754 001042 BNE 6$ ;IF SET, SKIP
1525
1526
1527 ;* MOV @ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
1528 010766 113737 015073 001152 MOVB $TMDAT+1,DMK2
1529 010774 123737 014771 001152 CMPB ADWORD2+1,DMK2 ;TEST
1530 011002 001415 BEQ 1$ ;BR IF SAME
1531 011004 113737 014771 015073 MOVB ADWORD2+1,$TMDAT+1 ;LOAD MUX CHANNELS
1532
1533 ;* MOV $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1534 011032 013737 011436 015032 MOV CPTIME,DELAY1
1535 011030 005337 015032 2$: DEC DELAY1
1536 011034 001375 BNE 2$ ;DELAY
1537 011036 005037 015030 1$: CLR DELAY
1538 011042 113737 014770 015072 4$: MOVB ADWORD2,$TMDAT ;LOAD CONTROL BYTE
1539
1540 ;* MOV $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1541 011060 000407 BR 5$
1542 011062 013737 014770 015072 6$: MOV ADWORD2,$TMDAT ;LOAD CONTROL WORD
1543
1544 ;* MOV $TMDAT,@ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1545 011100 5$: ;LOAD CH. & START CONVERT IF NOT 'EXT'
1546
1547 ;* MOV @ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1548 011110 105737 015074 TSTB $BDDAT
1549 011114 100371 BPL 5$ ;WAIT FOR DONE
1550 011116 005737 015074 TST $BDDAT
1551 011122 100016 BPL 11$ ;BRANCH IT NOT SET
1552
1553 ;* MOV $ZERO,@ADCS ;/ PUT DATA FROM $ZERO TO DEVICE REG ADCS
1554 011134 037727 167670 020000 BIT @SWR,#BIT13
1555 011142 001002 BNE 10$
1556 011144 104000 PRINT
1557 011146 014236 MES12
1558 011150 005777 167654 10$: TST @SWR
1559 011154 100001 BPL 11$
1560 011156 000000 HALT ;ERROR BIT SET
1561
1562 011160 005737 015034 11$: TST USED5H
1563 011164 001410 BEQ 12$ ;BRANCH IF NOT DUAL MODE
1564 011166 005137 015026 COM FIRST
1565 011172 001405 BEQ 12$
1566
1567 ;* MOV @ADDBR,BRLEV1 ;/READ DEVICE REG ADDBR,PUT DATA IN BRLEV1.
1568 011204 000716 BR 4$
1569 011206 12$:

```

1570						
1571				:*	MOV	2ADDBR,STMDAT ;/READ DEVICE REG ADDBR,PUT DATA IN STMDAT.
1572	011216	013724	015072		MOV	STMDAT,(4)+
1573	011222	005337	015046		DEC	TEMP1 ;DECREMENT COUNTER
1574	011226	003305			BGT	45 ;IF NOT '0' TAKE NEXT CONVERSION
1575	011230	000207			RTS	PC
1576						

```

1577
1578 ;POWER FAIL HANDLER
1579
1580 011232 010046 PWRFAL: MOV R0,-(SP)
1581 011234 010146 MOV R1,-(SP)
1582 011236 010246 MOV R2,-(SP)
1583 011240 010346 MOV R3,-(SP)
1584 011242 010446 MOV R4,-(SP)
1585 011244 010546 MOV R5,-(SP)
1586 011246 013746 000024 MOV 24,-(SP)
1587 011252 010637 015000 MOV SP,PROC
1588 011256 012737 011266 000024 MOV #PWRUP,24
1589 011264 000000 HALT
1590
1591 ;POWER UP HANDLER
1592
1593 011266 012777 000340 167522 PWRUP: MOV #340,@PSW
1594 011274 013706 015000 MOV PROC,SP
1595 011300 012637 000024 MOV (SP)+,24
1596 011304 012605 MOV (SP)+,R5
1597 011306 012604 MOV (SP)+,R4
1598 011310 012603 MOV (SP)+,R3
1599 011312 012602 MOV (SP)+,R2
1600 011314 012601 MOV (SP)+,R1
1601 011316 012600 MOV (SP)+,R0
1602 011320 004737 021526 JSR PC,$RESET
1603 011324 104000 PRINT
1604 011326 014565 MES21
1605 011330 000137 001546 JMP MONITR

```

```

1606
1607
1608 ;EMT DISPATCH SERVICE ROUTINE
1609 ;ARGUMENT OF EMT IS EXTRACTED AND USED AS OFFSET TO OBTAIN POINTER
1610 ;TO THE SELECTED SUBROUTINE.
1611
1612 011334 011646 EMTSRV: MOV (SP), -(SP) ;GET PC FOR TO RETURN
1613 011336 162716 000002 SUB #2, (SP) ;PC OF EMT
1614 011342 017616 000000 MOV @ (SP), (SP) ;GET EMT
1615 011346 005716 TST (SP) ;IS EMT VALID?
1616 011350 001001 BNE EMTOK
1617 011352 000000 HALT ;INVALID EMT
1618 011354 006316 EMTOK: ASL (SP) ;MULTIPLY EMT ARG BY '2'
1619 011356 042716 177001 BIC #177001, (SP) ;CLEAR UNWANTED BITS
1620 011362 062716 011374 ADD #EMTTAB, (SP) ;POINTER TO SUBROUTINE ADDRESS
1621 011366 017616 000000 MOV @ (SP), (SP) ;SUBROUTINE ADDRESS
1622 011372 000136 JMP @ (SP)+ ;GO TO SUBROUTINE
1623
1624 ;EMT DISPATCH TABLE
1625
1626 011374 011440 EMTTAB: TYPMES ;MESSAGE PRINT ROUTINE
1627 011376 010564 BCDBIN ;BINARY CONVERSION ROUTINE
1628 011400 013066 SCOPEC ;LOGIC TEST SCOPE ROUTINE
1629 011402 013162 SCOPEH ;LOGIC TEST SCOPE LOOP (10)
1630 011404 011774 CMPTE ;SUBROUTINE TO COMPUTE THE AVG
1631 011406 012162 CATORZ ;SUBROUTINE TO COMPUTE 'COUNT SPREAD'
1632 011410 011576 DEC,RT ;SUBROUTINE TO CONVERT OCT TO DEC + PRINT
1633 011412 010036 XSPACE ;SUBROUTINE TO TYPE SPACES
1634 011414 012522 OCTPRT ;OCTAL PRINT ROUTINE
1635 011416 010130 XTTYIN ;TELEPRINTER SERVICE ROUTINE
1636 011420 010070 XPRTAV ;SUBROUTINE TO PRINT OUT THE GAIN AVERAGES
1637 011422 012652 TKSFLG ;SUBROUTINE TO TEST FOR KEYBOARD FLAG
1638 011424 013222 SCOPEI ;SCOPE ROUTINE
1639
1640 011426 002000 CPDLAY: 2000 ;PDP-11/05
1641 011430 002400 2400 ;PDP-11/20
1642 011432 003500 3500 ;PDP-11/40
1643 011434 006000 6000 ;PDP-11/45
1644
1645 011436 002000 CPTIME: 2000

```



```

1646
1647
1648
1649
1650 011440 005077 167352
1651 011444 010537 012650
1652 011450 017605 000000
1653 011454 062716 000002
1654 011460 105777 167340
1655 011464 100375
1656 011466 122715 000100
1657 011472 061003
1658 011474 013705 012650
1659 011500 000002
1660 011502 122715 000045
1661 011506 001403
1662 011510 112577 167312
1663 011514 000761
1664 011516 012777 000015 167302
1665 011524 105777 167274
1666 011530 100375
1667 011532 012777 000012 167266
1668 011540 013737 001012 011574
1669 011546 105777 167252
1670 011552 100375
1671 011554 113777 001014 167244
1672 011562 005337 011574
1673 011566 100367
1674 011570 105725
1675 011572 000732
1676 011574 000002
1677
1678
1679
1680 011576 005077 167214
1681 011602 012737 177774 011756
1682 011610 012737 011764 011762
1683 011616 012737 000240 011760
1684 011624 012737 177777 011754
1685 011632 005237 011754
1686 011636 167702 000120
1687 011642 100373
1688 011644 067702 000112
1689 011650 004737 011674
1690 011654 005237 011756
1691 011660 001001
1692 011662 000002
1693 011664 062737 000002 011762
1694 011672 000754
1695 011674 005737 011754
1696 011700 001010
1697 011702 022737 177777 011756
1698 011710 001404
1699 011712 013737 011760 011754

```

```

;MESSAGE PRINT ROUTINE, ENTERED VIA EMT DISPATCH HANDLER.
;ROUTINE PICKS UP CONTENTS OF THE 'PC' AND USES THIS AS
;THE ADDRESS OF MESSAGE TO BE TYPED.

```

```

TYPMES: CLR @PSW ;ENABLE KEYBOARD INTR.
MOV R5, SAV5
MOV @($F) R5 ;GET THE MESSAGE ADDRESS FROM START
ADD #2, ($P) ;SET UP STACK TO EXIT
TYPERR: TSTB @TPS
BPL TYPERA ;WAIT FOR TTY DONE
CMPB #100, (R5) ;TEST FOR 'a'
BNE TYPER1 ;BRANCH IF NO EQUAL
MOV SAV5, R5
RTI ;OTHERWISE EXIT
TYPER1: CMPB #45, (R5) ;TEST FOR '%'
BEQ TYPECL ;IF = TYPE 'CR-LF'
TYPER2: MOVB (R5)+, @TPB ;OUTPUT CHAR.
BR TYPERA
TYPECL: MOV #15, @TPB ;TYPE 'CR'
TSTB @TPS
BPL :-4
MOV #12, @TPB
MOV FILLS, 2$ ;LOAD FILL COUNT
1$: TSTB @TPS
BPL 1$
MOVB FILCHR, @TPB ;FILL CHAR
DEC 2$
BPL 1$
TSTB (R5)+ ;INCREMENT BUFFER
BR TYPERA
2$: 2

```

```

;PRINT DECIMAL VALUE IN R2

```

```

DECPRT: CLR @PSW
MOV #-4, DIGCNT
MOV #DECPNT+2, DECPNT
MOV #240, ZERO
TYPT1: MOV #-1, DIGIT
TYPT2: INC DIGIT
SUB @DECPNT, %2
BPL TYPT2
ADD @DECPNT, %2
JSR PC, DECOU
INC DIGCNT
BNE TYPT3
RTI
TYPT3: ADD #2, DECPNT
BR TYPT1
DECOU: TST DIGIT
BNE DECI
CMP #-1, DIGCNT
BEQ DECI
MOV ZERO, DIGIT

```

