

# LPA/DR11-K

DIAGNOSTIC TEST  
MD-11-DRLPF-A

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

MAR 1978  
**digital**  
MADE IN USA



PRODUCT CODE: MAINDEC-11-DRLPF-A-D  
PRODUCT NAME: LPA/DR11-K 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.



## TABLE OF CONTENTS

SEQ 0002

- 1. ABSTRACT
- 2. EQUIPMENT REQUIREMENTS
- 3. LOADING PROCEDURE
- 4. STARTING PROCEDURE
  - 4.1 CONTROL SWITCH SETTINGS
  - 4.2 STARTING ADDRESSES
- 5. OPERATING PROCEDURE
- 6. ERRORS
- 7. RESTRICTIONS
- 8. MISCELANEOUS
  - 8.1 EXECUTION TIMES
  - 8.2 DEVICE ADDRESSES
- 9. PROGRAM DESCRIPTION
  - 9.1 LOGIC TEST
  - 9.2 CONTROL LINE LOOP
- 10. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY



1. ABSTRACT

THIS PROGRAM IS A LOGIC TEST OF THE DR11-K DIGITAL INPUT OUTPUT CONTROL OPTION. MOST OPTION FUNCTIONS CAN BE TESTED.

DUE TO THE FLEXIBILITY OF THE OPTION, THE OPERATOR WILL BE REQUIRED TO SUPPLY OPTION CHARACTERISTICS. THE PROGRAM WILL HANDLE ALL CONFIGURATIONS OF INPUT INTERRUPT SWITCHES AND INPUT DATA LATCHING JUMPERS. FOR SYSTEMS WITH CONSECUTIVE MULTIPLE DR11-K'S, THESE CONFIGURATIONS MUST BE THE SAME. THE FOLLOWING JUMPERS MUST BE INSERTED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A.

THIS PROGRAM WILL TEST SEQUENTIAL DR11-K'S STARTING AT THE BUS ADDRESS AND VECTOR IN LOCATIONS "SBASE" AND "SVECT1". FOR NORMAL FACTORY CONFIG., ALL QUESTIONS SHOULD BE ANSWERED WITH A VALUE OF 0.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZDRG-E" IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE DR11-K OPTION WHEN IT IS ON THE LP11-KX I/O BUS. NO RE-CABLING IS NEEDED. SOME TEST DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION TEST, WERE DELETED AS THEY COULD NOT BE CHECKED. IF THIS DIAGNOSTIC DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO RUN "MD-DZDRG-E" YOU SHOULD RUN "MD-11-DRLPA" BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 10.

AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC TEST MUST BE RUN.

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERMINAL AND 16K OF MEMORY  
DR11-K OPTION INSTALLED  
BCOBR-1 ONE FOOT OUTPUT TO INPUT WRAPAROUND CABLE  
LP11-KX OPTION

3. LOADING PROCEDURE

THE PROCEDURE FOR LOADING BINARY TAPES SHOULD BE FOLLOWED.



4. STARTING PROCEDURE

## 4.1 CONTROL SWITCH SETTINGS (LOGIC TEST)

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

## CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G (<↑G>); THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXX NEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
  - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED) IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
  - B) IF A CONTROL U (<↑U>) IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

WILL BE USED AS THE SOFTWARE DISPLAY REGISTER.

SW 15 = 1	100000	HALT ON ERROR
SW 14 = 1	040000	LOOP ON CURRENT SUB-TEST
SW 13 = 1	020000	INHIBIT ERROR TYPINGS
SW 12 = 1	010000	LOOP ON CURRENTLY SELECTED DR11-K
SW 11 = 1	004000	INHIBIT SUB-TEST INTERACTIONS
SW 10 = 1	002000	NO OUTPUT TO INPUT WRAPAROUND CABLE
SW 09 = 1	001000	LOOP ON ERROR
SW 08 = 1	000400	LOOP ON TEST IN SWR <7:0>

## 4.2 STARTING ADDRESSES

200 IS THE STARTING ADDRESS OF THE LOGIC TEST.  
 204 IS THE RESTART ADDRESS OF THE LOGIC TEST.  
 210 IS THE STARTING ADDRESS OF THE CONTROL LINE LOOP.



## 5. OPERATING PROCEDURE

THE FOLLOWING JUMPERS MUST BE INSTALLED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A  
 IF THE CUSTOMER HAS SELECTED THE "B" SECTION OF THESE JUMPERS IT MUST BE RETURNED TO THE "FACTORY" POSITION BEFORE RUNNING THE LOGIC TEST. \*\* WORST CASE WILL ONLY BE CHANGING THREE JUMPERS. \*\*  
 THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH REGISTER WHEN REQUESTED BY THE PROGRAM OR AN ERROR WILL OCCUR. WITH THE INPUT INTERRUPT SWITCHES AND DATA LATCHING JUMPERS IN THE FACTORY POSITION, ALL SWITCH REGISTER BITS SHOULD BE RESET. ONCE STARTED THE TEST WILL RUN IN IT'S NORMAL MANNER WITHOUT OPERATOR INTERVENTION OR SWITCH SELECTION.  
 \*\*\* NOTE: OPERATOR MUST INSERT INFORMATION WHEN REQUESTED BY PROGRAM. THE MACHINE WILL TYPE: NEW = AFTER NEW = THE INFORMATION IS INSERTED. REFER TO SECTION 4.1 2) \*\*\*

## 6. ERRORS

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

BYTE	\$TSINM: (LOC. 1102)	CURRENT TEST NUMBER
BYTE	\$ITEMB: (LOC. 1114)	ITEM #N ERROR TABLE INDEX
WORD	\$ERRPC: (LOC. 1116)	ERRORING P.C.
WORD	\$PASS: (LOC. 1176)	CURRENT PASS COUNT

## 7. RESTRICTIONS

1. IF SEQUENTIAL DR11-K'S, ALL DR11-K'S MUST BE IN THE SAME INTERRUPT SWITCHES AND DATA PATH JUMPER CONFIGURATION.
2. THE FOLLOWING JUMPERS MUST BE IN THE "FACTORY" POSITION:  
W21A, W22A AND W23A
3. THE OPERATOR MUST SUPPLY THE CORRECT INTERRUPT SWITCHES AND DATA PATH JUMPER AND SWITCH CONFIGURATION INFORMATION TO THE INITIALIZATION QUESTIONS OR AN ERROR WILL OCCUR.
4. FOR MULTIPLE GROUPS OF CONSECUTIVE DR11-K'S:  
THIS DIAGNOSTIC MUST BE RUN FOR EACH GROUP.
5. AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC MUST BE RUN FIRST



## 8. MISCELANEOUS

## 8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 90 SECONDS FOR COMPLETION ON A PDP11/05 TYPE AND WILL TYPE 'END PASS NNNN.'. THE CONTROL LINE LOOP WILL NEVER EXIT.

## 8.2 DEVICE ADDRESS PROGRAM LOCATIONS

"\$BASE" (LOC. 1244) CONTAINS THE DR11-K BASE DEVICE ADDRESS <767770>  
 "\$VECT1" (LOC. 1240) THE LOW 9 BITS CONTAIN THE DR11-K BASE INTERRUPT VECTOR <300>  
 "\$VECT1" (LOC. 1240) THE HIGH 3 BITS CONTAIN THE DR11-K BR LEVEL #4 <200>

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

## 8.3 USER LINK TO I/O DEVICE

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

## PROCEDURE:

1) START THE PROCESSOR AT LOCATION \$UTK:

2) THE DIALOG TO EXAMINE A LOCATION IS AS FOLLOWS:

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

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

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

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

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

NOTE: THE OPERATORS RESPONSE IS ENCLOSED IN QUOTES.

## 9. PROGRAM DESCRIPTION THE LOGIC TEST MUST BE RUN FIRST AFTER INITIAL PROGRAM LOADING

## 9.1 LOGIC TEST &lt;SA 200 AND 204&gt;

THE LOGIC TEST IS A TEST OF THE CONTROL AND INPUT/OUTPUT REGISTERS. ALL JUMPERS AND SWITCHES COMBINATIONS EXCEPT:

W21A, W22A AND W23A CAN BE DIAGNOSED.

H01

THE PROGRAM CHECKS THAT THE DR11-K  
"RESET" WILL WORK CORRECTLY.

SEQ 0007

9.2 CONTROL LINE LOOP <SA 210>

THIS TEST LOOP PROVIDES THE OPERATOR WITH A  
SCOPE LOOP FOR CHECKING W21, W22 AND W23 IN THE "B" POSITION.



10. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY  
-----

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

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

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

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

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

# J01

THE LPA11-KX DIAGNOSTIC KIT WILL INCLUDE:

SEQ 0009

<u>OPTION</u>	<u>GROUP</u>	<u>DIAG. #</u>	<u>DIAG. TITLE</u>
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/DMC BASIC MICRO-CPU R/W TEST
	B	MD-11-DZLPM	LPA/DMC JMP+ROM READ TEST



63	BASIC DEFINITIONS
173	OPERATIONAL SWITCH SETTINGS
186	TRAP CATCHER
195	STARTING ADDRESS(ES)
204	ACT11 HOOKS
215	APT PARAMETER BLOCK
237	COMMON TAGS
280	APT MAILBOX-ETABLE
329	ERROR POINTER TABLE
482	INITIALIZE THE COMMON TAGS
583	TYPE PROGRAM NAME
590	GET VALUE FOR SOFTWARE SWITCH REGISTER
638	DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
750	T1 TEST FOR NO BUS ERRORS
765	T2 TEST THAT OUTPUT REG. CAN HOLD #-1
779	T3 TEST THAT RESET CLEARS OUTPUT REG.
796	T4 TEST THAT OUTPUT REG. CAN HOLD #52525
811	T5 TEST THAT OUTPUT REG. CAN HOLD #125252
826	T6 TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
844	T7 FLOAT A 1 ACROSS THE OUTPUT REGISTER
866	T10 FLOAT A 0 ACROSS THE OUTPUT REGISTER
890	T11 TEST FOR SLOW OUTPUT GATES WITH #125252
943	T12 TEST FOR SLOW OUTPUT GATES WITH #52525
997	T13 TEST OUTPUT DATA ACCEPT FLAG
1017	T14 TEST OUTPUT INTERRUPT ENABLE
1034	T15 TEST INPUT DATA READY FLAG
1048	T16 TEST INPUT INTERRUPT ENABLE
1067	T17 TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
1094	T20 TEST INPUT WITH #-1
1116	T21 TEST INPUT WITH #52525
1138	T22 TEST INPUT WITH #125252
1159	T23 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
1208	T24 FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
1254	T25 FLOAT A 1 ACROSS LATCHING INPUT BITS
1293	T26 FLOAT A 0 ACROSS LATCHING INPUT BITS
1335	T27 TEST FOR SLOW INPUT GATES WITH #125252
1515	T30 TEST FOR SLOW INPUT GATES WITH #52525
1672	T31 TEST THAT RESET CLEARS INPUT REGISTER BITS
1696	T32 TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
1716	T33 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
1739	T34 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
1810	T35 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
1864	T36 DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
1885	END OF PASS ROUTINE
1924	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1992	MISC. EXTERNAL LOGIC TEST
2035	COULTER INTERFACE TEST
2090	LOC-BOX LAMP AND SWITCH LOOP
2164	XLO1 ADJUSTMENT ROUTINE
2499	SCOPE HANDLER ROUTINE
2564	ERROR HANDLER ROUTINE
2616	ERROR MESSAGE TIMEOUT ROUTINE
2664	BINARY TO OCTAL (ASCII) AND TYPE
2742	POWER DOWN AND UP ROUTINES