```

1700 011720 000406
1701 011722 012737 000260 011760 DEC1: MOV DEC2
1702 011730 052737 000260 011754 BIS #260,ZERO
1703 011736 105777 167062 DEC2: TSTB #260,DIGIT
1704 011742 100375 BPL .-4
1705 011744 013777 011754 167054 MOV DIGIT,ATPB
1706 011752 000207 RTS 7
1707
1708 011754 000000 DIGIT: 0
1709 011756 000000 DIGCNT: 0
1710 011760 000240 ZERO: 240
1711 011762 011764 DECPNT: .+2
1712 011764 001750 1000.
1713 011766 000144 100.
1714 011770 000012 10.
1715 011772 000001 1.
1716
1717 ; COMPUTE THE RESULTS OF 512 CONVERSIONS AS HIGH, LOW AND AVERAGE
1718
1719 011774 012737 000777 015046 CMPTE: MOV #777,TEMP1 ; SET UP TO COMPARE '511' NUMBERS
1720 012002 012704 015214 MOV #A0BUFF,R4 ; SET UP DATA BUFFER ADDRESS
1721 012006 012737 000011 012160 MOV #11,CMPCNT ; LOAD AVRG. COUNT
1722 012014 005037 015064 CMPTEA: CLR HIORDV
1723 012020 012437 015114 MOV (R4)+,AVRAGE ; STORE 1ST VALUE AS AVERAGE
1724 012024 013737 015114 015066 MOV AVRAGE,HIGH ; HIGH
1725 012032 013737 015114 015070 MOV AVRAGE,LOW ; & LOW
1726 012040 012437 015050 GETDAT: MOV (R4)+,TEMP2
1727 012044 023737 015050 015066 CMP TEMP2,HIGH ; IS NEW NO. GREATER THAN OLD NO.
1728 012052 003403 BLE TSLO ; BRANCH IF NOT GREATER
1729 012054 013737 015050 015066 MOV TEMP2,HIGH ; OTHERWISE SAVE AS NEW HIGH
1730 012062 023737 015050 015070 TSLO: CMP TEMP2,LOW
1731 012070 003003 BGT TAGA
1732 012072 013737 015050 015070 MOV TEMP2,LOW ; OTHERWISE SAVE AS NEW LOW
1733 012100 063737 015050 015114 TAGA: ADD TEMP2,AVRAGE ; ADD LOW ORDER
1734 012106 005537 015064 ADC HIORDV ; ADD CARRY TO HI ORDER
1735 012112 005337 015046 DEC TEMP1
1736 012116 001350 BNE GETDAT ; 512 OR 16 ADDITIONS?
1737 012120 013737 012160 015046 MOV CMPCNT,TEMP1 ; YES, DIVIDE/512 OR 16
1738 012126 006237 015064 AVGDAT: ASR HIORDV ; SHIFT CARRY BIT INTO LO ORDER
1739 012132 006037 015114 ROR AVRAGE
1740 012136 005337 015046 DEC TEMP1 ; DONE?
1741 012142 001371 BNE AVGDAT ; YES, ADD REMAINDER TO LO ORDER
1742 012144 005537 015114 ADC AVRAGE
1743 012150 004537 013244 JSR S,LEDS
1744 012154 015114 AVRAGE
1745 012156 000002 RTI
1746
1747 012160 000011 CMPCNT: 11 ; 11 OR 4

```

```

1748 ;SUBROUTINE TO CALCULATE THE PLUS & MINUS 5 COUNT LIMITS FROM AN AVERAGE
1749
1750 012162 012737 000005 015046 CATORZ: MOV #5,TEMP1
1751 012170 013737 015114 015050 MOV AVERAGE,TEMP2 ;MOV AVER. TO WORK AREA
1752 012176 012703 015116 MOV #AVERP1,R3 ;SETUP DISTRIBUTION TABLE (POS.)
1753 012202 005237 015050 FILE1: INC TEMP2 ;A=A+1
1754 012206 013723 015050 MOV TEMP2,(R3)+ ;SAVE A+1
1755 012212 005337 015046 DEC TEMP1 ;SAVED '5' COUNTS?
1756 012216 001371 BNE FILE1 ;BRANCH IF NO
1757 ;SET UP TABLE OF AVG. -1 TO -5
1758 012220 012737 000005 015046 MOV #5,TEMP1
1759 012226 013737 015114 015050 MOV AVERAGE,TEMP2 ;MOV AVG. TO WORK AREA.
1760 012234 012703 015114 MOV #AVERAGE,R3 ;SET UP DISTRIBUTION TABLE NEG.
1761 012240 005337 015050 FILE2: DEC TEMP2 ;A=1-1
1762 012244 013743 015050 MOV TEMP2,-(R3) ;SAVE 'A-1'
1763 012250 005337 015046 DEC TEMP1 ;SAVED '5' COUNTS?
1764 012254 001371 BNE FILE2 ;BRANCH IF NO
1765
1766 ;CATEGORIZE THE COUNT SPREAD AS '+6 & -6' COUNTS FROM THE AVERAGE
1767
1768 012256 012703 015130 CATORZ: MOV #ORLOW,R3 ;CLEAR COUNTS
1769 012262 005023 CATR1: CLR (R3)+
1770 012264 022703 015162 CMP #ORHIGH+2,R3 ;FINISHED?
1771 012270 001374 BNE CATR1 ;NO, CLEAR NEXT COUNTER
1772 012272 012737 001001 015046 MOV #1001,TEMP1 ;COMPARE '512' COUNTS
1773 012300 012700 015214 CATR2: MOV #ADBUFF,R0 ;SET UP A/D BUFFER
1774 012304 005337 015046 DEC TEMP1
1775 012310 001437 BEQ CATR5 ;EXIT IF '0'
1776 012312 012037 015050 MOV (R0)+,TEMP2
1777 012316 023737 015126 015050 CMP AVERP5,TEMP2
1778 012324 100423 BMI OVRHI
1779 012326 023737 015050 015102 CMP TEMP2,AVERM5
1780 012334 100422 BMI OVRLO
1781 012336 005001 CLR R1
1782 012340 012702 015102 CATR3: MOV #AVERM5,R2
1783 012344 022237 015050 CMP (R2)+,TEMP2
1784 012350 001405 BEQ CATR4
1785 012352 005201 INC R1
1786 012354 022701 000013 CMP #13,R1
1787 012360 001371 BNE CATR3
1788 012362 000000 HALT ;FATAL ERROR
1789 012364 006301 CATR4: ASL R1 ;MULTIPLY 'OFFSET' X2
1790 012366 005261 015132 INC MINUSS(R1)
1791 012372 000744 BR CATR2
1792 012374 005237 015160 OVRHI: INC ORHIGH
1793 012400 000741 BR CATR2
1794 012402 005237 015130 OVRLO: INC ORLOW
1795 012406 000736 BR CATR2

```

```

1796
1797
1798
1799 012410 013737 015144 015162
1800 012416 063737 015146 015162
1801 012424 063737 015142 015162
1802 012432 013737 015162 015164
1803 012440 063737 015150 015164
1804 012446 063737 015140 015164
1805 012454 013737 015164 015166
1806 012462 063737 015152 015166
1807 012470 063737 015136 015166
1808 012476 013737 015166 015170
1809 012504 063737 015154 015170
1810 012512 063737 015134 015170
1811 012520 000002
1812
1813
1814
1815
1816 012522 010537 012650
1817 012526 017605 000000
1 18 012532 062716 000002
1819 012536 012737 000006 012642
1820 012544 012737 000376 012646
1821 012552 000401
1822 012554 006115
1823 012556 006115
1824 012560 006115
1825 012562 111537 012644
1826 012566 143737 012646 012644
1827 012574 052737 000260 012644
1828 012602 132777 000200 166214
1829 012610 100374
1830 012612 113777 012644 166206
1831 012620 012737 000370 012646
1832 012626 005337 012642
1833 012632 001350
1834 012634 013705 012650
1835 012640 000002
1836 012642 000000
1837 012644 000000
1838 012646 000376
1839 012650 000000
1840
1841
1842
1843 012652 105777 166142
1844 012656 100001
1845 012660 104011
1846
1847
1848 012662 000002
1849

;ADD THE COUNTS AND SAVE TOTAL IN SPREADS OF '1-4'
CATRS: MOV AVGCNT,XSPRD1
        ADD PLUS1,XSPRD1
        ADD MINUS1,XSPRD1 ;=TO NO. COUNTS AT SPREAD OF '1'
        MOV XSPRD1,XSPRD2
        ADD PLUS2,XSPRD2
        ADD MINUS2,XSPRD2 ;=TO NO. COUNTS AT SPREAD OF '2'
        MOV XSPRD2,XSPRD3
        ADD PLUS3,XSPRD3
        ADD MINUS3,XSPRD3 ;=TO NO. COUNTS AT SPREAD OF '3'
        MOV XSPRD3,XSPRD4
        ADD PLUS4,XSPRD4
        ADD MINUS4,XSPRD4 ;=TO NO. COUNTS AT SPREAD OF '4'
        RTI ;EXIT

;SUBROUTINE TO TYPEOUT A '6' DIGIT OCTAL NO. THE 'PC' CONTAINS
;THE ADDRESS OF 'WORD' TO BE TYPED
OCTPRT: MOV R5,SAVS
        MOV @($P),R5 ;THE ADDRESS OF WORD TO BE TYPED
        ADD #2,($P) ;SET UP STACK TO EXIT
        MOV #6,10$ ;LOAD COUNTER
        MOV #376,MASK ;MASK FOR FIRST BIT
        BR .+4
1$: ROL (R5)
    ROL (R5)
    ROL (R5)
    MOVB (R5),11$
    BICB MASK,11$
    BIS #260,11$
    BITB #200,@TPS
    BPL ;-6 ;WAIT FOR PRINTER READY
    MOVB 11$,@TPB ;PRINT CHAR.
    MOV #370,MASK ;MASK FOR NEXT '5' DIGITS
    DEC 10$
    BNE 1$
    MOV SAVS,R5
    RTI
10$: 0
11$: 0
MASK: 376
SAVS: 0

;SUBROUTINE TO TEST FOR THE KEYBOARD FLAG BEING SET
TKSFLG: TSTB @TKS ;FLAG SET?
        BPL .+4 ;NO, EXIT
        TTYIN ;YES, INQUIRE

RTI
;ENTERED WITH SYSTEM TRAP CALL (ERROR)

```

```

1850 012664 104013 LOGERR: TSTTKS ;TEST FOR KEYBOARD INTERRUPT
1851 012666 037727 166136 020000 BIT @SWR, @20000 ;TEST FOR INHIBIT PRINT OUT
1852 012674 001067 BNE CK ;INHIBIT, CHECK FOR HALT
1853 012676 012737 000002 000006 MOV @RTI, @#6 ;SET UP FOR BUSS ERROR
1854 012704 011637 015012 MOV (SP), KSTOR3 ;PC OF FAILING ROUTINE
1855 012710 162737 000002 015012 SUB #2, KSTOR3
1856 012716 004537 013244 JSR 5, LEDS
1857 012722 015012 KSTOR3
1858
1859 ;* MOV @ADCS, KSTOR4 ;/READ DEVICE REG ADCS, PUT DATA IN KSTOR4.
1860
1861 ;* MOV @ADDBR, KSTOR5 ;/READ DEVICE REG ADDBR, PUT DATA IN KSTOR5.
1862 012744 005037 015022 CLR KSTR12 ;SAVE ZERO IN KSTR12
1863 012750 013737 015044 015020 MOV TEMP, KSTR11 ;SAVE TEMP
1864 012756 012737 000000 000006 MOV #0, @#6 ;RESET BUSS ERROR
1865 012764 005737 014772 TST PRINT1
1866 012770 001006 BNE LGERR1
1867 012772 104000 PRINT
1868 012774 014020 CRLF
1869 012776 104000 PRINT
1870 013000 013453 MES1
1871 013002 005237 014772 INC PRINT1
1872 013006 104000 LGERR1: PRINT ;OUTPUT CARRIAGE RETURN AND LINE FEED
1873 013010 014020 CRLF
1874 013012 104010 PRTOCT ;PRINT FAILING PC+2
1875 013014 015012 KSTOR3
1876 013016 104007 SPACE ;OUTPUT A SPACE
1877 013020 104010 PRTOCT ;PRINT
1878 013022 015014 KSTOR4 ;CONTENTS OF A/D STATUS REG.
1879 013024 104007 SPACE
1880 013026 104010 PRTOCT
1881 013030 015016 KSTOR5
1882 013032 104007 SPACE
1883 013034 104010 PRTOCT
1884 013036 015022 KSTR12
1885 013040 104007 SPACE
1886 013042 104010 PRTOCT
1887 013044 015020 KSTR11
1888 013046 105777 165752 TSTB @TPS
1889 013052 100375 BPL .-4
1890 013054 005777 165750 CK: TST @SWR ;CHECK SR FOR HALT SWITCH
1891 013060 100001 BPL .+4 ;BRANCH IF NOT SET
1892 013062 000000 HALT ;HALT ON ERROR UP
1893 013064 000002 RTI ;RETURN TO MAIN LINE

```

```

1894
1895 ;SCOPE AND/OR ITERATION LOOP FOR SOME TEST 4000 TIMES
1896
1897 013066 104013 SCOPEC: TSTTKS ;TEST FOR KEYBOARD INIT
1898 013070 032777 040000 165732 BIT #40000,@SWR ;TEST SR FOR SCOPE
1899 013076 001015 BNE SCOPEB ;YES, SCOPE
1900 013100 032777 004000 165722 BIT #4000,@SWR ;NO-TEST FOR ITERATION
1901 013106 001016 BNE SCOPEC ;INHIBIT ITERATION
1902 013110 005737 001034 TST PASSCT ;TEST IF FIRST PASS
1903 013114 001413 BEQ SCOPEG ;BR IF FIRST PASS -- QUICK PASS
1904 013116 023737 013156 001036 CMP SCOPEF,ICOUNT ;COMPARE CURRENT COUNT TO MAX NUMBER
1905 013124 001407 SCOPEJ: BEQ SCOPEG ;EXIT-DONE
1906 013126 005237 013156 INC SCOPEF ;INCREMENT COUNT
1907 013132 022606 SCOPEB: CMP (6)+,SP ;REPOSITION STACK
1908 013134 012677 165656 MOV (6)+,@PSW ;RESTORE PREVIOUS PROCESSOR STATUS
1909 013140 000177 000014 JMP @RETURN ;REPEAT TEST
1910 013144 005037 013156 SCOPEG: CLR SCOPEF ;CLEAR COUNT
1911 013150 011637 013160 MOV @SP,RETURN ;SAVE SCOPE RETURN POINTER
1912 013154 000002 RTI ;RETURN INLINE-NEXT TEST
1913 013156 000000 SCOPEF: 0 ;COUNT LOCATION FOR ITERATION LOOP
1914 013160 002162 RETURN: ADT1+2 ;ADDRESS OF LAST TEST
1915
1916 ;SCOPE AND/OR INTERATION LOOP FOR SOME TESTS 10 TIMES
1917
1918 013162 104013 SCOPEH: TSTTKS
1919 013164 032777 040000 165636 BIT #40000,@SWR
1920 013172 001357 BNE SCOPEB
1921 013174 032777 004000 165626 BIT #4000,@SWR
1922 013202 001360 BNE SCOPEC
1923 013204 005737 001034 TST PASSCT ;TEST IF FIRST PASS
1924 013210 001755 BEQ SCOPEG ;BR IF YES
1925 013212 023727 013156 000010 CMP SCOPEF,#10
1926 013220 000741 BR SCOPEJ
1927
1928 ;SCOPE LOOP FOR SOME TEST 1 TIME
1929
1930 013222 104013 SCOPEI: TSTTKS ;TEST KEYBOARD
1931 013224 032777 040000 165576 BIT #40000,@SWR
1932 013232 001337 BNE SCOPEB
1933 013234 000743 BR SCOPEG

```

```

1934
1935 013236 104000
1936 013240 013354
1937 013242 000207
1938
1939
1940
1941 013244 012737 000006 013344
1942 013252 005037 013352
1943 013256 013537 013346
1944 013262 013737 013346 013350
1945 013270 042737 177770 013350
1946 013276 113737 013352 013351
1947
1948
1949 013314 006237 013346
1950 013320 006237 013346
1951 013324 006237 013346
1952 013330 005237 013352
1953 013334 005337 013344
1954 013340 001350
1955 013342 000205
1956
1957 013344 000006
1958 013346 000000
1959 013350 000000
1960 013352 000000
1961
1962 013354 007 007
1963 013356 042445 042116 050040
1964 013364 051501 020123 020040
1965 013372 100

BELL: PRINT
ENDPAS
RTS PC ;EXIT

;LOAD THE LPS DISPLAY LIGHTS

LEDS: MOV #6,CNTLED
CLR LEDSV3
MOV 2(S)+,LEDSV1
LEDSA: MOV LEDSV1,LEDSV2
BIC #177770,LEDSV2
MOVB LEDSV3,LEDSV2+1

;* MOV LEDSV2,ADDBR ;/ PUT DATA FROM LEDSV2 TO DEVICE REG ADDBR
ASR LEDSV1
ASR LEDSV1
ASR LEDSV1
INC LEDSV3
DEC CNTLED
BNE LEDSA
RTS 5

CNTLED: 6
LEDSV1: 0
LEDSV2: 0
LEDSV3: 0

ENDPAS: .BYTE 7,7
.ASCII '%END PASS 2'
```

Year	Code 1	Code 2	Code 3	Code 4	Code 5	Message
1966						
1967						
1968	013373	000				
1969	013374	022445	046045	051520		TITLE: .ASCII '%%LPS DIAGNOSTIC TEST I, (MAINDEC-11-DRLPH-A)@'
1970	013402	042040	040511	047107		
1971	013410	051517	044524	020103		
1972	013416	042524	052123	044440		
1973	013424	020054	046450	044501		
1974	013432	042116	041505	030455		
1975	013440	026461	051104	050114		
1976	013446	026510	024501	100		
1977						
1978	013453	040	050040	020103		MES1: .ASCII " PC "
1979	013460	020040				
1980	013462	042101	052123	052101		.ASCII "ADSTAT "
1981	013470	040				
1982	013471	101	041104	043125		.ASCII "ADBUFF "
1983	013476	020106				
1984	013500	040440	042104	040515		.ASCII " ADDMA "
1985	013506	040				
1986						
1987	013507	040	042524	050115		.ASCII " TEMP@"
1988	013514	100				
1989						
1990	013515	045	054524	042520		MES4: .ASCII "%TYPE LETTER ' ' TO RUN DESIRED TEST:%"
1991	013522	046040	052105	042524		
1992	013530	020122	020047	020047		
1993	013536	047524	051040	047125		
1994	013544	042040	051505	051111		
1995	013552	042105	052040	051505		
1996	013560	035124	045			
1997	013563	047	023501	040475		.ASCII "'A'=A TO d LOGIC TEST%"
1998	013570	052040	020117	020104		
1999	013576	047514	044507	020103		
2000	013604	042524	052123	045		
2001	013611	047	023503	040475		.ASCII "'C'=A TO D CALIBRATION%"
2002	013616	052040	020117	020104		
2003	013624	040503	044514	051102		
2004	013632	052101	047511	022516		
2005	013640	042047	036447	020101		.ASCII "'D'=A TO D REPEATABILITY%"
2006	013646	047524	042040	051040		
2007	013654	050105	040505	040524		
2008	013662	044502	044514	054524		
2009	013670	045				
2010	013671	047	023505	040475		.ASCII "'E'=A TO D RECOVERY%"
2011	013676	052040	020117	020104		
2012	013704	042522	047503	042526		
2013	013712	054522	045			
2014	013715	047	023506	040475		.ASCII "'F'=A TO D DUAL MODE DYNAMIC LOGIC %@"
2015	013722	052040	020117	020104		
2016	013730	052504	046101	046440		
2017	013736	042117	020105	054504		
2018	013744	040516	044515	020103		
2019	013752	047514	044507	020103		

2020	013760	040045							
2021									
2022	013762	041536	040045		CNTRLC:	.ASCII	'↑C%a'		
2023									
2024	013766	040536	100		CNTRLA:	.ASCII	'↑Aa'		
2025	013771	136	022507	046117	CNTRLG:	.ASCII	'↑G%OLD SWR = a'		
2026	013776	020104	053523	020122					
2027	014004	020075	100						
2028	014007	040	047040	053505	NEWSWR:	.ASCII	' NEW = a'		
2029	014014	036440	040040						
2030									
2031	014020	040045			CRLF:	.ASCII	'%a'		
2032									
2033	014022	027045	100		DOT:	.ASCII	'%.a'		
2034									
2035	014025	077	040040		QMARK:	.ASCII	'? a'		
2036									
2037	014030	021045	020101	047524	MESS:	.ASCII	'%"A TO D LOGIC TEST"% a'		
2038	014036	042040	046040	043517					
2039	014044	041511	052040	051505					
2040	014052	021124	020045	100					
2041									
2042	014057	045	040442	052040	MES7:	.ASCII	'%"A TO D CALIBRATION"%a'		
2043	014064	020117	020104	040503					
2044	014072	044514	051102	052101					
2045	014100	047511	021116	040045					
2046	014106	021045	020101	047524	MES8:	.ASCII	'%"A TO D RECOVERY"%a'		
2047	014114	042040	051040	041505					
2048	014122	053117	051105	021131					
2049	014130	100							
2050									
2051	014131	045	044103	040040	MES9:	.ASCII	'%CH a'		
2052									
2053	014136	023445	023505	052130	MES10:	.ASCII	'%"E'XT. OR 'I'NT. OR 'C'LOCK. SYNC? a"'		
2054	014144	020056	051117	023440					
2055	014152	023511	052116	020056					
2056	014160	051117	023440	023503					
2057	014166	047514	045503	020056					
2058	014174	054523	041516	020077					
2059	014202	100							
2060									
2061									
2062	014203	045	040442	052040	MES11:	.ASCII	'%"A TO D DMA LOGIC TEST"% a'		
2063	014210	020117	020104	046504					
2064	014216	020101	047514	044507					
2065	014224	020103	042524	052123					
2066	014232	022442	040040						
2067	014236	051105	047522	020122	MES12:	.ASCII	'ERROR BIT SET!%a'		
2068	014244	044502	020124	042523					
2069	014252	020524	040045						
2070									
2071	014256	021045	020101	047524	MES13:	.ASCII	'%"A TO D REPEATIBILITY"%a'		
2072	014264	042040	051040	050105					
2073	014272	040505	044524	044502					

H05

2074	014300	044514	054524	040042	
2075					
2076	014306	022445	044103	024056	MES14: .ASCII '%CH.(S)? @'
2077	014314	024523	020077	100	
2078	014321	045	040442	052040	MES15: .ASCII '%A TO D DUAL MODE"%@'
2079	014326	020117	020104	052504	
2080	014334	046101	046440	042117	
2081	014342	021105	040045		
2082					
2083	014346	047503	047125	020124	MES16: .ASCII 'COUNT SPREAD? @'
2084	014354	050123	042522	042101	
2085	014362	020077	100		
2086	014365	045	044447	047047	MES18: .ASCII '"%I'INTERNAL OR 'B'URST OR 'D'UAL SAMPLE MODE? @"
2087	014372	042524	047122	046101	
2088	014400	047440	020122	041047	
2089	014406	052447	051522	020124	
2090	014414	051117	023440	023504	
2091	014422	040525	020114	040523	
2092	014430	050115	042514	046440	
2093	014436	042117	037505	040040	
2094	014444	020045	041440	027110	MES19: .ASCII '% CH. LO AV HI@'
2095	014452	020040	020040	047514	
2096	014460	020040	020040	040440	
2097	014466	020126	020040	020040	
2098	014474	044510	100		
2099					
2100	014477	045	020040	047514	MES20: .ASCII '% LO -5 -4 -3 -2 -1 AV +1 +2 +3 +4 +5 HI@'
2101	014504	020040	032455	020040	
2102	014512	032055	020040	031455	
2103	014520	020040	031055	020040	
2104	014526	030455	020040	053101	
2105	014534	020040	030453	020040	
2106	014542	031053	020040	031453	
2107	014550	020040	032053	020040	
2108	014556	032453	020040	044510	
2109	014564	100			
2110					
2111	014565	045	047520	042527	MES21: .ASCII '%POWER FAILURE @'
2112	014572	020122	040506	046111	
2113	014600	051125	020105	100	
2114	014605	111	050116	052125	MES22: .ASCII 'INPUT CHANNEL PAIR HAVE THE SAME INPUT VALUE%@'
2115	014612	041440	040510	047116	
2116	014620	046105	050040	044501	
2117	014626	020122	040510	042526	
2118	014634	052040	042510	051440	
2119	014642	046501	020105	047111	
2120	014650	052520	020124	040526	
2121	014656	052514	022505	100	
2122	014663	105	051122	051117	MES23: .ASCIZ 'ERROR - FIRST CONVERSION EQUAL TO THE SECOND%@'
2123	014670	026440	043040	051111	
2124	014676	052123	041440	047117	
2125	014704	042526	051522	047511	
2126	014712	020116	050505	040525	
2127	014720	020114	047524	052040	

2128	014726	042510	051440	041505
2129	014734	047117	022504	000100
2130	014742	051461	020124	047503
2131	014750	053116	020056	047516
2132	014756	020124	020075	047062
2133	014764	022504	000100	
2134				
2135				
2136	014770	000700		
2137	014772	000000		
2138	014774	001000		
2139	014776	001540		
2140	015000	000000		
2141	015002	000000		
2142	015004	000000		
2143	015006	000000		
2144	015010	000000		
2145	015012	000000		
2146	015014	000000		
2147	015016	000000		
2148	015020	000000		
2149	015022	000000		
2150	015024	000000		
2151	015026	000000		
2152	015030	000000		
2153	015032	000000		
2154	015034	000000		
2155	015036	000000		
2156	015040	000000		
2157	015042	000000		
2158	015044	000000		
2159	015046	000000		
2160	015050	000000		
2161	015052	000000		
2162	015054	007000		
2163	015056	000000		
2164	015060	000000		
2165	015062	000000		
2166	015064	000000		
2167	015066	000000		
2168	015070	000000		
2169	015072	000000		
2170	015074	000000		
2171	015076	000000		
2172	015100	000000		
2173	015102	000000		
2174	015104	000000		
2175	015106	000000		
2176	015110	000000		
2177	015112	000000		
2178	015114	000000		
2179	015116	000000		
2180	015120	000000		
2181	015122	000000		

MES24: .ASCIZ '1ST COMV. NOT = 2ND%3'