LO1

DR11-K LOGIC TEST MAINDEC-11-DRLPF-A MACY11 27(654) 14-DEC-77 20:26  
DRLPF.P11 TABLE OF CONTENTS

SEQ 0011

2793	TYPE ROUTINE
2872	READ AN OCTAL NUMBER FROM THE TTY
2910	TTY INPUT ROUTINE
3080	APT COMMUNICATIONS ROUTINE
3137	TRAP DECODER
3160	TRAP TABLE



.REM [

LPA.MAC

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

GOOD LUCK !

[  
.GLOBL DRLPX2

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

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

.REM !

THIS IS A LIST OF TESTS DELETED FROM THIS DIAGNOSTIC. THESE TESTS COULD NOT BE DONE THROUGH THE LPA-11

TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN  
 TEST FOR UNEXPECTED INTERRUPT  
 TEST THAT THE INPUT CAN INTR. USING MAINT. BIT  
 TEST THAT THE INPUT INTR. CLEAR INT. ENABLE VIA MAINT. BIT  
 TEST THAT THE OUTPUT CAN INTR. USING MAINT. BIT  
 TEST FOR INTR. FROM DRA INPUT TEST FOR INTR. FROM DRA OUTPUT  
 PRE INTERRUPT SETUP  
 TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT  
 TEST FOR NO INTR. FROM DRA INPUT LEVEL INDICATED VIA MAINT. INT.  
 TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN  
 TEST THAT RESET CLEARS DIGITAL STATUS REGISTER

```

        .TITLE DR11-K LOGIC TEST MAINDEC-11-DRLPF-A
        *COPYRIGHT (C) 1978
        *DIGITAL EQUIPMENT CORP.
        *MAYNARD, MASS. 01754
        *
        *PROGRAM BY EDWARD C. BADGER
        *
        *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
        *PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
        *
    
```

167770  
100300

```

        ABASE=167770
        AVECT1=100300
        .SBTTL BASIC DEFINITIONS
    
```

001100

```

        *INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
        STACK= 1100
        .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
        .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
    
```

000011  
000012  
000015  
000200  
177776  
177774  
177772  
177570  
177570

```

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

000000  
000001

```

        *GENERAL PURPOSE REGISTER DEFINITIONS
        R0= %0 ;;GENERAL REGISTER
        R1= %1 ;;GENERAL REGISTER
    
```



```

84      000002      R2=      %2      :: GENERAL REGISTER
85      000003      R3=      %3      :: GENERAL REGISTER
86      000004      R4=      %4      :: GENERAL REGISTER
87      000005      R5=      %5      :: GENERAL REGISTER
88      000006      R6=      %6      :: GENERAL REGISTER
89      000007      R7=      %7      :: GENERAL REGISTER
90      000006      SP=      %6      :: STACK POINTER
91      000007      PC=      %7      :: PROGRAM COUNTER
92
93      : *PRIORITY LEVEL DEFINITIONS
94      000000      PR0=      0      :: PRIORITY LEVEL 0
95      000040      PR1=      40      :: PRIORITY LEVEL 1
96      000100      PR2=      100      :: PRIORITY LEVEL 2
97      000140      PR3=      140      :: PRIORITY LEVEL 3
98      000200      PR4=      200      :: PRIORITY LEVEL 4
99      000240      PR5=      240      :: PRIORITY LEVEL 5
100     000300      PR6=      300      :: PRIORITY LEVEL 6
101     000340      PR7=      340      :: PRIORITY LEVEL 7
102
103     : * "SWITCH REGISTER" SWITCH DEFINITIONS
104     100000      SW15=     100000
105     040000      SW14=      40000
106     020000      SW13=      20000
107     010000      SW12=      10000
108     004000      SW11=       4000
109     002000      SW10=       2000
110     001000      SW09=       1000
111     000400      SW08=        400
112     000200      SW07=        200
113     000100      SW06=        100
114     000040      SW05=         40
115     000020      SW04=         20
116     000010      SW03=         10
117     000004      SW02=          4
118     000002      SW01=          2
119     000001      SW00=          1
120     .EQUIV      SW09, SW9
121     .EQUIV      SW08, SW8
122     .EQUIV      SW07, SW7
123     .EQUIV      SW06, SW6
124     .EQUIV      SW05, SW5
125     .EQUIV      SW04, SW4
126     .EQUIV      SW03, SW3
127     .EQUIV      SW02, SW2
128     .EQUIV      SW01, SW1
129     .EQUIV      SW00, SW0
130
131     : * DATA BIT DEFINITIONS (BIT00 TO BIT15)
132     100000      BIT15=     100000
133     040000      BIT14=      40000
134     020000      BIT13=      20000
135     010000      BIT12=      10000
136     004000      BIT11=       4000
137     002000      BIT10=       2000

```

138	001000	BIT09=	1000
139	000400	BIT08=	400
140	000200	BIT07=	200
141	000100	BIT06=	100
142	000040	BIT05=	40
143	000020	BIT04=	20
144	000010	BIT03=	10
145	000004	BIT02=	4
146	000002	BIT01=	2
147	000001	BIT00=	1
148		.EQUIV	BIT09,BIT9
149		.EQUIV	BIT08,BIT8
150		.EQUIV	BIT07,BIT7
151		.EQUIV	BIT06,BIT6
152		.EQUIV	BIT05,BIT5
153		.EQUIV	BIT04,BIT4
154		.EQUIV	BIT03,BIT3
155		.EQUIV	BIT02,BIT2
156		.EQUIV	BIT01,BIT1
157		.EQUIV	BIT00,BIT0

158		:*BASIC "CPU" TRAP VECTOR ADDRESSES	
159		ERRVEC=	4
160	000004	RESVEC=	10
161	000010	TBITVEC=	14
162	000014	TRTVEC=	14
163	000014	BPTVEC=	14
164	000014	IOTVEC=	20
165	000020	PWRVEC=	24
166	000024	EMTVEC=	30
167	000030	TRAPVEC=	34
168	000034	TKVEC=	60
169	000060	TPVEC=	64
170	000064	PIRQVEC=	240
171	000240		

  

;;	TIME OUT AND OTHER ERRORS
;;	RESERVED AND ILLEGAL INSTRUCTIONS
;;	"T" BIT
;;	TRACE TRAP
;;	BREAKPOINT TRAP (BPT)
;;	INPUT/OUTPUT TRAP (IOT) **SCOPE**
;;	POWER FAIL
;;	EMULATOR TRAP (EMT) **ERROR**
;;	"TRAP" TRAP
;;	TTY KEYBOARD VECTOR
;;	TTY PRINTER VECTOR
;;	PROGRAM INTERRUPT REQUEST VECTOR



```

172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187 000000
188
189
190
191 000174
192 000174 000000
193 000176 000000
194
195 000200 000137 001546
196 000204 000137 001602
197 000210 000137 013600
198 000214 000137 014036
199 000220 000137 001556
200 000224 000137 001570
201 000230 000137 014312
202 000234 000137 014624
203
204
205
206
207 000240
208 000046
209 000046 013320
210 000052
211 000052 000000
212 000240
213 001000
214
215
216
217
218
219 001000
220 000024
221 000024 000200
222 000044
223 000044 001000
224 001000
225

```

```

.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;* SWITCH USE
;* -----
;* 15 HALT ON ERROR
;* 14 LOOP ON TEST
;* 13 INHIBIT ERROR TYPEOUTS
;* 12 LOOP ON CURRENTLY SELECTED DR11-K
;* 11 INHIBIT ITERATIONS
;* 10 OUTPUT TO INPUT WRAPAROUND CABLE NOT CONNECTED
;* 9 LOOP ON ERROR
;* 8 LOOP ON TEST IN SWR<7:0>

.SBTTL TRAP CATCHER
.=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
.=174
DISPREG: .WORD 0 ;; SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;; SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP @#BEGIN ;; JUMP TO STARTING ADDRESS OF PROGRAM
JMP @#BEGIN1 ;; JUMP TO THE RESTART ADDRESS
JMP @#EXTTST ;; JUMP TO THE CONTROL LINES LOOP
JMP @#COULTR ;; JUMP TO THE COULTER FUNCTION LOOP
JMP COULTJ ;; JUMP TO SA WITH THE COULTER JUMPER CONFIG.
JMP H322J ;; JUMP TO SA WITH THE LOC-BOX JUMPER CONFIG.
JMP LOCBOX ;; JUMP TO SA OF LOC-BOX FUNCTION LOOP
JMP XLOIAD ;; JUMP TO SA OF XLO1 ADJUSTMENT LOOP

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

.SBTTL APT PARAMETER BLOCK
;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
.SX=. ;; SAVE CURRENT LOCATION
.=24 ;; SET POWER FAIL TO POINT TO START OF PROGRAM
200 ;; FOR APT START UP
.=44 ;; POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR ;; POINT TO APT HEADER BLOCK
.=.SX ;; RESET LOCATION COUNTER
;*****

```

```

226
227
228
229 001000
230 001000 000000
231 001002 001170
232 001004 000030
233 001006 000010
234 001010 000030
235 001012 000031

```