```

.EVEN
;ADDRESS AND CONSTANTS TABLE
ADWORD2: 0 ;LOW BYTE OF 'ADWORD'
PRINT1: 0
STACK: 1000 ;INITIAL SP. ADDRESS
AVECTR: INITA ;'IA' VECTOR ADDRESS
PROC: 0 ;TEMP STORAGE FOR 'PSW'
CHRCNT: 0 ;TEMP STORAGE
COUNT: 0 ;TEMP STORAGE
KSTOR1: 0 ;PERMANENT STORAGE
KSTOR2: 0 ;PERMANENT STORAGE
KSTOR3: 0 ;PERMANENT STORAGE
KSTOR4: 0 ;PERMANENT STORAGE
KSTOR5: 0
KSTR11: 0
KSTR12: 0
STCHAN: 0
FIRST: 0
DELAY: 0
DELAY1: 0
USEDSH: 0
USEDMA: 0
USECLK: 0
OPSI: 0
TEMP: 0
TEMP1: 0 ;TEMPORARY STORAGE
TEMP2: 0 ;TEMPORARY STORAGE
TEMP3: 0 ;TEMPORARY STORAGE
BRLEV1: 0
BRLEV2: 0
BRLEV3: 0
MESPR1: 0
HIOROV: 0
HIGH: 0
LOW: 0
$TMDAT: 0
$BDDAT: 0
$GDDAT: 0
$ZERO: 0
AVERM5: 0
AVERM4: 0
AVERM3: 0
AVERM2: 0
AVERM1: 0
AVRAGE: 0
AVERP1: 0
AVERP2: 0
AVERP3: 0

```

2182 015124 000000
 2183 015126 000000
 2184 015130 000000
 2185 015132 000000
 2186 015134 000000
 2187 015136 000000
 2188 015140 000000
 2189 015142 000000
 2190 015144 000000
 2191 015146 000000
 2192 015150 000000
 2193 015152 000000
 2194 015154 000000
 2195 015156 000000
 2196 015160 000000
 2197 015162 000000
 2198 015164 000000
 2199 015166 000000
 2200 015170 000000
 2201 015172 000000
 2202 015174 000000
 2203 015176 000000
 2204 015200 015202
 2205 015202 023420
 2206 015204 001750
 2207 015206 000144
 2208 015210 000012
 2209 015212 000001
 2210
 2211 015214 000000
 2212
 2213
 2214 015256 015256
 2215 000000 015320
 2216 015320 000000
 2217 015362 015362
 2218 000000 015424
 2219 015424 000000
 2220 015466 015466
 2221 000000 015530
 2222 015530 000000
 2223 015572 015572
 2224 000000 015634
 2225 015634 000000
 2226 000000 015676
 2227 015676 001000
 2228
 2229
 2230
 2231
 2232
 2233
 2234
 2235

AVERP4: 0
 AVERP5: 0
 ORLOW: 0
 MINUS5: 0
 MINUS4: 0
 MINUS3: 0
 MINUS2: 0
 MINUS1: 0
 AVGCNT: 0
 PLUS1: 0
 PLUS2: 0
 PLUS3: 0
 PLUS4: 0
 PLUS5: 0
 ORHIGH: 0
 XSPRD1: 0
 XSPRD2: 0
 XSPRD3: 0
 XSPRD4: 0
 DIAB: 0
 DIAB: 0
 DIAC: 0
 DIAD: .+2
 10000.
 1000.
 100.
 10.
 1.
 ;HERE STARTS THE '512' WORD A/D DATA BUFFER.
 ADBUFF: 0
 ADTB0: 0 .+.40
 ADTB1: 0 .+.40
 ADTB2: 0 .+.40
 ADTB3: 0 .+.40
 ADTB4: 0 .+.40
 ADTB5: 0 .+.40
 ADTB6: 0 .+.40
 ADTB7: 0 .+.40
 .BLKW 512.
 ;*
 ;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
 ;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
 ;*NEXT WE WILL INIT BOTH UPROCESSORS

2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289

017676 013746 000004
017702 000413
017704 012737 017730 000004
017712 005237 170000
017716 104000
017720 017724
017722 000401
017724
017726
017726 000401
017730 022626
017732 012637 000004
017736 005037 020554
017742 004537 020556
017746 000000G
017750 052777 040000 161062
017756
017756 010146
017760 005001
017762 005201
017764 001376
017766 012777 104000 161044
017774 105201
017776 001376

SLPAI:

```

; *THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
; *THE ORDER OF LOAD IS DETERMINED BY THE USER.
; *
; *      CALL=   JSR      R5,SLPAI
; *      .WORD   0              ;ADDR. OF DEVICE ADDRESS.
; * ROUTINES REQUIRED: .LOADLP
; * PROGRAMS REQUIRED: DRLPX2
; *
; *
; *      ;RETURNS WITH $AERR=1 IF SLAVE
; *      ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
; *
MOV      4,-(SP)
BR       31$

;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.

MOV      #30$,4
INC      170000
PRINT
64$:
BR       65$
;ASCII <7>##
65$:
BR       31$
30$:
CMP      (SP)+,(SP)+
31$:
MOV      (SP)+,4
CLR      $AERR
JSR      R5,$LOAD
.WORD   DRLPX2
;LOAD MICRO-CODE.
;FILE "DRLPX2.OBJ"

BIS      #BIT14,$KMADO ;ISSUE KMC+DMC INIT.

1$:
; "HANGS" HERE THEN KMC-11 ERROR.

MOV      R1,-(SP)
CLR      R1
2$:
INC      R1
;STALL FOR DMC-UP
BNE     2$
MOV      #BIT15:BIT11,$KMADO ;SET RUN, AND ENABLE ARBITRATION.
25$:
INCB    R1
BNE     25$

```

```

2290 020000 032777 000040 161032 BIT #BITS, @KMADO ;SLAVE READY? (READING IPBM SR)
2291 020006 001401 BEQ 3$ ;FATAL LPA-11 ERROR SLAVE NOT READY.
2292 ERROR ;
2293 020010 104400 ERROR ;
2294 ;
2295 020012 012777 000004 161024 3$: MOV #4, @KMAD2 ;READ FAST PATH
2296 020020 4$: JSR R5, $TOUT ;-TOUT-CHECK FOR TIMEOUT
2297 020020 004537 021466 ;
2298 ;
2299 020024 104400 ERROR ;/TIME-OUT ERROR
2300 ;/WE FAILED TO COMPLETE
2301 ;/CURRENT OPERATION.
2302 ;/CONTINUES IN THIS LOOP
2303 ;/WOULD MAKE US "HANG" HERE
2304 ;
2305 020026 000774 BR 4$ ;
2306 ;
2307 ;/RETURNS HERE-FROM-TIMED OUT.
2308 020030 122777 000377 161006 CMPB #377, @KMAD2 ;WAIT TILL KMC DONE COMMAND.
2309 020036 001370 BNE 4$ ;
2310 020040 122777 000377 161002 CMPB #377, @KMAD4 ;IF FAST PATH=377 THEN ERROR.
2311 020046 001001 BNE 35$ ;
2312 020050 104400 ERROR ;IPBM ERROR (SLAVE SIDE)
2313 ;YOU MUST RUN IPBM DIAGNOSTIC.
2314 ;
2315 020052 122777 000004 160770 35$: CMPB #4, @KMAD4 ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
2316 020060 001543 BEQ 5$ ;YES-CONTINUE.
2317 020062 005227 177777 INC #-1 ;
2318 020066 001140 BNE 5$ ;
2319 020070 005227 177777 INC #-1 ;
2320 020074 001135 BNE 5$ ;
2321 020076 104000 PRINT ;
2322 020100 020104 66$ ;
2323 020102 000440 BR 67$ ;
2324 020104 66$: ;ASCII <200>"W A R N I N G THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
2325 020204 67$: ;
2326 020204 104000 PRINT ;
2327 020206 020212 68$ ;
2328 020210 000430 BR 69$ ;
2329 020212 68$: ;ASCII <200>"MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED."
2330 020272 69$: ;
2331 020272 104000 PRINT ;
2332 020274 020300 70$ ;
2333 020276 000434 BR 71$ ;
2334 020300 70$: ;ASCII <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
2335 020370 71$: ;
2336 ;
2337 020370 112737 177777 020522 5$: MOVB #0-1, 11$ ;DAC CODE FOR SLAVE.
2338 020376 012501 MOV (5)+, R1 ;GET NEXT DEVICE ADDR.
2339 020400 021127 000000 6$: CMP (R1), #0 ;TERM REACHED?
2340 020404 001444 BEQ 10$ ;
2341 020406 105237 020522 INCB 11$ ;
2342 020412 113777 020522 160430 MOVB 11$, @KMAD4 ;FIFO DATA
2343 020420 004737 020524 JSR PC, 20$ ;ISSUE SEND

```