```

;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.

```

```

$APTHD:
$HIBTS: .WORD 0 ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM: .WORD 30 ;; RUN TIM OF LONGEST TEST
$PASTM: .WORD 10 ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 30 ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
$ETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)

```



236  
237  
238  
239  
240  
241  
242  
243 001100  
244 001100 000000  
245 001102 000  
246 001103 000  
247 001104 000000  
248 001106 000000  
249 001110 000000  
250 001112 000000  
251 001114 000  
252 001115 001  
253 001116 000000  
254 001120 000000  
255 001122 000000  
256 001124 000000  
257 001126 000000  
258 001130 000000  
259 001132 000000  
260 001134 000  
261 001135 000  
262 001136 000000  
263 001140 177570  
264 001142 177570  
265 001144 177560  
266 001146 177562  
267 001150 177564  
268 001152 177566  
269 001154 000  
270 001155 002  
271 001156 012  
272 001157 000  
273 001160 000000  
274 001162 000000  
275 001164 077  
276 001165 015  
277 001166 000012  
278  
279  
280  
281  
282  
283 001170  
284 001170 000000  
285 001172 000000  
286 001174 000000  
287 001176 000000  
288 001200 000000  
289 001202 000000

.SBTTL COMMON TAGS

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

```

.=1100
SCMTAG:                ;; START OF COMMON TAGS
                        .WORD 0                ;; CONTAINS THE TEST NUMBER
STSTNM:                .BYTE 0                ;; CONTAINS ERROR FLAG
SERFLG:                .BYTE 0                ;; CONTAINS SUBTEST ITERATION COUNT
$ICNT:                 .WORD 0                ;; CONTAINS SCOPE LOOP ADDRESS
SLPADR:                .WORD 0                ;; CONTAINS SCOPE RETURN FOR ERRORS
SLPERR:                .WORD 0                ;; CONTAINS TOTAL ERRORS DETECTED
SERTTL:                .WORD 0                ;; CONTAINS ITEM CONTROL BYTE
$ITEMB:                .BYTE 0                ;; CONTAINS MAX. ERRORS PER TEST
SERMAX:                .BYTE 1                ;; CONTAINS PC OF LAST ERROR INSTRUCTION
SERRPC:                .WORD 0                ;; CONTAINS ADDRESS OF 'GOOD' DATA
SGDADR:                .WORD 0                ;; CONTAINS ADDRESS OF 'BAD' DATA
$BDADR:                .WORD 0                ;; CONTAINS 'GOOD' DATA
SGDDAT:                .WORD 0                ;; CONTAINS 'BAD' DATA
SBDDAT:                .WORD 0                ;; RESERVED--NOT TO BE USED
                        .WORD 0
SAUTOB:                .BYTE 0                ;; AUTOMATIC MODE INDICATOR
SINTAG:                .BYTE 0                ;; INTERRUPT MODE INDICATOR
                        .WORD 0
SWR:                   .WORD DSWR            ;; ADDRESS OF SWITCH REGISTER
DISPLAY:               .WORD DDISP          ;; ADDRESS OF DISPLAY REGISTER
$TKs:                  177560                ;; TTY KBD STATUS
$TKB:                  177562                ;; TTY KBD BUFFER
$TPS:                  177564                ;; TTY PRINTER STATUS REG. ADDRESS
$TPB:                  177566                ;; TTY PRINTER BUFFER REG. ADDRESS
$NULL:                 .BYTE 0                ;; CONTAINS NULL CHARACTER FOR FILLS
$FILLS:                .BYTE 2                ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC:                .BYTE 12             ;; INSERT FILL CHARS. AFTER A "LINE FEED"
$TPFLG:                .BYTE 0                ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$TIMES:                0                    ;; MAX. NUMBER OF ITERATIONS
$ESCAPE:               0                    ;; ESCAPE ON ERROR ADDRESS
$QUES:                 .ASCII /?/          ;; QUESTION MARK
$CRLF:                 .ASCII <15>          ;; CARRIAGE RETURN
$LF:                   .ASCII <12>         ;; LINE FEED

```

.SBTTL APT MAILBOX-ETABLE

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

290	001204	000000	\$MSGAD: .WORD	AMSGAD	:: MESSAGE ADDRESS
291	001206	000000	\$MSGLG: .WORD	AMSLG	:: MESSAGE LENGTH
292	001210		\$ETABLE:		:: APT ENVIRONMENT TABLE
293	001210	000	\$ENV: .BYTE	AENV	:: ENVIRONMENT BYTE
294	001211	000	\$ENVM: .BYTE	AENVM	:: ENVIRONMENT MODE BITS
295	001212	000000	\$SWREG: .WORD	ASWREG	:: APT SWITCH REGISTER
296	001214	000000	\$USWR: .WORD	AUSWR	:: USER SWITCHES
297	001216	000000	\$CPUOP: .WORD	ACPUOP	:: CPU TYPE, OPTIONS
298			*		BITS 15-11=CPU TYPE
299			*		11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
300			*		11/70=06, PDQ=07, Q=10
301			*		BIT 10=REAL TIME CLOCK
302			*		BIT 9=FLOATING POINT PROCESSOR
303			*		BIT 8=MEMORY MANAGEMENT
304	001220	000	\$MAMS1: .BYTE	AMAMS1	:: HIGH ADDRESS, M.S. BYTE
305	001221	000	\$MTYP1: .BYTE	AMTYP1	:: MEM. TYPE, BLK#1
306			*		MEM. TYPE BYTE -- (HIGH BYTE)
307			*		900 NSEC CORE=001
308			*		300 NSEC BIPOLAR=002
309			*		500 NSEC MOS=003
310	001222	000000	\$MADR1: .WORD	AMADR1	:: HIGH ADDRESS, BLK#1
311			*		MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
312	001224	000	\$MAMS2: .BYTE	AMAMS2	:: HIGH ADDRESS, M.S. BYTE
313	001225	000	\$MTYP2: .BYTE	AMTYP2	:: MEM. TYPE, BLK#2
314	001226	000000	\$MADR2: .WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
315	001230	000	\$MAMS3: .BYTE	AMAMS3	:: HIGH ADDRESS, M.S. BYTE
316	001231	000	\$MTYP3: .BYTE	AMTYP3	:: MEM. TYPE, BLK#3
317	001232	000000	\$MADR3: .WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
318	001234	000	\$MAMS4: .BYTE	AMAMS4	:: HIGH ADDRESS, M.S. BYTE
319	001235	000	\$MTYP4: .BYTE	AMTYP4	:: MEM. TYPE, BLK#4
320	001236	000000	\$MADR4: .WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
321	001240	100300	\$VECT1: .WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
322	001242	000000	\$VECT2: .WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
323	001244	167770	\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
324	001246	000000	\$DEV: .WORD	ADEV	:: DEVICE MAP
325	001250	000000	\$CDW1: .WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
326	001252		\$TEND:		
327			.MEXIT		



.SBTTL ERROR POINTER TABLE

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

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

001252

\$ERRTB:

;ITEM 1  
EM1 ;:STATUS REGISTER IN ERROR  
DH1 ;:ERRPC DRADD STATUS EXPECTED  
DT1 ;:SERRPC DRADD \$BDDAT \$GDDAT  
DF0

001252 016435  
001254 017055  
001256 017346  
001260 017410

;ITEM 2  
EM2 ;:INPUT REGISTER IN ERROR  
DH2 ;:ERRPC DRADD INPUT EXPECTED  
DT1 ;:SERRPC DRADD \$BDDAT \$GDDAT  
DF0

001262 016466  
001264 017125  
001266 017346  
001270 017410

;ITEM 3  
EM3 ;:OUTPUT REGISTER IN ERROR  
DH3 ;:ERRPC DRADD OUTPUT EXPECTED  
DT1 ;:SERRPC DRADD \$BDDAT \$GDDAT  
DF0

001272 016516  
001274 017173  
001276 017346  
001300 017410

;ITEM 4  
EM4 ;:INPUT FAILED TO INTERRUPT  
DH4 ;:ERRPC DRADD  
DT4 ;:SERRPC DRADD  
DF0

001302 016547  
001304 017241  
001306 017362  
001310 017410

;ITEM 5  
EM5 ;:OUTPUT FAILED TO INTERRUPT  
DH4 ;:ERRPC DRADD  
DT4 ;:SERRPC DRADD  
DF0

001312 016601  
001314 017241  
001316 017362  
001320 017410

;ITEM 6  
EM6 ;:UNEXPECTED INTERRUPT  
DH4 ;:ERRPC DRADD  
DT4 ;:SERRPC DRADD  
DF0

001322 016634  
001324 017241  
001326 017362  
001330 017410

;ITEM 7  
EM7 ;:OPERATOR INTERVENTION ERROR

001332 016661

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

```

382 001334 017241          DH4          ;ERRPC DRADD
383 001336 017362          DT4          ;SERRPC DRADD
384 001340 017410          DFO
385
386          ; ITEM      10
387 001342 016715          EM10         ; INTERRUPT INPUT BIT FAILED TO SET INPUT READY
388 001344 017266          DH10         ; ERRPC DRADD STATUS EXPECTED INPUT BIT
389 001346 017372          DT10         ; SERRPC DRADD $BDDAT $GDDAT BRLEV3
390 001350 017410          DFO
391
392          ; ITEM      11
393 001352 017000          EM11         ; NON-INTERRUPTING INPUT BIT SET INPUT READY
394 001354 017266          DH10         ; ERRPC DRADD STATUS EXPECTED INPUT BIT
395 001356 017372          DT10         ; SERRPC DRADD $BDDAT $GDDAT BRLEV3
396 001360 017410          DFO
397
398          ; ADDRESS OF KMC-11 OF LPA-11      THE ADDR FOR KMADD MAY BE
399          ; .....                          CHANGED BY THE USER TO REFLECT
400          ; .....                          A DIFFERENT KMC-11 ADDR. THE
401          ; .....                          REST OF THE ADDRESSES WILL
402          ; .....                          BE CHANGED BY THE PROGRAM.
403
404
405 001362          LPCI:
406 001362 170460          KMADD: .WORD 170460          ; BASE KMC ADDR. MAY BE PATCHED BY USER.
407
408 001364          LPMR:
409 001364 170461          KMAD1: .WORD 170460+1      ; >DO NOT      <; KMC-CSR ADDR
410 001366          LPCO:
411 001366 170462          KMAD2: .WORD 170460+2      ; >PATCH      <;
412 001370          LPSO:
413 001370 170463          KMAD3: .WORD 170460+3      ; >THIS AREA   <
414 001372          LPADL:
415 001372 170464          KMAD4: .WORD 170460+4      ;
416 001374          LPADH:
417 001374 170465          KMAD5: .WORD 170460+5      ; >DO NOT      <
418 001376          LPMS1:
419 001376 170466          KMAD6: .WORD 170460+6      ; >PATCH      <
420 001400          LPMS2:
421 001400 170467          KMAD7: .WORD 170460+7      ; >THIS AREA   <
422
423 001402 000300          VECTOR: .WORD AVECT1&777   ; BASE VECTOR OF KMC
424 001404 000304          VECTPS: .WORD 4+AVECT1&777 ; VECOTR ADDR.+2
425
426 001406 000004          VERSN: .WORD 4              ; CURRENT VERSION NUMBER OF MICROCODE.
427
428 001410 000000          .DVLS: .WORD 0              ; /DEVICE LIST OF I/O ADDR. DEFINED
429 001412 000020          .BLKW 16.                  ; /BY INIT.
430
431
432 001452 167770          BASEBA: 167770              ; STARTING BASE BUS ADDRESS
433 001454 000300          BASEIV: 300                 ; STARTING BASE INTERRUPT ADDRESS
434 001456 000200          BASEBR: 200
435 001460 000240          CPU: 240                   ; 1 MS. CPU DELAY FACTOR 240 FOR 11/05

```



620 FOR 11/40

```

436
437 001462 000000 NMBEXT: 0 ; ADDITIONAL DR-11-K
438 001464 000000 NBEXT: 0
440 001466 167770 DRADD: 167770 ; CURRENT DR11-K STARTING ADDRESS
441 001470 000300 DRIV: 300 ; CURRENT DR11-K STARTING INTERRUPT VECTOR
442
443
444 001472 167770 LOC1: 167770 ; LOC BOX #1 ADDR
445 001474 167760 LOC2: 167760 ; LOC BOX #2 ADDR
446 001476 167750 LOC3: 167750 ; LOC BOX #3 ADDR
447 001500 167770 GRSTAT: 167770 ; DR STATUS
448 001502 167772 GRDAI: 167772 ; INPUT REG.
449 001504 167774 GRDIO: 167774 ; OUTPUT REG.
450 001506 167775 GRBHIO: 167775 ; OUTPUT REG HIGH BYTE
451 001510 000000 NOTLCH: 0
452 001512 000000 INTBIT: 0
453 001514 000300 GRIVA: 300
454 001516 000302 GRIVSA: 302
455 001520 000304 GRIVB: 304
456 001522 000306 GRIVSB: 306
457 001524 000200 DIOBRL: 200
458 001526 000000 BRLEV1: 0
459 001530 000000 BRLEV2: 0
460 001532 000000 BRLEV3: 0
461 001534 000000 ODDJMP: 0
462 001536 000000 MINSIN: 0
463 001540 000000 TRANST: 0
464 001542 000000 JUMPER: 0 ; 1 IF RUNNING H322 JUMPER CONFIG ; 2 IF COULTER JUMPER CONFIG
465 001544 000000 STMDAT: 0

```

```

466
467
468 ; *****
469 ; ***** DIGITAL I-O LOGIC TEST *****
470 001546 005000 BEGIN: CLR RO ; CLEAR RO
471 001550 005037 001542 CLR JUMPER ; INDICATE NORMAL FACTORY BUILD
472 001554 000414 BR RBEG
473 001556 012737 000002 001542 COULTJ: MOV #2, JUMPER ; INDICATE COULTER JUMPER CONFIG
474 001564 005000 CLR RO
475 001566 000407 BR RBEG
476 001570 012737 000001 001542 H322J: MOV #1, JUMPER ; INDICATE LOC BOX JUMPER CONFIG
477 001576 005000 CLR RO
478 001600 000402 BR RBEG
479 001602 012700 177777 BEGIN1: MOV #-1, RO ; LOAD RO
480 001606 000240 RBEG: NOP
481 .SBTTL INITIALIZE THE COMMON TAGS
482 ; ; CLEAR THE COMMON TAGS ($CMTAG) AREA
483 001610 012706 001100 MOV #CMTAG, R6 ; FIRST LOCATION TO BE CLEARED
484 001614 005026 001140 CLR (R6)+ ; CLEAR MEMORY LOCATION
485 001616 022706 001140 CMP #SWR, R6 ; ; DONE?
486 001622 001374 BNE -6 ; LOOP BACK IF NO
487 001624 012706 001100 MOV #STACK SP ; SETUP THE STACK POINTER
488 ; ; INITIALIZE A FEW VECTORS
489 001630 012737 017430 000020 MOV #SCOPE, @#IOTVEC ; ; IOT VECTOR FOR SCOPE ROUTINE

```

```

490 001636 012737 000340 000022      MOV      #340, @#IOTVEC+2 ;:LEVEL 7
491 001644 012737 017710 000030      MOV      #ERROR, @#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
492 001652 012737 000340 000032      MOV      #340, @#EMTVEC+2 ;:LEVEL 7
493 001660 012737 022440 000034      MOV      #STRAP, @#STRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
494 001666 012737 000340 000036      MOV      #340, @#TRAPVEC+2 ;:LEVEL 7
495 001674 012737 020464 000024      MOV      #SPWRDN, @#PWRVEC ;:POWER FAILURE VECTOR
496 001702 012737 000340 000026      MOV      #340, @#PWRVEC+2 ;:LEVEL 7
497 001710 013737 013266 013260      MOV      SENDCT, SEOPCT ;:SETUP END-OF-PROGRAM COUNTER
498 001716 005037 001160 ;:INITIALIZE NUMBER OF ITERATIONS
499 001722 005037 001162 ;:CLEAR THE ESCAPE ON ERROR ADDRESS
500 001726 112737 000001 001115      MOV      #1, $ERMAX ;:ALLOW ONE ERROR PER TEST
501 001734 012737 001734 001106      MOV      #., $LPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
502 001742 012737 001742 001110      MOV      #., $LPERR ;:SETUP THE ERROR LOOP ADDRESS
503 ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
504 ;:EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
505 001750 013746 000004 ;:SAVE ERROR VECTOR
506 001754 012737 002010 000004      MOV      @#ERRVEC, -(SP) ;:SET UP ERROR VECTOR
507 001762 012737 177570 001140      MOV      #DSWR, SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
508 001770 012737 177570 001142      MOV      #DDISP, DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
509 001776 022777 177777 177134      CMP      #-1, @SWR ;:TRY TO REFERENCE HARDWARE SWR
510 002004 001012 ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
511 ;:AND THE HARDWARE SWR IS NOT = -1
512 002006 000403 ;:BRANCH IF NO TIMEOUT
513 002010 012716 002016 64$: BR      65$ ;:SET UP FOR TRAP RETURN
514 002014 000002 ;:
515 002016 012737 000176 001140 65$: MOV      #SWREG, SWR ;:POINT TO SOFTWARE SWR
516 002024 012737 000174 001142      MOV      #DISPREG, DISPLAY ;:
517 002032 012637 000004 66$: MOV      (SP)+, @#ERRVEC ;:RESTORE ERROR VECTOR
518 ;:
519 002036 005037 001176 ;:CLEAR PASS COUNT
520 002042 132737 000200 001211      BITB     #APTSIZE, $ENVM ;:TEST USER SIZE UNDER APT
521 002050 001403 ;:YES, USE NON-APT SWITCH
522 002052 012737 001212 001140 67$: MOV      #SSWREG, SWR ;:NO, USE APT SWITCH REGISTER
523 ;:
524 ;:
525 ;:THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
526 ;:
527 ;:
528 002060 010046 ;:
529 002062 010146 ;:
530 002064 013700 001362 ;:GET KMC-11 ADDRESS.
531 002070 012701 001364 ;:GET ADDR. OF ADDR. LIST.
532 ;:
533 002074 005200 68$: INC      RO ;:UPDATE ADDR.
534 002076 010021 ;:WRITE ADDR.
535 002100 020127 001402      CMP      R1, #KMAD7+2 ;:DONE ALL ADDRESSES?
536 002104 001373 ;:NO - DO NEXT ADDR.
537 002106 005037 001410      BNE     68$ ;:CLR ADDR. LIST.
538 002112 012601 ;:
539 002114 012600 ;:
540 002116 013737 001244 001452      MOV      $BASE, BASEBA ;:GET APT DEFINED ADDRESS
541 002124 013737 001240 001454      MOV      $VECT1, BASEIV ;:GET VECTOR
542 002132 042737 160000 001454      BIC     #160000, BASEIV ;:
543 002140 113737 001241 001456      MOV      $VECT1+1, BASEBR ;:GET PRIORITY

```



```

544 002146 042737 177437 001456 BIC #177437, BASEBR
545 002154 012737 002222 000004 MOV #15, 2#ERRVEC ;LOAD BUS ERROR
546 002162 013702 001452 MOV BASEBA, R2 ;LOAD STARTING ADDRESS
547 002166 005003 CLR R3 ;CLEAR COUNT
548 002170 010237 001544 2$: MOV R2, $TMDAT ;TEST IF EXISTENT
549
550 ;* MOV 2$TMDAT, $GDDAT ;/READ DEVICE REG $TMDAT, PUT DATA IN $GDDAT.
551 002204 005737 023402 TST $AERR
552 002210 001004 BNE 1$
553 002212 162702 000010 SUB #10, R2 ;EXIST, UPDATE TEST ADDRESS
554 002216 005203 INC R3 ;UPDATE # OF DR11'S
555 002220 000763 BR 2$
556 002222 005703 1$: TST R3 ;TEST IF FIRST DOES EXIST
557 002224 001014 BNE 3$ ;BR
558 002226 032777 020000 176704 BIT #SW13, 2$SWR ;TEST IF INHIBIT TYPEOUT
559 002234 001002 BNE 4$ ;BR IF YES
560 002236 104401 015647 TYPE, BUSTRP ;TELL OPERATOR BASE DR11K DOESN'T EXIST
561 002242 032777 100000 176670 4$: BIT #SW15, 2$SWR ;TEST HALT ON ERROR
562 002250 001401 BEQ 5$ ;BR IF NO HALT
563 002252 000000 HALT ;FIRST DR11-K DOES NOT EXIST
564 ;CHECK THE PROGRAM DEVICE ADDRESS
565 002254 000745 5$: BR 2$ ;TRY AGAIN
566
567 002256 005303 3$: DEC R3 ;ADJUST R3
568 002260 010337 001462 MOV R3, NMBEXT ;SAVE THE NUMBER OF ADDITIONAL DR11-K
569 002264 012737 003022 000004 MOV #VECTRP, 2#ERRVEC ;RESET BUS ERROR
570 002272 012737 000340 000006 MOV #340, 2#ERRVEC+2
571 002300 004737 024354 RBEG1: JSR PC, $RESET
572 002304 005737 000042 TST 2#42 ;TEST IF UNDER A MONITOR
573 002310 001002 BNE 4$ ;BR IF
574 002312 005700 TST R0 ;TEST R0
575 002314 001402 BEQ 2$ ;BR IF CLEARED
576 002316 000137 003332 4$: JMP IOTEST ;JUMP IF SET
577 2$:
578 002322 104401 002330 TYPE , 65$ ;TYPE ASCIZ STRING
579 002326 000426 BR 64$ ;GET OVER THE ASCIZ
580 ;:65$: .ASCIZ <15><12><15><12>/DR11-K DIGITAL INPUT OUTPUT LOGIC TEST/
581 002404 64$:
582 .SBTTL TYPE PROGRAM NAME
583 ;:TYPE THE NAME OF THE PROGRAM IF FIRST PASS
584 002404 005227 177777 INC #-1 ;FIRST TIME?
585 002410 001044 BNE 66$ ;BRANCH IF NO
586 002412 022737 013320 000042 CMP #SENDAD, 2#42 ;ACT-11?
587 002420 001440 BEQ 66$ ;BRANCH IF YES
588 002422 104401 002470 TYPE , 67$ ;TYPE ASCIZ STRING
589 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
590 002426 005737 000042 TST 2#42 ;ARE WE RUNNING UNDER XXDP/ACT?
591 002432 001012 BNE 68$ ;BRANCH IF YES
592 002434 123727 001210 000001 CMPB $ENV, #1 ;ARE WE RUNNING UNDER APT?
593 002442 001406 BEQ 68$ ;BRANCH IF YES
594 002444 023727 001140 000176 CMP SWR, #SWREG ;SOFTWARE SWITCH REG SELECTED?
595 002452 001005 BNE 69$ ;BRANCH IF NO
596 002454 104406 GTSWR ;GET SOFT-SWR SETTINGS
597 002456 000403 BR 69$

```