```

2344 020424 112177 160420      MOVB   (R1)+, @KMA04      ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
2345 020430 004737 020524      JSR    PC, 20$           ;ISSUE SEND
2346 020434 112177 160410      MOVB   (R1)+, @KMA04      ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
2347 020440 004737 020524      JSR    PC, 20$
2348
2349 020444 032777 000002 160366 7$:  BIT    #BIT1, @KMA00      ;WAIT FOR FIFO DATA
2350 020452 001374 000002 160362 7$:  BNE    7$                ;=1 NO DATA. =0 DATA.
2351 020454 112777 000002 160362 7$:  MOVB   #2, @KMA02        ;READ FIFO.
2352
2353 020462 004537 021466 8$:      JSR    R5, $TOUT         ;-TOUT-CHECK FOR TIMEOUT
2354 020462 104400 021466 8$:      ERROR                    ;/TIME-OUT ERROR
2355                                     ;/WE FAILED TO COMPLETE
2356                                     ;/CURRENT OPERATION.
2357                                     ;/CONTINUES IN THIS LOOP
2358                                     ;/WOULD MAKE US "HANG" HERE
2359
2360
2361
2362 020470 000774 000774 8$:      BR     8$
2363
2364
2365 020472 122777 000377 160344 8$:  CMPB   #377, @KMA02      ;/RETURNS HERE-FROM-TIMED OUT.
2366 020500 001370 000377 160344 8$:  BNE    8$                ;WAIT FOR READ.
2367 020502 105777 160342 8$:  TSTB   @KMA04            ;WAS A ZERO RETURNED?
2368 020506 001734 160342 8$:  BEQ    6$                ;YES GET NEXT ADDR.
2369                                     ;SLAVE WILL RETURN CODE 0 IF
2370 020510 005237 020554 8$:  INC    $AERR             ;DEV PRESENT. ELSE
2371                                     ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
2372 020514 005041 020554 8$:  CLR    -(1)              ;GET RID OF REFERENCE TO BAD ADDR.
2373 020516 012601 020554 10$:  MOV    (SP)+, R1
2374 020520 000205 020554 10$:  RTS    R5                ;RETURN ALL ADDR. CHECKED.
2375
2376 020522 000000 000000 11$:  .WORD  0                 ;HOLDS DAC CODE PLUS OFFSET
2377                                     ;TO SLAVES ADDR. TABLE.
2378
2379 020524 112777 000003 160312 20$:  MOVB   #3, @KMA02        ;ISSUE FIFO WRITE
2380 020532 004537 021466 21$:  JSR    R5, $TOUT         ;-TOUT-CHECK FOR TIMEOUT
2381 020532 104400 021466 21$:  ERROR                    ;/TIME-OUT ERROR
2382                                     ;/WE FAILED TO COMPLETE
2383                                     ;/CURRENT OPERATION.
2384                                     ;/CONTINUES IN THIS LOOP
2385                                     ;/WOULD MAKE US "HANG" HERE
2386
2387
2388
2389 020540 000774 000774 21$:  BR     21$
2390
2391
2392 020542 122777 000377 160274 21$:  CMPB   #377, @KMA02      ;/RETURNS HERE-FROM-TIMED OUT.
2393 020550 001370 000377 160274 21$:  BNE    21$              ;KMC CODE WILL RETURN A "377"
2394 020552 000207 000207 21$:  RTS    PC                ;WHEN DONE COMMAND.
2395
2396 020554 000000 000000 $AERR: .WORD  0                 ;=0 IF ADDR. LIST OK, =1 IF BAD.
2397

```

```

2398
2399
2400
2401
2402
2403
2404
2405
2406 020556 010446          $LOAD: MOV R4,-(SP)          ;SAVE R4.
2407 020560 010046          MOV RO,-(SP)          ;SAVE RO.
2408 020562 012500          1$: MOV (5)+,RO          ;GET PROG. ADDR.
2409 020564 005077 160250  CLR 2KMAD0            ;CLEAR CSR
2410 020570 005077 160254  CLR 2KMAD4            ;CLEAR CRAM ADDR.
2411 020574 052777 002000 160236 2$: BIS 2000,2KMAD0        ;SELECT CRAM.
2412 020602 012077 160246  MOV (0)+,2KMAD6        ;WRITE DATA.
2413 020606 052777 020000 160224  BIS 20000,2KMAD0      ;SET CRAM WRITE
2414 020614 005077 160220  CLR 2KMAD0            ;DISABLE CRAM.
2415 020620 005277 160224  INC 2KMAD4            ;UPDATE CRAM ADDR.
2416 020624 021027 177777  CMP (0),#-1          ;ALL DONE?
2417 020630 001361          BNE 2$              ;NO LOOP.
2418 020632 005077 160212  CLR 2KMAD4            ;CLEAR CRAM ADDR.
2419 020636 016500 177776  MOV -2(5),RO         ;GET MICRO CODE ADDR.
2420
2421 020642 052777 002000 160170 3$: BIS 2000,2KMAD0        ;SELECT CRAM
2422 020650 022077 160200  CMP (RO)+,2KMAD6      ;DATA OK?
2423 020654 001013          BNE 5$              ;NO - REPORT AN ERROR.
2424 020656 021027 177777  CMP (0),#-1          ;ALL DONE?
2425 020662 001405          BEQ 4$              ;YES - EXIT
2426 020664 005077 160150  CLR 2KMAD0            ;NO - DESELECT CRAM.
2427 020670 005277 160154  INC 2KMAD4            ;UPDATE CRAM ADDR.
2428 020674 000762          BR 3$
2429
2430 020676 012600          4$: MOV (5)+,RO         ;RESTORE RO
2431 020700 012604          MOV (5P)+,R4         ;RESTORE R4
2432 020702 00 205          RTS R5              ;EXIT
2433
2434 020704          5$: ;COME HERE ON LOAD ERROR
2435 020704 005745          TST -(5)
2436 020706 105204          INCB R4              ;UPDATE ERROR COUNTER.
2437 020710 100324          BPL 1$              ;IF NOT TOO MANY, TRY AGAIN.
2438 020712 000000          HALT                ;MICRO CODE LOAD ERROR.
2439
2440 020714 000722          BR 1$                ;KMC-11 FAULT. YOU COULD TRY
2441 ;TO PRESS CONTINUE TO GIVE IT
2442 ;ANOTHER CHANCE, BUT I DOUBT
2443 ;THAT THAT WOULD WORK. SINCE I'VE
2444 ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
2445 ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
2446
2447
2448
2449
2450
2451
; *THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
; *
; * CALL = JSR R5,$TLKW
; * .WORD 0 ;OFFSET OF DEVICE ADDR.

```



```

2452          ;*          .WORD 0          ;DATA TO BE WRITTEN
2453          ;*
2454          ;*
2454 020716 010046          $TLKW: MOV      RO, -(SP)          ;SAVE RO
2455 020720 012500          MOV      (5)+, RO          ;GET DEVICE OFFSET
2456 020722 052700 000340  BIS      #340, RO          ;ADD WRITE CODE.
2457 020726 004737 021200  JSR      PC, $LPW          ;WAIT FOR FAST PATH READY
2458 020732 010037 021024  MOV      RO, W1
2459 020736 010077 160106  MOV      RO, @KMA4
2460 020742 112777 000005 160074  MOVB    #5, @KMA2          ;ISSUE FAST PATH WRITE
2461 020750 004737 021200  JSR      PC, $LPW          ;WAIT FOR RDY
2462 020754 011537 021026  MOV      (5), W2
2463 020760 112577 160064  MOVB    (5)+, @KMA4          ;WRITE LOW BYTE DATA.
2464
2465 020764 112777 000005 160052  MOVB    #5, @KMA2          ;FP WRITE
2466 020772 004737 021200  JSR      PC, $LPW
2467 020776 111537 021030  MOVB    (5), W3
2468 021002 112577 160042  MOVB    (5)+, @KMA4          ;WRITE HIGH BYTE
2469 021006 112777 000005 160030  MOVB    #5, @KMA2
2470 021014 004737 021200  JSR      PC, $LPW
2471 021020 012600          MOV      (SP)+, RO
2472 021022 000205          RTS      R5          ;EXIT DONE.
2473 021024 000000          W1:    0
2474 021026 000000          W2:    0
2475 021030 000000          W3:    0
2476
2477          ;*
2478          ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
2479          ;*
2480          ;*      CALL = JSR      R5, $TLKR
2481          ;*          .WORD 0          ;OFFSET OF DEVICE
2482          ;*          ;RETURNS HERE
2483          ;*DATA IN WORD $DATR
2484          ;*
2485
2486 021032 010046          $TLKR: MOV      RO, -(SP)          ;SAVE RO
2487 021034 012500          MOV      (5)+, RO          ;GET OFFSET
2488 021036 052700 000300  BIS      #300, RO          ;ADD READ CODE
2489 021042 004737 021200  JSR      PC, $LPW          ;WAIT TILL READY
2490 021046 110077 157776  MOVB    RO, @KMA4
2491 021052 112777 000005 157764  MOVB    #5, @KMA2          ;ISSUE WRITE FP
2492 021060 004737 021200  JSR      PC, $LPW
2493 021064 010037 021174  MOV      RO, RD1
2494 021070          IS:
2495 021070 004537 021466  JSR      R5, $TOUT          ;-TOUT-CHECK FOR TIMEOUT
2496
2497 021074 104400          ERROR          ;/TIME-OUT ERROR
2498          ;/WE FAILED TO COMPLETE
2499          ;/CURRENT OPERATION.
2500          ;/CONTINUES IN THIS LOOP
2501          ;/WOULD MAKE US "HANG" HERE
2502
2503 021076 000774          BR      1$
2504
2505          ;/RETURNS HERE-FROM-TIMED OUT.

```

```

2506 021100 032777 000040 157732 BIT #BITS, @KMADO ;FAST PATH GOT DATA?
2507 021106 001370 BNE 1$ ;
2508 021110 112777 000004 157726 MOVB #4, @KMAD2 ;ISSUE FAST PATH READ
2509 021116 004737 021200 JSR PC, $LPW
2510 021122 117737 157722 021176 MOVB @KMAD4, $DATR ;GET LOW BYTE
2511 021130 2$: JSR R5, $TOUT ;-TOUT-CHECK FOR TIMEOUT
2512 021130 004537 021466 ERROR ;/TIME-OUT ERROR
2513 021134 104400 ;/WE FAILED TO COMPLETE
2514 ;/CURRENT OPERATION.
2515 ;/CONTINUES IN THIS LOOP
2516 ;/WOULD MAKE US "HANG" HERE
2517
2518
2519
2520 021136 000774 BR 2$
2521
2522 ;/RETURNS HERE-FROM-TIMED OUT.
2523 ;/FAST PATH READY?
2524 021140 032777 000040 157672 BIT #BITS, @KMADO
2525 021146 001370 BNE 2$
2526 021150 112777 000004 157666 MOVB #4, @KMAD2 ;ISSUE FAST PATH READ
2527 021156 004737 021200 JSR PC, $LPW
2528 021162 117737 157662 021177 MOVB @KMAD4, $DATR+1 ;SAVE HIGH BYTE
2529 021170 012600 MOV (SP)+, R0
2530 021172 000205 RTS R5
2531 021174 000000 RD1: 0
2532 021176 000000 $DATR: .WORD 0
2533
2534 ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
2535 ;AS FAST PATH TO BE READ.
2536
2537 CALL = JSR PC, $LPW
2538
2539 ;IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
2540 ;THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
2541
2542 021200 010146 $LPW: MOV R1, -(SP) ;SAVE R1
2543 021202 005001 CLR R1
2544 021204 122777 000377 157632 1$: CMPB #377, @KMAD2 ;FINISHED INSTRUCTION?
2545 021212 001403 BEQ 2$
2546 021214 005201 INC R1 ;TIME OUT?
2547 021216 001372 BNE 1$
2548 021220 000411 BR 10$
2549
2550 021222 032777 000020 157610 2$: BIT #BIT4, @KMADO ;FAST PATH READ?
2551 021230 001403 BEQ 3$
2552 021232 005201 INC R1 ;NO - TIME OUT?
2553 021234 001372 BNE 2$
2554 021236 000402 BR 10$ ;YES - REPORT AN ERROR
2555
2556 021240 012601 3$: MOV (SP)+, R1 ;RESTORE R1
2557 021242 000207 RTS PC ;EXIT
2558
2559 021244 10$:

```

2560 021244 104000
 2561 021246 021252
 2562 021250 000407
 2563 021252
 2564 021270
 2565
 2566 021270 000000
 2567 021272 000776
 2568
 2569
 2570
 2571
 2572
 2573
 2574
 2575
 2576
 2577
 2578
 2579
 2580
 2581
 2582 021274 010046
 2583 021276 010146
 2584
 2585 021300 012700 001066
 2586 021304 005001
 2587 021306 005710
 2588 021310 001421
 2589 021312 027520 000000
 2590 021316 001402
 2591 021320 005201
 2592 021322 000771
 2593
 2594 021324 010137 021342
 2595 021330 005725
 2596 021332 013537 021344
 2597 021336 004537 020716
 2598 021342 000000
 2599 021344 000000
 2600 021346 012601
 2601 021350 012600
 2602 021352 000205
 2603 021354 017520 000000
 2604 021360 005010
 2605 021362 004537 017676
 2606 021366 001066
 2607 021370 000755
 2608
 2609
 2610
 2611
 2612
 2613

```

PRINT
64$
BR 65$
;ASCII <200>#LPA-11 FAULT#
64$:
65$:
11$: HALT ;LPA-11 FAULT RUN LPA-11
BR 11$ ;DIAGNOSTICS.

; *
; * 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
; * $TLKW
; *
SOUTLP: MOV R0, -(SP) ;SAVE R0
MOV R1, -(SP) ;SAVE R1
MOV #.DVLS, R0 ;PROGRAM DEFINED LIST.
CLR R1
1$: TST (0) ;TERMINATOR REACHED?
BEQ 10$ ;YES NEXT STEP.
CMP @ (S), (0)+ ;MATCH WITH ADDR IN LIST?
BEQ 2$
INC R1
BR 1$

2$: MOV R1, 3$ ;SAVE OFFSET, DEVICE KNOWN.
TST (S)+
MOV @ (S)+, 4$ ;GET DATA TO BE WRITTEN
JSR R5, $TLKW ;DO WRITE
3$: .WORD 0 ;DEVICE OFFSET
4$: .WORD 0 ;DATA TO BE WRITTEN.
MOV (SP)+, R1
MOV (SP)+, R0
RTS R5
10$: MOV @ (S), (0)+ ;SAVE ADDR.
CLR (0)
JSR R5, $LPAI
.WORD .DVLS
BR 2$

; *
; * THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE
; * TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
; *
; * FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN

```

```

2614 ;*USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
2615 ;*WITH THE NEW ADDR.
2616 ;*WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH
2617 ;*$TLK
2618 ;*
2619 ;* CALL THROUGH MOVEI DATA,ADDR.
2620 ;* WHICH EQUALS:
2621 ;* JSR R5,$INLP
2622 ;* .WORD XX ADDR OF DEVICE
2623 ;* .WORD YY ADDR TO STORE READ DATA.
2624 021372 010046 $INLP: MOV R0,-(SP) ;SAVE R0
2625 021374 010146 MOV R1,-(SP) ;SAVE R1
2626
2627 021376 012700 001066 MOV #.DVLs,R0 ;PROG DEFINED ADDR. LIST.
2628 021402 005001 CLR R1
2629 021404 005710 1$: TST (0) ;EOL REACHED?
2630 021406 001420 BEQ 10$ ;YES - DEFINE NEW ADDR.
2631
2632 021410 027520 000000 CMP @($),(0)+ ;ADDR. MATCH?
2633 021414 001402 BEQ 2$
2634 021416 005201 INC R1
2635 021420 000771 BR 1$
2636
2637 021422 010137 021434 2$: MOV R1,3$ ;SAVE LIST OFFSET
2638 021426 005725 TST ($)+
2639 021430 004537 021032 JSR R5,$TLKR ;GO READ DEVICE
2640 021434 000000 $OFS=.
2641 021434 000000 3$: .WORD 0 ;OFFSET OF DEVICE
2642
2643 021436 013735 021176 MOV $DATA,@($)+ ;STORE DATA.
2644 021442 012601 MOV (SP)+,R1 ;RESTORE R1
2645 021444 012600 MOV (SP)+,R0 ;RESTORE R2
2646 021446 000205 RTS R5 ;EXIT
2647
2648 021450 017520 000000 10$: MOV @($),(0)+
2649 021454 005010 CLR (0)
2650 021456 004537 017676 JSR R5,$LPAI
2651 021462 001066 .WORD .DVLs
2652 021464 000756 BR 2$
2653
2654 ;*
2655 ;*$STOUT ROUTINE USED TO WATCH IF
2656 ;* WE'RE IN A LOOP TOO-LONG
2657 ;* CALL= JSR R5,$STOUT
2658 ;* ERROR X ;RETURNS HERE ON TIMEOUT
2659 ;* BR
2660 ;* ;RETURNS HERE NO ERROR
2661 ;*
2662 021466 020537 021522 $STOUT: CMP R5,$SAD ;SAME ADDR?
2663 021472 001405 BEQ 1$
2664 021474 010537 021522 MOV R5,$SAD ;NO-SAVE THIS ADDR.
2665 021500 005037 021524 CLR $CNT ;CLR CNT AT ADDR.
2666 021504 000403 BR 2$
2667 021506 005237 021524 1$: INC $CNT ;OVERFLOW?

```

```

2668 021512 100402          BMI      3$          ;YES-ERROR RETURN
2669 021514 062705 000004  2$:      ADD      #4,R5      ;NO-NON ERROR RETURN
2670 021520 000205          3$:      RTS      R5          ;RETURN.
2671
2672 021522 000000          $$AD:   .WORD   0          ;CONTAINS LOOP ADDR.
2673 021524 000000          $CNT:   .WORD   0          ;# OF TIMES AT ADDR.
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684 021526 000005          $RESET: RESET          ;RESET THE WORLD.
2685
2686
2687 021540 005737 020554  ;*      MOV      @2$,1$ ;/READ DEVICE REG 2$,PUT DATA IN 1$.
2688 021544 001004          TST      $AERR          ;IF NO ERROR, LOOP
2689 021546 062737 000002 021562 BNE      10$           ;THERE WAS AN ERROR.
2690
2691
2692
2693
2694 021554 000764          ADD      #2,2$          ;UPDATE DEVICE ADDR.
2695 021556
2696 021556 000207          ;YOU SEE, WE HAVE TO PROTECT OUR SELF!
2697 021560 000000          ;IF 2$ CONTAINED A VALID ADDR,WE
2698 021562 160000          ;MUST KEEP TRYING UNTIL WE GENERATE
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713 021564
2714 021564 005737 021646          BR      $RESET          ;AN INVALID ADDR.
2715 021570 100016          10$:     RTS      PC          ;JUNK LOC.
2716 021572 012737 000002 021636 1$:      .WORD   0          ;DUMB ADDR. FORCES INIT OF DMC/KMC.
2717 021600 052777 000115 000040 2$:      .WORD   160000
2718 021606 005037 177776
2719 021612 005737 021636
2720 021616 001375
2721 021620 005077 000022

;* 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,$RESET          ;REPLACES "RESET INSTRUCTION"
;*
;*      ;RETURNS HERE.
;*
;*
;SDELAY- ROUTINE TO GIVE A MINOR DELAY.
;IS NOT TIME DEPENDENT CODE SENCE
;NOT USED TO GET SPECIFIC TIME BUT
;JUST A LITTLE DELAY.
;
;
;THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
;THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
;
;
;CALL= JSR PC, SDELAY
;SDELAY:
;TST      RTCCSR          ;CLOCK PRESENT?
;BPL      10$
;MOV      #2,TIME
;BIS      #15,@RTCCSR      ;START CLOCK
;CLR      PS
;1$:      TST      TIME
;BNE      1$
;CLR      @RTCCSR          ;STOP CLOCK

```

```

2722
2723 021624 000207
2724 021626 105237 021636
2725 021632 001375
2726 021634 000207
2727
2728 021636 000000
2729
2730 021640 005337 021636
2731 021644 000002
2732 021646 000000
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754 021650 012537 021660
2755 021654 004537 021372
2756 021660 000000
2757 021662 021756
2758 021664 113777 021434 157162
2759 021672 113777 021434 157156
2760 021700 013737 021660 021720
2761 021706 062737 000002 021720
2762 021714 004537 021372
2763 021720 000000
2764 021722 021756
2765 021724 113777 021434 157114
2766 021732 152777 000340 157114
2767 021740 152777 000300 157110
2768 021746 152777 000300 157072
2769 021754 000205
2770 021756 000000
2771
2772 000001

```

```

RTS PC
10$: INCB TIME
      BNE 10$
      RTS PC
TIME: .WORD 0
CLKINT: DEC TIME
RTI
RTCCSR: .WORD 0 ;CLOCK CSR IF USED.

```