```

598 002460 112737 000001 001134 68$: MOVB #1,SAUTOB ;;SET AUTO-MODE INDICATOR
599 002466 000415 69$: BR 66$
600 002466 000415 ::67$: .ASCIZ <CRLF><15><12>/MAINDEC-11-DRLPF-A/<15><12><CRLF>
601 002522 66$: MOV NMBEXT,-(SP)
602 002522 013746 001462 TYPOS
603 002522 104403 WORD 2
604 002526 104403 TYPE 71$ ::TYPE ASCIZ STRING
605 002530 000002 002540 BR 70$ ::GET OVER THE ASCIZ
606 002532 104401 70$: .ASCIZ /(8) ADDITIONAL DR11-K'S CONNECTED/<15><12>
607 002536 000422 ::71$:
608 002604 70$: CMP #1,JUMPER ;TEST IF LOC-BOX CONFIG.
609 002604 022737 000001 001542 BNE 1$ ;BR IF NOT
610 002612 001013 CLR NOTLCH ;LOAD LOC-BOX JUMPER CONFIG
611 002614 005037 001510 MOV #-1,INTBIT
612 002620 012737 177777 001512 CLR MINSIN
613 002626 005037 001536 CLR TRNST
614 002632 005037 001540 JMP IOTEST ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
615 002632 005037 001540 1$: CMP #2,JUMPER ;TEST IF COULTER CONFIG.
616 002636 000137 003332 BNE 3$ ;BR IF NOT
617 002642 022737 000002 001542 1$: MOV #-1,NOTLCH ;LOAD COULTER JUMPER CONFIG.
618 002650 001015 MOV #170000,INTBIT
619 002652 012737 177777 001510 CLR MINSIN
620 002660 012737 170000 001512 MOV #170000,TRNST
621 002666 005037 001536 JMP IOTEST ;RUN THE LOGIC TEST WITH COULTER JUMPERS
622 002672 012737 170000 001540 JSR RS,TALK ;TALK TO THE ANIMALS
623 002700 000137 003332 3$: SWNLB
624 002704 004537 003246 JSR RS,TALK ;ABOUT NON LATCH INPUT BITS
625 002710 015740 NOTLCH ;TALK TO THE ANIMAL AGAIN
626 002712 001510 JSR RS,TALK
627 002714 004537 003246 SWINTB
628 002720 016036 INTBIT ;ABOUT THE INTERRUPTING BITS
629 002722 001512 JSR RS,TALK ;ISN'T THIS BOARING
630 002724 004537 003246 JSR SWPOSB
631 002730 016134 MINSIN ;AGAIN-ABOUT THE MINUS INPUT BITS
632 002732 001536 JSR RS,TALK ;HO-HUM
633 002734 004537 003246 JSR SWTRAB
634 002740 016230 TRNST ;AGAIN-ABOUT THE TRANSITION BITS
635 002742 001540
636
637 .SBTTL DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
638 002744 012737 010000 001124 MOV #BIT12,$GDDAT ;LOAD TEST BIT
639 002752 033737 001124 001536 10$: BIT $GDDAT,MINSIN ;TEST IF NEG INPUT BIT
640 002760 001407 BEQ 11$ ;BR IF NOT
641 002762 033737 001124 001540 BIT $GDDAT,TRNST ;TEST IF TRANS. INPUT
642 002770 001403 BEQ 11$ ;BR IF NOT
643 002772 104007 ERROR 7 ;LOGICAL OPERATOR ERROR
644 ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
645 002774 000137 001546 JMP BEGIN
646
647 003000 006137 001124 11$: ROL $GDDAT ;MOVE LEFT
648 003004 103362 BCC 10$ ;BR UNTIL DONE
649
650 003006 004537 003246 JSR RS,TALK ;ASK THE OPERATOR
651 003012 016326 SWDPOB

```



```

652 003014 001532          BRLEV3          ;ABOUT PROGRAM OPTIONS
653 003016 000137 003332  JMP          IOTEST
654
655          ; INTERRUPT TO UNEXPECTED VECTOR
656 003022 021627 001000  VECTRP: CMP      (SP),#1000          ;TEST IF IN PROGRAM CODE
657 003026 103402          BLO          1$          ;BR IF NOT
658 003030 000000 70$:  HALT          ;FATAL BUS TRAP IN PROGRAM AREA
659 003032 000776          BR          70$
660 003034 021627 000200  1$:  CMP      (SP),#200          ;TEST IF IN SACRED VECTOR AREA
661 003040 101002          BHI          2$          ;BR IF NOT
662 003042 000000 71$:  HALT          ;FATAL VECTOR TRAP
663 003044 000776          BR          71$
664 003046 012637 003244  2$:  MOV      (SP)+,72$          ;GET ADDR THE TRAP OCCURRED TO
665 003052 162737 000004 003244  SUB      #4,72$          ;MAKE REAL ADDRESS
666 003060 032777 020000 176052  BIT      #SW13,SWR          ;TEST IF TYPE ERROR INHIBIT
667 003066 001050          BNE          3$          ;BR IF INHIBIT
668 003070 104401 003076          TYPE      ,65$          ;;TYPE ASCIZ STRING
669 003074 000417          BR          64$          ;;GET OVER THE ASCIZ
670
671 003134          ;;65$: .ASCIZ <15><12>/DR11K INTERRUPTED TO LOC. /
672 003134 013746 003244  64$:  MOV      72$,-(SP)          ;LOAD VALUE TO BE TYPED
673 003140 104402          TYP0C
674 003142 104401 003150  TYPE      ,67$          ;;TYPE ASCIZ STRING
675 003146 000420          BR          66$          ;;GET OVER THE ASCIZ
676
677 003210          ;;67$: .ASCIZ / AND WILL NOW USE THAT VECTOR/<15><12>
678 003210 042737 000007 003244  66$:  BIC      #7,72$          ;MASK OFF LOWER 3 BITS
679 003216 032777 100000 175714  BIT      #SW15,SWR          ;TEST IF HALT ON ERROR
680 003224 001401          BEQ          4$          ;BR IF NOT
681 003226 000000          HALT          ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
682 003230 022626 4$:  CMP      (SP)+,(SP)+          ;CLEAN THE STACK
683 003232 013737 003244 001470  MOV      72$,DRIV          ;LOAD NEW VECTOR ADDRESS
684 003240 000137 003360          JMP      RBEG2          ;TEST THE DR11K AT THE NEW VECTOR
685 003244 000000 72$:  C
686
687          ;OPERATOR QUESTIONS AND ANSWER ROUTINE
688
689 003246 012537 003260  TALK:  MOV      (R5)+,10$          ;GET ASCII POINTER
690 003252 012537 003324  MOV      (R5)+,11$          ;GET POINTER TO DATA FROM OPERATOR
691 003256 104401          TYPE
692 003260 015740 10$:  SWNLB          ;TELL OPERATOR TO LOAD SWITCHES
693 003262 023727 001140 000176  CMP      SWR,#SWREG          ;TEST IF SWITCH REG. EXISTS
694 003270 001006          BNE          1$          ;BR IF NO
695 003272 104401 022147  TYPE      ,MSWR          ;TYPE "SWR ="
696 003276 104412          RDOCT          ;READ OCTAL
697 003300 012677 175634  MOV      (SP)+,SWR          ;SAVE THE SWITCHES
698 003304 000403          BR          2$
699 003306 104401 1$:  TYPE
700 003310 016413          DEPCNT          ;TELL OPERATOR TO DEPRESS CONT
701 003312 000000          HALT          ;WAIT FOR OPERATOR
702 003314 017777 175620 000002  2$:  MOV      SWR,SWR,SWR          ;LOAD SWITCH VALUE
703 003322 000205          RTS          ;EXIT
704 003324 001510 11$:  NOTLCH          ;POINTER TO DATA TO BE LOADED
705
    