```

THIS ROUTINE LOOKS THROUGH CURENT .DVL$ 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 RS, $PUTS ;CALL SET UP ROUTINE.
      .WORD ADDR ;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 MOV B 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 RS, $INLP
1$: .WORD 0
   .WORD 10$
   MOVB $OFS, @KMA06
   MOVB $OFS, @KMA07
   MOV 1$, 2$
   ADD #2, 2$
   JSR RS, $INLP
2$: .WORD 0
   .WORD 10$
   MOVB $OFS, @KMA03
   BISB #340, @KMA06
   BISB #300, @KMA07
   BISB #300, @KMA03
   RTS RS
10$: .WORD 0
      .END

```


CALB2C	005364	906	910*																	
CALB2D	005444	912	921*																	
CATORI=	104005	126*	997																	
CATORZ	012162	1631	1750*																	
CATR1	012262	1769*	1771																	
CATR2	012304	1774*	1791	1793	1795															
CATR3	012344	1783*	1787																	
CATR4	012364	1784	1789*																	
CATR5	012410	1775	1799*																	
CHRCNT	015002	1332*	1382*	1399*	1400	1427	1429*	1484	1486*	1505	2141*									
CK	013054	1852	1890*																	
CLKINT	021640	2730*																		
CMPCNT	012160	1126*	1721*	1737	1747*															
CMpra	007724	1302*	1305	1321*																
CMprb	007726	1303*	1304	1322*																
CMprc	007730	1304*	1305*	1308*	1311*	1315*	1318*	1323*												
CMpr1	007676	1307	1311*																	
CMpr2	007710	1306	1309	1312	1315*															
CMpr3	007716	1310	1313	1318*																
CMpte	011774	1630	1719*																	
CMpTEa	012014	1129	1722*																	
CMpUTE=	104004	125*	996																	
CNTLED	013344	1941*	1953*	1957*																
CNTRLA	013766	1412	2024*																	
CNTRLC	013762	297	2022*																	
CNTRLG	013771	1450	2025*																	
COMPR	007634	1206	1259	1275	1279	1302*														
COUNT	015004	897*	991*	1097*	1520	2142*														
CPDLAY	011426	268	1640*																	
CPTIME	011436	268*	688	704	1533	1645*														
CPUTYP	001370	245	256*																	
CRLF	014020	891	930	1024	1031	1056	1213	1266	1294	1438	1868	1873	2031*							
CSB	001140	192*	232*	236*	918															
CSR	001136	191*	231*	235*	915	921														
DECOCT=	104001	122*	964	975	1089															
DECOUT	011674	1689	1695*																	
DECNT	011762	1682*	1686	1688	1693*	1711*														
DECPRT	011576	1632	1680*																	
DEC1	011722	1696	1698	1701*																
DEC2	011736	1700	1703*																	
DELAY	015030	1101*	1106*	1523	1536*	2152*														
DELAY1	015032	688*	689*	690*	691*	695*	704*	705*	1533*	1534*	2153*									
DIAA	015172	2201*																		
DIAB	015174	2202*																		
DIAC	015176	2203*																		
DIAD	015200	2204*																		
DIGCNT	011756	1681*	1690*	1697	1709*															
DIGIT	011754	1684*	1685*	1695	1699*	1702*	1705	1708*												
DISPLA	001032	150*	248*	352*	933*	998*	1173*													
DMK2	001152	200*	1527*	1528																
DOT	014022	311	2033*																	
DRLPX2=	*****	14*	2276																	
DSHTST	006772	115	328	1163*																
DSHTO	007000	1163	1164*																	

G

KSTR12	015022	1862*	1884	2149*				
LAOTB	006752	1127	1150*					
LEDS	013244	304	924	1168	1743	1856	1941*	
LEDSA	013262	1944*	1954					
LEDSV1	013346	1943*	1944	1949*	1950*	1951*	1958*	
LEDSV2	013350	1944*	1945*	1946*	1949	1959*		
LEDSV3	013352	1942*	1946	1952*	1960*			
LGERR1	013006	1866	1872*					
LOGERR	012664	101	1850*					
LOGICA	002022	104	337*					
LOW	015070	1041	1725*	1730	1732*	2168*		
LPAOH	001052	172*						
LPADL	001050	170*						
LPCI	001040	161*						
LPCO	001044	166*						
LPMR	001042	164*						
LPMS1	001054	174*						
LPMS2	001056	176*						
LPSAOD	001000	136*	228	229	230	231	232	
LPSO	001046	168*						
LPSVCT	001002	137*	237	238				
MANCAL	005246	116	893*					
MASK	012646	1820*	1826	1831*	1838*			
MESPR1	015062	957*	1025	1048	1052*	2165*		
MES1	013453	1870	1978*					
MES10	014136	886	2053*					
MES11	014203	2062*						
MES12	014236	1557	2067*					
MES13	014256	956	2071*					
MES14	014306	962	1087	2076*				
MES15	014321	1165	2078*					
MES16	014346	973	2083*					
MES18	014365	978	2086*					
MES19	014444	1028	2094*					
MES20	014477	1051	2100*					
MES21	014565	1604	2111*					
MES22	014605	1215	2114*					
MES23	014663	1268	2122*					
MES24	014742	1296	2130*					
MES4	013515	289	1990*					
MES5	014030	348	2037*					
MES7	014057	884	2042*					
MES8	014106	1085	2046*					
MES9	014131	1138	2051*					
MINUS1	015142	1801	2189*					
MINUS2	015140	1804	2188*					
MINUS3	015136	1807	2187*					
MINUS4	015134	1810	2186*					
MINUS5	015132	1790*	2185*					
MONITR	001546	110	292*	1424	1605			
NEWIN	010144	1380*						
NEWSWR	014007	1454	2028*					
NODMA	002036	332*	342*					
OCTPRT	012522	1634	1816*					

OPSI	015042	960*	2157*											
ORHIGH	015160	1770	1792*	2196*										
ORLOW	015130	1057	1768	1794*	2184*									
OUTPTA	010266	1402*	1403	1432										
OVRHI	012374	1778	1792*											
OVRLO	012402	1780	1794*											
PASSCT	001034	151*	350*	352	839	870*	1166*	1173	1902	1923				
PC	=%000007	86*	227*	292*	335*	337*	349*	780*	793*	813*	826*	865*	869*	922*
		995*	1102*	1107*	1129*	1575*	1602*	1689*	1937*	2343*	2345*	2347*	2394*	2457*
		2461*	2466*	2470*	2489*	2492*	2509*	2526*	2557*	2696*	2723*	2726*		
POPDLY	001006	139*	269*	271*	849									
PIRQ	= 177772	254*	260											
PLUS1	015146	1800	2191*											
PLUS2	015150	1803	2192*											
PLUS3	015152	1806	2193*											
PLUS4	015154	1809	2194*											
PLUS5	015156	2195*												
PRINT	= 104000	121*	286	288	296	310	329	347	883	885	890	929	955	961
		972	977	1023	1027	1030	1050	1055	1084	1086	1137	1164	1212	1214
		1265	1267	1293	1295	1411	1437	1449	1453	1469	1508	1556	1603	1867
		1869	1872	1935	2266	2321	2326	2331	2560					
		285*	1865	1871*	2137*									
PRINT1	014772	889*	893*	901	905	1587*	1594	2140*						
PROC	015000	131*	928	1144										
PRTAVG	= 104012	129*	1037	1040	1043	1046	1141	1364	1451	1874	1877	1880	1883	1886
PRTOCT	= 104010	255*	263	2718*										
PS	= 177776	144*	209*	294*	354*	841*	1175*	1519*	1593*	1650*	1680*	1908*		
PSW	001016	93	298	1580*										
PWFAL	011232	1588	1593*											
PWRUP	011266	330	1470	1509	2035*									
QMARK	014025	2493*	2530*											
RO1	021174	325	1081*											
RECVRY	006356	1081	1086*											
RECVY1	006400	1092*	1136	1145										
RECVY2	006424	322	954*											
REPTST	005540	954	957*	970										
REPT1	005552	968	972*											
REPT2	005642	983	985	988*	1071									
REPT2A	005726	981	984*											
REPT2B	005704	990*	1070											
REPT3	005740	1000	1006	1011	1017	1020	1023*							
REPT4	006136	1026	1029*											
REPT5	006154	1049	1052*											
REPT6	006252	1058*	1061											
REPT6A	006300	1002	1007	1012	1018	1022	1063	1065*						
REPT7	006322	1068	1071*											
REPT7A	006350	1035	1037*											
REPT8	006212	1032*	1033*	1036*	1038	1073*	1140*	1142						
REPT8A	006354	351*	1344*	1909	1911*	1914*								
RETURN	013160	2714	2717*	2721*	2732*									
RTCCSR	021646	79*	214	216*	219*	220	225*	258*	261*	264*	267*	268	269	336*
RO	=%000000	650*	665*	682*	723*	729*	730*	745*	850*	853*	932*	933	1109*	1110
		1111	1112	1113	1114	1115	1116	1117	1122*	1132*	1374	1418*	1444*	1457*
		1458	1460	1494*	1499	1580	1601*	1773*	1776	2407	2408*	2419*	2422	2430*

	130#	312	887	963	974	979	1088	1455	1510	1845					
*TYIN = 104011															
TYPECL 011516	1661	1664#													
TYPERA 011460	1654#	1655	1663	1675											
TYPER1 011502	1657	1660#													
TYPER2 011510	1662#														
TYPNES 011440	1626	1650#													
TYPT1 011624	1684#	1694													
TYPT2 011632	1685#	1687													
TYPT3 011664	1691	1693#													
UAOMR 001142	194#														
USECLK 015040	895*	908*	911	2156#											
USEDMA 015036	894*	958*	982*	1082*	2155#										
USEDSH 015034	896*	959*	986*	1034	1083*	1562	2154#								
VECTOR 001060	179#														
VECTPS 001062	180#														
VERSM 001064	182#														
WHAT 002002	308	332#	341												
WHATA 002012	334#	868													
WHATB 002012	335#														
W1 021024	2458#	2473#													
W2 021026	2463#	2474#													
W3 021030	2467#	2475#													
XLOOP 010020	1343#	1344													
XPRTAV 010070	1363#	1636													
XPTA1 010076	1364#	1370													
XSPACE 010036	1352#	1356	1633												
XSPRD1 015162	1021	1060	1739*	1800*	1801*	1802	2197#								
XSPRD2 015164	1016	1802*	1803*	1804*	1805	2198#									
XSPRD3 015166	1010	1805*	1806*	1807*	1808	2199#									
XSPRD4 015170	1005	1808*	1809*	1810*	2200#										
XTTYIN 010130	96	1374#	1471	1635											
ZERO 011760	1683#	1699	1701*	1710#											
\$AERR 020554	367	2274*	2370*	2396#	2687										
\$BDDAT 015074	379	386	398	405	417	424	437	444	456	464	470	482	489		
	501	508	520	527	539	546	558	565	577	584	596	603	615		
	622	634	641	650	657	665	668	673	682	686	699	712	723		
	734	745	756	783	796	808	816	829	1184	1201	1241	1254	1548		
	1550	2170#													
\$CNT 021524	2665*	2667*	2673#												
\$DATR 021176	2510*	2527*	2531#	2643											
\$GDDAT 015076	2171#														
\$INLP 021372	379	386	398	405	417	424	437	444	456	464	470	482	489		
	501	508	520	527	539	546	558	565	577	584	596	603	615		
	622	634	641	650	657	665	668	673	682	686	699	712	723		
	734	745	748	756	783	796	808	816	829	1184	1188	1190	1195		
	1201	1205	1235	1241	1245	1250	1254	1258	1527	1548	1568	1572	1860		
	1862	2624#	2687	2755	2762										
\$LOAD 020556	2275	2406#													
\$LPAI 017676	2248#	2605	2650												
\$LPW 021200	2457	2461	2466	2470	2489	2492	2509	2526	2542#						
\$OFS = 021434	2640#	2758	2759	2765											
\$OUTLP 021274	362	364	367	377	384	396	403	415	422	435	442	454	462		
	468	480	487	499	506	518	525	537	544	556	563	575	582		
	594	601	613	620	632	639	655	686	695	704	721	729	743		

	751	780	793	806	813	826	849	915	918	921	1178	1181	1193
\$PUTS 021650	1198	1230	1233	1238	1248	1533	1540	1544	1554	1949	2582*		
\$RESET 021526	2754*												
\$SAD 021522	227	292	335	349	780	793	813	826	865	1602	2684*	2694	
\$TLKR 021032	2662	2664*	2672*										
\$TLKW 020716	2486*	2639											
\$TMDAT 015072	2454*	2597											
	364*	367	374*	377	393*	396	412*	415	432*	435	437	451*	454
	465*	468	477*	480	496*	499	515*	518	534*	537	553*	575	572*
	575	591*	594	610*	613	629*	632	652*	655	683*	686	692*	695
	726*	729	748*	751	777*	780	790*	793	803*	806	810*	813	823*
	826	846*	849	915*	918*	921	1174*	1178*	1181	1190*	1193	1195*	1198
	1227*	1230*	1233	1235*	1238	1245*	1248	1527	1530*	1533	1537*	1540	1541*