```

```

706 ;UNEXPECTED INTERRUPT
707 003326 104006 UNEXPT: ERROR 6 ;UNEXPECTED INTERRUPT DURING A SUB-TEST
708 003330 000002 RTI ;EXIT
709
710 003332 004737 024354 IOTEST: JSR PC,$RESET
711 003336 013737 001452 001466 MOV BASEBA,DRADD ;LOAD INITIAL STARTING ADDRESS
712 003344 013737 001454 001470 MOV BASEIV,DRIV ;LOAD INITIAL STARTING VECTOR
713 003352 013737 001462 001464 MOV NMBEXT,NBEXT ;RELOAD # AVAILABLE
714 003360 004737 024354 RBEG2: JSR PC,$RESET
715 003364 012701 000240 MOV #240,R1 ;LOAD R1
716 003370 012702 000242 MOV #242,R2
717 003374 010221 004700 5S: MOV R2,(R1)+ ;SAVE THE ADDRESS
718 003376 012721 004700 MOV #4700,(R1)+ ;LOAD "JSR PC,RO"
719 003402 022222 CMP (R2)+(R2)+ ;BUMP R2
720 003404 020227 001076 CMP R2,#SCMTAG-2 ;TEST FOR LAST
721 003410 001371 BNE 5S ;BR UNTIL DONE
722
723 003412 004737 024354 IOTST1: JSR PC,$RESET
724 003416 004737 003424 JSR PC,SETADD ;SET UP BUS ADDRESS AND VECTOR
725 003422 000453 BR IOTTS1
726 003424 013737 001466 001500 SETADD: MOV DRADD,GRSTAT ;LOAD 1ST ADDRESS
727 003432 013737 001466 001502 MOV DRADD,GRDAI ;LOAD 2ND ADDRESS
728 003440 062737 000002 001502 ADD #2,GRDAI
729 003446 013737 001466 001504 MOV DRADD,GRDIO ;LOAD 3RD ADDRESS
730 003454 062737 000004 001504 ADD #4,GRDIO
731 003462 013737 001466 001506 MOV DRADD,GRBHIO ;LOAD 4TH ADDRESS
732 003470 062737 000005 001506 ADD #5,GRBHIO
733 003476 013737 001470 001514 MOV DRIV,GRIVA ;LOAD FIRST VECTOR
734 003504 013737 001470 001516 MOV DRIV,GRIVSA
735 003512 062737 000002 001516 ADD #2,GRIVSA
736 003520 013737 001470 001520 MOV DRIV,GRIVB ;LOAD 2ND VECTOR
737 003526 062737 000004 001520 ADD #4,GRIVB
738 003534 013737 001470 001522 MOV DRIV,GRIVSB
739 003542 062737 000006 001522 ADD #6,GRIVSB
740 003550 000207 RTS PC
741 003552 013737 001456 001524 IOTTS1: MOV BASEBR,DI0BRL ;LOAD BR LEVEL
742 003560 005037 001534 CLR ODDJMP
743 003564 053737 001536 001534 BIS MINSIN,ODDJMP ;SET MINUS INPUT BITS
744 003572 053737 001540 001534 BIS TRANST,ODDJMP ;SET TRANSITION INPUT BITS
745 003600 012777 003326 175706 MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
746 003606 012777 000340 175702 MOV #340,@GRIVSA
747 003614 012777 003326 175676 MOV #UNEXPT,@GRIVB ;RESET OUTPUT VECTOR
748 003622 012777 000340 175672 MOV #340,@GRIVSB
749
750 ;*****
751 ;#TEST 1 TEST FOR NO BUS ERRORS
752 ;*****
752 003630 000004 TST1: SCOPE
753 003632 012737 000001 001174 MOV #1,$TESTN
754 003640 012737 000001 001102 MOV #1,$STNM
755 003646 012737 000000 001544 MOV #0,$TMDAT
756
757 ;* MOV $TMDAT,@GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
758 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
759

```



```

760
761      ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
762
763      ;*****
764      ;*TEST 2      TEST THAT OUTPUT REG. CAN HOLD #-1
765      ;*****
766      003704  000004      TST2:  SCOPE
767      003706  012737  177777  001124      MOV      #-1, $GDDAT      ;LOAD EXPECTED
768      003714  012737  177777  001544      MOV      #-1, STMDAT      ;ALL ONES TO REGISTER
769
770      ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
771
772      ;*      MOV      @GRDIO, $BDDAT      ;/READ DEVICE REG GRDIO, PUT DATA IN $BDDAT.
773      003742  023737  001124  001126      CMP      $GDDAT, $BDDAT      ;COMPARE
774      003750  001401      BEQ      TST3      ;;BR IF EQUAL
775      003752  104003      ERROR    3      ;REG WILL NOT HOLD ONES
776      ;*****
777      ;*TEST 3      TEST THAT RESET CLEARS OUTPUT REG.
778      ;*****
779      003754  000004      TST3:  SCOPE
780      003756  012737  000002  001160      MOV      #2, $TIMES      ;;DO 2 ITERATIONS
781      003764  005037  001124      CLR      $GDDAT
782      003770  012737  177777  001544      MOV      #-1, STMDAT
783
784      ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
785      004006  004737  024354      JSR      PC, $RESET      ;SET DATA TO ALL ONES
786
787      ;*      MOV      @GRDIO, $BDDAT      ;/READ DEVICE REG GRDIO, PUT DATA IN $BDDAT.
788      004022  005737  001126      TST      $BDDAT
789      004026  001401      BEQ      TST4      ;;BR IF EQUAL
790      004030  104003      ERROR    3      ;REG FAILED TO CLEAR
791
792      ;*****
793      ;*TEST 4      TEST THAT OUTPUT REG. CAN HOLD #52525
794      ;*****
795      004032  000004      TST4:  SCOPE
796      004034  012737  052525  001124      MOV      #52525, $GDDAT      ;LOAD EXPECTED VALUE
797      004042  012737  052525  001544      MOV      #52525, STMDAT
798
799      ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
800
801      ;*      MOV      @GRDIO, $BDDAT      ;/READ DEVICE REG GRDIO, PUT DATA IN $BDDAT.
802      004070  023737  001124  001126      CMP      $GDDAT, $BDDAT      ;COMPARE
803      004076  001401      BEQ      TST5      ;;BR IF EQUAL
804      004100  104003      ERROR    3      ;DATA NOT=52525
805
806      ;*****
807      ;*TEST 5      TEST THAT OUTPUT REG. CAN HOLD #125252
808      ;*****
809      004102  000004      TST5:  SCOPE
810      004104  012737  125252  001124      MOV      #125252, $GDDAT      ;LOAD EXPECTED VALUE
811      004112  012737  125252  001544      MOV      #125252, STMDAT
812
813      ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO

```

```

814
815
816 004140 023737 001124 001126 ;* MOV 2GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
817 004146 001401 ;* CMP $GDDAT,$BDDAT ;COMPARE
818 004150 104003 ;* BEQ TST6 ;;BR IF EQUAL
819 ;* ERROR 3 ;DATA NOT=125252
820 ;*****
821 ;*TEST 6 TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
822 ;*****
823 004152 000004 TST6: SCOPE
824 004154 012737 000001 001160 MOV #1,$TIMES ;;DO 1 ITERATION
825 004162 012737 004174 001110 MOV #2,$SLPERR ;LOAD SCOPE ERROR RETURN
826 004170 005037 001124 CLR $GDDAT ;CLEAR PATTERN
827 004174 2S:
828 ;* MOV $GDDAT,2GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
829 ;*
830 ;* MOV 2GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
831 ;* CMP $GDDAT,$BDDAT ;COMPARE
832 004214 023737 001124 001126 ;* BEQ 1S ;;BR IF EQUAL
833 004222 001401 ;* ERROR 3 ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
834 004224 104003 ;* INC $GDDAT ;UPDATE THE PATTERN
835 004226 005237 001124 1S: BNE 2S ;TRY AGAIN
836 004232 001360 ;*****
837 ;*TEST 7 FLOAT A 1 ACROSS THE OUTPUT REGISTER
838 ;*****
839 004234 000004 TST7: SCOPE
840 004236 012737 004252 001110 MOV #1,$SLPERR ;LOAD SCOPE ERROR RETURN
841 004244 012737 000001 001124 MOV #BIT0,$GDDAT ;LOAD EXPECTED VALUE
842 ;* MOV #0,$TMDAT ;CLEAR OUTPUT
843 ;* MOV $TMDAT,2GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
844 004252 012737 000000 001544 1S: BIS $GDDAT,$TMDAT ;SET THAT BIT
845 ;* MOV $TMDAT,2GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
846 004270 053737 001124 001544 ;*
847 ;* MOV 2GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
848 ;* CMP $GDDAT,$BDDAT ;TEST RESULTS
849 ;* BEQ 2S ;BR IF EQUAL ?
850 ;* ERROR 3
851 ;* ASL $GDDAT ;SHIFT EXPECTED DATA
852 004316 023737 001124 001126 ;* BNE 1S ;BR UNTIL DONE
853 004324 001401 ;*****
854 004326 104003 ;*TEST 10 FLOAT A 0 ACROSS THE OUTPUT REGISTER
855 ;*****
856 004330 006337 001124 2S: ASL $GDDAT ;SHIFT EXPECTED DATA
857 004334 001346 ;* BNE 1S ;BR UNTIL DONE
858 ;*****
859 ;*TEST 10 FLOAT A 0 ACROSS THE OUTPUT REGISTER
860 ;*****
861 004336 000004 TST10: SCOPE
862 004340 012737 004354 001110 MOV #1,$SLPERR ;LOAD SCOPE ERROR RETURN
863 004346 012737 000001 001124 MOV #BIT0,$GDDAT ;LOAD EXPECTED VALUE
864 ;* MOV #-1,$TMDAT ;LOAD OUTPUT TO A ONE
865 004354 012737 177777 001544 1S:
866 ;* MOV $TMDAT,2GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
867

```



```

868 004372 043737 001124 001544      BIC      $GDDAT,$STMDAT      ;CLEAR A BIT
869
870      ;*      MOV      $STMDAT,$GRDIO      ;/ PUT DATA FROM $STMDAT TO DEVICE REG GRDIO
871
872      ;*      MOV      $GRDIO,$SBDDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $SBDDAT.
873 004420 005137 001126      COM      $SBDDAT      ;COMPLEMENT IT
874 004424 023737 001124 001126      CMP      $GDDAT,$SBDDAT      ;EQUAL ?
875 004432 001401      BEQ      2$      ;BR IF EQUAL
876 004434 104003      ERROR      3
877
878 004436 006337 001124      2$:      ASL      $GDDAT      ;SHIFT LEFT
879 004442 001344      BNE      1$      ;BRANCH UNTIL DONE
880
881      ;:*****
882      ;*TEST 11      TEST FOR SLOW OUTPUT GATES WITH #125252
883      ;:*****
884 004444 000004      TST11:  SCOPE
885 004446 012737 125252 001124      MOV      #125252,$GDDAT      ;LOAD EXPECTED VALUE
886
887      ;*      MOV      $GDDAT,$GRDIO      ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
888
889      ;*      MOV      $GRDIO,$STMDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $STMDAT.
890 004474 005137 001544      COM      $STMDAT
891
892      ;*      MOV      $STMDAT,$GRDIO      ;/ PUT DATA FROM $STMDAT TO DEVICE REG GRDIO
893
894      ;*      MOV      $GRDIO,$STMDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $STMDAT.
895 004520 005137 001544      COM      $STMDAT
896
897      ;*      MOV      $STMDAT,$GRDIO      ;/ PUT DATA FROM $STMDAT TO DEVICE REG GRDIO
898
899      ;*      MOV      $GRDIO,$STMDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $STMDAT.
900 004544 005137 001544      COM      $STMDAT
901
902      ;*      MOV      $STMDAT,$GRDIO      ;/ PUT DATA FROM $STMDAT TO DEVICE REG GRDIO
903
904      ;*      MOV      $GRDIO,$STMDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $STMDAT.
905 004570 005137 001544      COM      $STMDAT
906
907      ;*      MOV      $STMDAT,$GRDIO      ;/ PUT DATA FROM $STMDAT TO DEVICE REG GRDIO
908
909      ;*      MOV      $GRDIO,$STMDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $STMDAT.
910 004614 005137 001544      COM      $STMDAT
911
912      ;*      MOV      $STMDAT,$GRDIO      ;/ PUT DATA FROM $STMDAT TO DEVICE REG GRDIO
913
914      ;*      MOV      $GRDIO,$STMDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $STMDAT.
915 004640 005137 001544      COM      $STMDAT
916
917      ;*      MOV      $STMDAT,$GRDIO      ;/ PUT DATA FROM $STMDAT TO DEVICE REG GRDIO
918
919      ;*      MOV      $GRDIO,$STMDAT      ;/READ DEVICE REG GRDIO,PUT DATA IN $STMDAT.
920 004664 005137 001544      COM      $STMDAT
921

```