	1544	1572	2169*										
\$TOuT 021466	2297	2354	2381	2495	2512	2662*							
\$VECT1 001150	199*												
\$ZERO 015100	362	364	384	403	422	442	462	487	506	525	544	563	582
	601	620	639	704	721	743	915	1554	2172*				
	89*	92*	95*	98*	103*	105*	107*	135*	185*	282	307	315	318
	321	324	327	368	380	387	399	406	418	425	438	445	457
	471	483	490	502	509	521	528	540	547	559	566	578	585
	597	604	616	623	635	642	669	674	700	713	731	735	757
	762	765	784	797	809	817	830	867	892	1340	1353	1475*	1666
	1704	1711	1821	1829	1844	1889	1891	2204	2213*	2215*	2217*	2219*	2221*
	2223*	2225*	2227*	2229*	2230*	2640							
.DVL: 001066	184*	223*	356*	2585	2606	2627	2651						

ADC	1734	1742													
ADD	234	235	236	239	772	1036	1118	1308	1331	1342	1366	1465	1499	1620	1653
	1688	1693	1733	1800	1801	1803	1804	1806	1807	1809	1810	1818	2669	2689	2761
ASL	267	1334	1335	1336	1462	1463	1464	1496	1497	1498	1618	1789			
ASR	689	690	691	1738	1949	1950	1951								
BEQ	282	307	368	380	387	399	406	418	425	438	445	457	471	483	490
	502	509	521	528	540	547	559	566	578	585	597	604	616	623	635
	642	674	784	797	817	830	840	867	912	927	938	968	1022	1035	1068
	1285	1306	1333	1428	1434	1459	1488	1529	1563	1565	1661	1698	1775	1784	1903
	1905	1924	2291	2316	2340	2368	2425	2545	2551	2588	2590	2630	2633	2663	
BGT	1356	1492	1502	1574	1731										
BIC	313	761	888	899	993	1033	1066	1099	1104	1330	1381	1393	1461	1493	1619
	1945														
BICB	1826														
BIS	355	726	748	842	903	907	910	987	994	1052	1100	1105	1171	1178	1195
	1230	1235	1420	1445	1702	1827	2278	2411	2413	2421	2456	2488	2717		
BISB	1190	2766	2767	2768											
BIT	673	851	926	999	1001	1135	1210	1263	1291	1554	1851	1898	1900	1919	1921
	1931	2290	2349	2506	2523	2550									
BITB	1828														
BLE	1312	1485	1728												
BLT	1490														
BMI	658	700	970	1395	1397	1401	1778	1780	2668						
BNE	222	270	280	315	318	321	324	327	660	731	753	762	765	838	852
	854	856	859	862	902	906	981	985	1000	1002	1004	1006	1009	1011	1015
	1017	1020	1026	1049	1061	1120	1133	1136	1209	1211	1262	1264	1278	1282	1292
	1370	1410	1423	1426	1436	1448	1506	1524	1535	1555	1616	1657	1691	1696	1736
	1741	1756	1764	1771	1787	1833	1852	1866	1899	1901	1920	1922	1932	1954	2285
	2288	2309	2311	2318	2320	2350	2366	2393	2417	2423	2507	2524	2547	2553	2688
	2720	2725													
BPL	669	687	696	706	713	735	757	809	1063	1185	1202	1217	1242	1255	1270
	1298	1307	1309	1340	1353	1391	1403	1549	1551	1559	1655	1666	1670	1673	1687
	1704	1829	1844	1889	1891	2437	2715								
BR	245	290	331	892	904	909	931	939	983	1007	1012	1018	1145	1272	1289
	1310	1313	1337	1405	1432	1466	1468	1500	1540	1568	1663	1675	1694	1700	1791
	1793	1795	1821	1926	1933	2251	2268	2271	2305	2323	2328	2333	2362	2389	2428
	2440	2503	2520	2548	2554	2562	2567	2592	2607	2635	2652	2666	2694		
CLR	223	265	266	276	285	300	302	332	350	354	356	651	729	751	841
	843	893	894	895	896	957	958	959	960	1082	1083	1095	1166	1167	1175
	1287	1315	1329	1343	1357	1382	1383	1384	1385	1386	1387	1388	1389	1456	1481
	1482	1483	1519	1522	1536	1650	1680	1722	1769	1781	1862	1910	1942	2274	2283
	2372	2409	2410	2414	2418	2426	2543	2586	2604	2628	2649	2665	2718	2721	
CLRB	465														
CMP	221	246	279	379	398	417	437	456	482	501	520	539	558	577	596
	615	634	764	861	901	937	969	1003	1005	1008	1010	1014	1016	1019	1021
	1053	1060	1067	1284	1339	1400	1419	1491	1501	1697	1727	1730	1770	1777	1779
	1783	1786	1904	1907	1925	2272	2339	2416	2422	2424	2589	2632	2662		
CMPB	314	317	320	323	326	905	980	984	1394	1396	1409	1422	1425	1433	1435
	1447	1487	1489	1528	1656	1660	2308	2310	2315	2365	2392	2544			
COM	982	986	1318	1564											
DEC	695	705	853	855	858	1119	1132	1332	1355	1369	1429	1486	1534	1573	1672
	1735	1740	1755	1761	1763	1774	1832	1953	2730						
EMT	121	122	123	124	125	126	127	128	129	130	131	132	133		
HALT	90	91	92	763	766	1064	1218	1271	1299	1560	1589	1617	1788	1892	2438

INC	2566	233	271	659	730	752	860	870	1069	1101	1106	1245	1283	1399	1685
	219	1753	1785	1790	1792	1794	1871	1906	1952	2265	2284	2317	2319	2370	2415
INCB	1690	2546	2552	2591	2634	2667									
JMP	2427	2287	2341	2436	2724										
JSR	857	111	112	115	116	283	308	316	319	322	325	328	333	341	868
	110	1070	1071	1286	1300	1341	1345	1421	1424	1471	1511	1605	1622	1909	
	871	292	304	335	337	349	362	364	367	377	379	384	386	396	398
	227	405	415	417	422	424	435	437	442	444	454	456	462	464	468
	403	480	482	487	489	499	501	506	508	518	520	525	527	537	539
	470	546	556	558	563	565	575	577	582	584	594	596	601	603	613
	544	620	622	632	634	639	641	650	655	657	665	668	673	682	686
	615	699	704	712	721	723	726	729	734	743	745	748	751	756	780
	695	793	796	806	808	813	816	826	829	849	865	869	915	918	921
	783	924	995	1102	1107	1129	1168	1178	1181	1184	1188	1190	1193	1195	1198
	922	1205	1206	1230	1233	1235	1238	1241	1245	1248	1250	1254	1258	1259	1275
	1201	1527	1533	1540	1544	1548	1554	1568	1572	1602	1689	1743	1856	1860	1862
	1279	2275	2297	2343	2345	2347	2354	2381	2457	2461	2466	2470	2489	2492	2495
	1949	2512	2526	2597	2605	2639	2650	2687	2755	2762					
MOV	2509	208	209	214	215	216	217	220	225	228	229	230	231	232	237
	208	243	247	248	256	257	258	268	269	273	274	275	277	284	293
	238	295	298	299	301	303	309	336	351	352	353	364	374	393	412
	294	451	477	496	515	534	553	572	591	610	629	650	652	665	682
	432	688	692	704	723	745	768	770	771	777	790	803	810	823	844
	683	846	849	850	882	889	897	900	908	915	918	921	923	932	933
	845	965	966	971	976	988	989	991	998	1032	1057	1058	1065	1081	1090
	954	1092	1094	1096	1097	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117
	1091	1123	1124	1125	1126	1127	1128	1130	1140	1163	1172	1173	1302	1303	1304
	1122	1344	1354	1363	1367	1374	1375	1376	1377	1378	1379	1380	1392	1398	1413
	1328	1415	1416	1417	1418	1430	1439	1440	1441	1442	1443	1444	1457	1460	1467
	1414	1480	1494	1495	1504	1520	1521	1533	1541	1572	1580	1581	1582	1583	1584
	1479	1586	1587	1588	1593	1594	1595	1596	1597	1598	1599	1600	1601	1612	1614
	1585	1651	1652	1658	1664	1667	1668	1681	1682	1683	1684	1699	1701	1705	1719
	1621	1721	1723	1724	1725	1726	1729	1732	1737	1750	1751	1752	1754	1758	1759
	1720	1762	1768	1772	1773	1776	1782	1799	1802	1805	1808	1816	1817	1819	1820
	1760	1834	1853	1854	1863	1864	1908	1911	1941	1943	1944	2249	2264	2273	2282
	1831	2295	2338	2373	2406	2407	2408	2412	2419	2430	2431	2454	2455	2458	2459
	2286	2471	2486	2487	2493	2528	2542	2556	2582	2583	2585	2594	2596	2600	2601
	2462	2624	2625	2627	2637	2643	2644	2645	2648	2664	2716	2754	2760		
	2603	992	1098	1103	1174	1227	1404	1527	1530	1537	1662	1671	1825	1830	1946
MOVB	898	2342	2344	2346	2351	2379	2460	2463	2465	2467	2468	2469	2490	2491	2508
	2337	2525	2527	2758	2759	2765									
	2510	114	338	339	340	707	708	709	935	1054	1170				
NOP	113														
RESET	2684														
ROL	1822	1823	1824												
ROR	1739														
RTI	256	1358	1371	1446	1507	1659	1692	1745	1811	1835	1848	1853	1893	1912	2731
RTS	1316	1319	1575	1706	1937	1955	2374	2394	2432	2472	2529	2557	2602	2646	2670
	2696	2723	2726	2769											
SBC	261	264													
SEC	259	262													
SUB	769	1305	1311	1613	1686	1855									
TRAP	120														
TST	244	260	278	281	306	367	386	405	424	444	464	470	489	508	527

	546	565	584	603	622	641	699	712	783	796	816	829	837	839	866
	911	967	1025	1034	1048	1062	1131	1216	1269	1297	1427	1431	1458	1484	1503
	1505	1523	1550	1558	1562	1615	1695	1865	1890	1902	1923	2435	2587	2595	2629
TSTB	2638	2687	2714	2719											
	263	657	668	686	734	756	808	1184	1201	1241	1254	1352	1390	1402	1548
.ASCII	1654	1665	1669	1674	1703	1843	1988	2367							
	1963	1969	1978	1980	1982	1984	1967	1990	1997	2001	2005	2010	2014	2022	2024
	2025	2028	2031	2033	2035	2037	2042	2046	2051	2053	2062	2067	2071	2076	2078
	2083	2086	2094	2100	2111	2114	2270	2325	2330	2335	2564				
.ASCIZ	2122	2130													
.ASECT	14														
.BLKW	185	2230													
.BYTE	1962	1968													
.ENABL	30														
.END	2772														
.ENDC	182	226	2300	2307	2357	2364	2384	2391	2498	2505	2515	2522	2734		
.EVEN	2134	2270	2325	2330	2335	2564									
.GLOBL	14														
.IF	179	226	2299	2305	2356	2362	2383	2389	2497	2503	2514	2520	2734		
.IFF	2299	2305	2356	2362	2383	2389	2497	2503	2514	2520					
.LIST	14	30	92	362	364	367	377	379	384	386	396	398	403	405	415
	417	422	424	435	437	442	444	454	456	462	464	468	470	480	482
	487	489	499	501	506	508	518	520	525	527	537	539	544	546	556
	558	563	565	575	577	582	584	594	596	601	603	613	615	620	622
	632	634	639	641	650	655	657	665	668	673	682	686	695	699	704
	712	721	723	726	729	734	743	745	748	751	756	780	783	793	796
	806	808	813	816	826	829	849	915	918	921	1178	1181	1184	1188	1190
	1193	1195	1198	1201	1205	1230	1233	1235	1238	1241	1245	1248	1250	1254	1258
	1527	1533	1540	1544	1548	1554	1568	1572	1860	1862	1949	2270	2325	2330	2335
.MACRO	2564	2687													
.NLIST	17	18	19	20	22	24	25	26	27	28	29	30	187	202	
	14	30	92	362	364	367	377	379	384	386	396	398	403	405	415
	417	422	424	435	437	442	444	454	456	462	464	468	470	480	482
	487	489	499	501	506	508	518	520	525	527	537	539	544	546	556
	558	563	565	575	577	582	584	594	596	601	603	613	615	620	622
	632	634	639	641	650	655	657	665	668	673	682	686	695	699	704
	712	721	723	726	729	734	743	745	748	751	756	780	783	793	796
	806	808	813	816	826	829	849	915	918	921	1178	1181	1184	1188	1190
	1193	1195	1198	1201	1205	1230	1233	1235	1238	1241	1245	1248	1250	1254	1258
	1527	1533	1540	1544	1548	1554	1568	1572	1860	1862	1949	2270	2325	2330	2335
.PSECT	14														
.REM	1	14	30												
.REPT	92														
.TITLE	30														
.WORD	162	165	167	169	171	173	175	177	179	180	182	184	199	362	364
	367	377	379	384	386	396	398	403	405	415	417	422	424	435	437
	442	444	454	456	462	464	468	470	480	482	487	489	499	501	506
	508	518	520	525	527	537	539	544	546	556	558	563	565	575	577
	582	584	594	596	601	603	613	615	620	622	632	634	639	641	650
	655	657	665	668	673	682	686	695	699	704	712	721	723	726	729
	734	743	745	748	751	756	780	783	793	796	806	808	813	816	826
	829	849	915	918	921	1178	1181	1184	1188	1190	1193	1195	1198	1201	1205
	1230	1233	1235	1238	1241	1245	1248	1250	1254	1258	1527	1533	1540	1544	1548

1554	1568	1572	1860	1862	1949	2276	2376	2396	2531	2598	2599	2606	2641	2651
2672	2673	2687	2697	2698	2728	2732	2756	2757	2763	2764	2770			

000000

ERRORS DETECTED: 0

J07

LPS-11 DIAGNOSTIC TEST I MAINDEC-11-DRLPH-A
DRLPH.P11

MACY11 27(654) 15-DEC-77 08:34 PAGE 76

SEQ 0087

*DRLPH,DRLPH/SOL/CRF=DRLPA.MAC,DRLPH
RUN-TIME: 7 10 1 SECONDS
CORE USED: 21K