```

922          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
923
924          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
925 004710    005137    001544
926          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
927
928          ;*      MOV      @GRDIO, $BDDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN $BDDAT.
929          ;*      CMP      $GDDAT, $BDDAT      ; TEST REGISTER
930 004734    023737    001124    001126          ;*      BEQ      TST12      ;; BR IF CORRECT
931 004742    001401
932 004744    104003          ;*      ERROR      3
933          ;*****
934          ;*TEST 12      TEST FOR SLOW OUTPUT GATES WITH #52525
935          ;*****
936 004746    000004          †ST12: SCOPE
937 004750    012737    052525    001124          MOV      #52525, $GDDAT      ;LOAD EXPECTED VALUE
938
939          ;*      MOV      $GDDAT, @GRDIO      ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
940
941          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
942 004776    005137    001544
943
944          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
945
946          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
947 005022    005137    001544
948
949          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
950
951          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
952 005046    005137    001544
953
954          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
955
956          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
957 005072    005137    001544
958
959          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
960
961          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
962 005116    005137    001544
963
964          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
965
966          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
967 005142    005137    001544
968
969          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
970
971          ;*      MOV      @GRDIO, STMDAT      ;/ READ DEVICE REG GRDIO, PUT DATA IN STMDAT.
972 005166    005137    001544
973
974          ;*      MOV      STMDAT, @GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
975

```



```

976          ;*      MOV      @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
977 005212 005137 001544          ;*      COM      $TMDAT
978          ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
979          ;*      MOV      @GRDIO,$BDDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
980          ;*      CMP      $GDDAT,$BDDAT  ;TEST PATTERN
981 005236 023737 001124 001126          ;*      BEQ      TST13      ;;BR IF EQUAL
982 005244 001401          ;*      ERROR 3      ;OUTPUT REGISTER IN ERROR
983 005246 104003
984          ;*****
985          ;*TEST 13      TEST OUTPUT DATA ACCEPT FLAG
986          ;*****
987          ;*ST13: SCOPE
988          ;*      MOV      #0,$TMDAT
989 005250 000004          ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
990 005252 012737 000000 001544          ;*      MOV      #-1,$TMDAT
991          ;*      MOV      $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
992 005270 012737 177777 001544          ;*      MOV      #BIT15,$GDDAT  ;LOAD EXPECTED
993          ;*      MOV      $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
994          ;*      MOV      @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
995 005306 012737 100000 001124          ;*      BIT      $GDDAT,$BDDAT
996          ;*      BNE      TST14      ;;BR IF SET
997          ;*      ERROR 1      ;ERROR, BIT 15 FAILED TO SET
998          ;*****
999          ;*TEST 14      TEST OUTPUT INTERRUPT ENABLE
1000          ;*****
1001          ;*ST14: SCOPE
1002          ;*      MOV      #0,$TMDAT      ;CLEAR STATUS
1003          ;*      MOV      $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1004          ;*      MOV      #BIT14,$GDDAT  ;LOAD EXPECTED
1005          ;*      MOV      $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
1006          ;*      MOV      @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
1007 005346 000004          ;*      BIT      $GDDAT,$BDDAT
1008 005350 012737 000000 001544          ;*      BNE      TST15      ;;BR IF SET
1009          ;*      ERROR 1      ;ERROR BIT 14 FAILED TO SET
1010          ;*****
1011          ;*TEST 15      TEST INPUT DATA READY FLAG
1012          ;*****
1013          ;*ST15: SCOPE
1014          ;*      MOV      #BIT7,$GDDAT  ;LOAD EXPECTED
1015          ;*      MOV      $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
1016          ;*      MOV      @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
1017 005414 033737 001124 001126
1018 005422 001001
1019 005424 104001
1020          ;*
1021          ;*
1022          ;*
1023          ;*
1024 005426 000004
1025 005430 012737 000200 001124
1026          ;*
1027          ;*
1028          ;*
1029          ;*

```

```

1030 005456 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1031 005464 001401                      BEQ      TST16              ;;BR IF EQUAL
1032 005466 104001                      ERROR    1                  ;ERROR, BIT 7 FAILED TO SET
1033
1034 ;:*****
1035 ;:TEST 16      TEST INPUT INTERRUPT ENABLE
1036 ;:*****
1037 005470 000004      †TST16: SCOPE
1038 005472 012737 000000 001544      MOV      #0,$TMDAT          ;CLEAR STATUS
1039
1040 ;*      MOV      $TMDAT,$GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1041 005510 012737 000100 001124      MOV      #BIT6,$GDDAT      ;LOAD EXPECTED
1042
1043 ;*      MOV      $GDDAT,$GRSTAT ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
1044
1045 ;*      MOV      $GRSTAT,$BDDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
1046 005536 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1047 005544 001401                      BEQ      TST17              ;;BR IF EQUAL
1048 005546 104001                      ERROR    1                  ;ERROR, BIT 6 FAILED TO SET
1049
1050
1051 ;:*****
1052 ;:TEST 17      TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
1053 ;:*****
1054 ;:*****
1055 005550 000004      †TST17: SCOPE
1056 005552 032777 002000 173360      BIT      #BIT10,$SWR        ;TEST SWITCH BIT
1057 005560 001405                      BEQ      1$                 ;BRANCH IF DOWN
1058 005562 112737 000036 001102      MOV      #36,$STSINM        ;LOAD NEW TEST NUMBER
1059 005570 000137 012250                      JMP      DRT21              ;BYPASS SOME TEST USING THE EXTERNAL CABLE
1060
1061 005574 012737 000000 001544      1$:      MOV      #0,$TMDAT          ;CLEAR OUTPUT REGISTER
1062
1063 ;*      MOV      $TMDAT,$GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1064 005612 012737 177777 001544      MOV      #-1,$TMDAT
1065
1066 ;*      MOV      $TMDAT,$GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1067 005630 005037 001124 001544      CLR      $GDDAT            ;CLEAR EXPECTED
1068 005634 012737 000000 001544      MOV      #0,$TMDAT          ;LOAD THE OUTPUT
1069
1070 ;*      MOV      $TMDAT,$GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1071
1072 ;*      MOV      $GRDAI,$BDDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1073 005662 043737 001534 001126      BIC      ODDJMP,$BDDAT      ;MASK
1074 005670 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1075 005676 001401                      BEQ      TST20              ;;BR IF EQUAL
1076 005700 104002                      ERROR    2                  ;ERROR, INPUT DID NOT EQUAL THE OUTPUT REG.
1077
1078 ;:*****
1079 ;:TEST 20      TEST INPUT WITH #-1
1080 ;:*****
1081 005702 000004      †TST20: SCOPE
1082 005704 012737 000000 001544      MOV      #0,$TMDAT          ;CLEAR OUTPUT REGISTER
1083

```



```

1084      ;*      MOV      STMDAT,GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1085      005722  012737  177777  001544      MOV      #-1,STMDAT
1086
1087      ;*      MOV      STMDAT,GRDAI      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1088      005740  012737  177777  001124      MOV      #-1,SGDDAT      ;LOAD EXPECTED
1089      005746  043737  001534  001124      BIC      ODDJMP,SGDDAT      ;MASK
1090
1091      ;*      MOV      SGDDAT,GRDIO      ;/ PUT DATA FROM SGDDAT TO DEVICE REG GRDIO
1092
1093      ;*      MOV      GRDAI,$BDDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1094      005774  043737  001534  001126      BIC      ODDJMP,$BDDAT      ;MASK
1095      006002  023737  001124  001126      CMP      SGDDAT,$BDDAT      ;COMPARE
1096      006010  001401      BEQ      TST21      ;;BR IF EQUAL
1097      006012  104002      ERROR    2      ;ERROR, INPUT DID NOT EQUAL THE OUTPUT
1098
1099      ;*****
1100      ;*TEST 21      TEST INPUT WITH #52525
1101      ;*****
1102      †ST21:  SCOPE
1103      006014  000004      MOV      #0,STMDAT      ;CLEAR OUTPUT REGISTER
1104      006016  012737  000000  001544      MOV
1105      ;*      MOV      STMDAT,GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1106      006034  012737  177777  001544      MOV      #-1,STMDAT
1107
1108      ;*      MOV      STMDAT,GRDAI      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1109      006052  012737  052525  001124      MOV      #52525,SGDDAT      ;LOAD EXPECTED
1110      006060  043737  001534  001124      BIC      ODDJMP,SGDDAT      ;MASK
1111
1112      ;*      MOV      SGDDAT,GRDIO      ;/ PUT DATA FROM SGDDAT TO DEVICE REG GRDIO
1113
1114      ;*      MOV      GRDAI,$BDDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1115      006106  043737  001534  001126      BIC      ODDJMP,$BDDAT      ;MASK
1116      006114  023737  001124  001126      CMP      SGDDAT,$BDDAT      ;COMPARE
1117      006122  001401      BEQ      TST22      ;;BR IF EQUAL
1118      006124  104002      ERROR    2      ;ERROR, INPUT DID NOT EQUAL OUTPUT
1119
1120      ;*****
1121      ;*TEST 22      TEST INPUT WITH #125252
1122      ;*****
1123      †ST22:  SCOPE
1124      006126  000004      MOV      #0,STMDAT      ;CLEAR OUTPUT REGISTER
1125      006130  012737  000000  001544      MOV
1126      ;*      MOV      STMDAT,GRDIO      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1127      006146  012737  177777  001544      MOV      #-1,STMDAT
1128
1129      ;*      MOV      STMDAT,GRDAI      ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1130      006164  012737  125252  001124      MOV      #125252,SGDDAT      ;LOAD EXPECTED
1131      006172  043737  001534  001124      BIC      ODDJMP,SGDDAT      ;MASK
1132
1133      ;*      MOV      SGDDAT,GRDIO      ;/ PUT DATA FROM SGDDAT TO DEVICE REG GRDIO
1134
1135      ;*      MOV      GRDAI,$BDDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1136      006220  043737  001534  001126      BIC      ODDJMP,$BDDAT      ;MASK
1137      006226  023737  001124  001126      CMP      SGDDAT,$BDDAT      ;COMPARE

```

```

1138 006234 001401 BEQ TST23 ;;BR IF EQUAL
1139 006236 104002 ERROR 2 ;ERROR, INPUT DID NOT EQUAL OUTPUT
1140 *****
1141 :;TEST 23 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
1142 *****
1143 006240 000004 TST23: SCOPE
1144 006242 012737 000001 001160 MOV #1,STIMES ;;DO 1 ITERATION
1145 006250 005737 001534 TST ODDJMP ;TEST IF ANY ODD JUMPERS
1146 006254 001514 BEQ TST24 ;;BR IF NONE
1147 006256 012737 010000 001124 MOV #BIT12,$GDDAT ;LOAD TEST BIT
1148 006264 033737 001534 001124 1$: BIT ODDJMP,$GDDAT ;TEST IF ODD JUMPER BIT
1149 006272 001502 BEQ 2$ ;BR IF NOT
1150 006274 033737 001510 001124 BIT NOTLCH,$GDDAT ;TEST IF LATCHING INPUT BIT
1151 006302 001076 BNE 2$ ;BR IF NOT
1152 *****
1153 ;* MOV $GDDAT,$GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1154 006314 012737 177777 001544 MOV #-1,$TMDAT ;CLEAR INPUT
1155 *****
1156 ;* MOV $TMDAT,$GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1157 *****
1158 ;* MOV $GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1159 006342 043737 001124 001544 BIC $GDDAT,$TMDAT
1160 *****
1161 ;* MOV $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1162 *****
1163 ;* MOV $GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1164 006370 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1165 006376 001401 BEQ 3$ ;BR IF EQUAL
1166 006400 104002 ERROR 2 ;ERROR, NEG. INPUT OR NEG. TRANSITION
1167 ; INPUT BIT FAILED TO SET INPUT REGISTER
1168 *****
1169 006402 033737 001124 001536 3$: BIT $GDDAT,MINSIN ;TEST IF NEG. INPUT BIT
1170 006410 001033 BNE 2$ ;BR IF
1171 006412 012737 177777 001544 MOV #-1,$TMDAT ;CLEAR INPUT
1172 *****
1173 ;* MOV $TMDAT,$GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1174 *****
1175 ;* MOV $GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1176 006440 053737 001124 001544 BIS $GDDAT,$TMDAT
1177 *****
1178 ;* MOV $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1179 *****
1180 ;* MOV $GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1181 006466 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1182 006474 001401 BEQ 2$ ;BR IF EQUAL
1183 006476 104002 ERROR 2 ;ERROR, POSITIVE INPUT TRANSITION
1184 ; LOGIC FAILED TO SET INPUT REGISTER BIT
1185 *****
1186 006500 006137 001124 2$: ROL $GDDAT ;CHANGE DATA PATTERN
1187 006504 103267 BCC 1$ ;BR IF MORE DATA
1188 *****
1189 :;TEST 24 FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
1190 *****
1191 006506 000004 TST24: SCOPE

```



```

1192 006510 012737 006536 001110      MOV    #25,$LPERR        ;LOAD ERROR SCOPE RETURN
1193                                     ;
1194 006516 012737 000001 001530      MOV    #BIT0,BRLEV2     ;LOAD EXPECTED
1195 006524 013737 001510 001532      MOV    NOTLCH,BRLEV3    ;GET NON-LATCH
1196 006532 005137 001532                COM    BRLEV3           ;COMPLEMENT
1197 006536 013737 001530 001124 2$:  MOV    BRLEV2,$GDDAT    ;LOAD GOOD
1198 006544 033737 001124 001534      BIT    $GDDAT,ODDJMP    ;TEST IF ODD JUMPER
1199 006552 001055                BNE    1$              ;BYPASS IF ODD JUMPER
1200 006554 033737 001124 001510      BIT    $GDDAT,NOTLCH    ;TEST FOR NON-LATCH
1201 006562 001451                BEQ    1$              ;BR IF LATCHING
1202                                     ;
1203                                     ;
1204                                     ;*   MOV    $GDDAT,$GRDIO  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1205                                     ;
1206                                     ;*   MOV    $GRDAI,$BDDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1207 006604 043737 001532 001126      BIC    BRLEV3,$BDDAT    ;MASK TO LATCH BITS
1208 006612 023737 001124 001126      CMP    $GDDAT,$BDDAT    ;COMPARE
1209 006620 001401                BEQ    3$              ;;BR IF EQUAL
1210 006622 104002                ERROR  2              ;INPUT REGISTER IN ERROR
1211                                     ;
1212                                     ;
1213                                     ;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
1214                                     ;
1215 006624                3$:
1216                                     ;
1217                                     ;*   MOV    $GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1218 006634 043737 001124 001544      BIC    $GDDAT,$TMDAT
1219                                     ;
1220                                     ;*   MOV    $TMDAT,$GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1221                                     ;
1222                                     ;*   MOV    $GRDAI,$BDDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1223 006662 043737 001532 001126      BIC    BRLEV3,$BDDAT    ;MASK TO LATCH BITS
1224 006670 005037 001124                CLR    $GDDAT           ;CLEAR EXPECTED
1225 006674 033737 001530 001126      BIT    BRLEV2,$BDDAT    ;TEST FOR BIT
1226 006702 001401                BEQ    1$              ;;BR IF CLEARED
1227 006704 104002                ERROR  2              ;INPUT BIT LATCHED IN ERROR
1228                                     ;
1229                                     ;
1230 006706 006337 001530                1$:   ASL    BRLEV2         ;CHANGE PATTERN
1231 006712 001311                BNE    2$              ;BR UNTIL DONE
1232                                     ;
1233                                     ;
1234                                     ;*****
1235                                     ;*TEST 25      FLOAT A 1 ACROSS LATCHING INPUT BITS
1236                                     ;*****
1236 006714 000004                $T25: SCOPE
1237 006716 012737 006766 001110      MOV    #25,$LPERR        ;LOAD ERROR SCOPE RETURN
1238 006724 012737 000001 001124      MOV    #BIT0,$GDDAT     ;LOAD EXPECTED VALUE
1239 006732 012737 000000 001544      MOV    #0,$TMDAT        ;CLEAR OUTPUT REG
1240                                     ;
1241                                     ;*   MOV    $TMDAT,$GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1242 006750 012737 177777 001544      MOV    #-1,$TMDAT      ;CLEAR INPUT
1243                                     ;
1244                                     ;*   MOV    $TMDAT,$GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1245

```

1246	006766	033737	001124	001510	2S:	BIT	\$GDDAT,NOTLCH	;TEST FOR NON-LATCHING
1247	006774	001030				BNE	1\$	;BR IF NON-LATCH
1248	006776	033737	001124	001534		BIT	\$GDDAT,ODDJMP	;TEST IF ODD JUMPER
1249	007004	001024				BNE	1\$	;BYPASS IF ODD JUMPER
1250								



```

1251
1252
1253 007016 012737 000000 001544 ;* MOV SGDDAT,@GRDIO ;/ PUT DATA FROM SGDDAT TO DEVICE REG GRDIO
1254 ;* MOV #0,STMDAT ;CLEAR OUTPUT REGISTER
1255
1256 ;* MOV STMDAT,@GRDIO ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1257
1258 007044 023737 001124 001126 ;* MOV @GRDAI,@SBDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN SBDDAT.
1259 007052 001401 ;* CMP SGDDAT,@SBDDAT ;COMPARE
1260 007054 104002 ;* BEQ #1 ;BR IF EQUAL
1261 ;* ERROR 2 ;INPUT REGISTER FAILED TO LATCH DATA
1262 007056 1$:
1263
1264 ;* MOV @GRDAI,STMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1265 007066 053737 001124 001544 ;* BIS SGDDAT,STMDAT
1266
1267 ;* MOV STMDAT,@GRDAI ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1268 007104 006337 001124 ;* ASL SGDDAT ;CHANGE PATTERN
1269 007110 001326 ;* BNE #25 ;BR UNTIL DONE
1270
1271 ;*****
1272 ;*TEST 26 FLOAT A 0 ACROSS LATCHING INPUT BITS
1273 ;*****
1274 007112 000004 ;*ST26: SCOPE
1275 007114 012737 007164 001110 MOV #25,$LPERR ;LOAD ERROR SCOPE RETURN
1276 007122 012737 000001 001532 MOV #BIT0,BRLEV3 ;LOAD EXPECTED
1277 007130 012737 000000 001544 MOV #0,STMDAT ;CLEAR OUTPUT REGISTER
1278
1279 ;* MOV STMDAT,@GRDIO ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1280 007146 012737 177777 001544 ;* MOV #-1,STMDAT
1281
1282 ;* MOV STMDAT,@GRDAI ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1283
1284 007164 033737 001532 001510 2$: BIT BRLEV3,NOTLCH ;TEST FOR LATCHING
1285 007172 001041 ;* BNE #1 ;BR IF NOT
1286 007174 033737 001124 001534 BIT SGDDAT,ODDJMP ;TEST IF ODD JUMPER
1287 007202 001035 ;* BNE #1 ;BYPASS IF ODD JUMPER BIT
1288
1289 007204 012737 177777 001124 MOV #-1,$GDDAT ;LOAD
1290 007212 043737 001510 001124 BIC NOTLCH,$GDDAT
1291 007220 043737 001532 001124 BIC BRLEV3,$GDDAT ;MAKE BRLEV3
1292
1293 ;* MOV SGDDAT,@GRDIO ;/ PUT DATA FROM SGDDAT TO DEVICE REG GRDIO
1294 007236 012737 000000 001544 ;* MOV #0,STMDAT ;CLEAR OUTPUT REGISTER
1295
1296 ;* MOV STMDAT,@GRDIO ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1297
1298
1299 ;* MOV @GRDAI,@SBDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN SBDDAT.
1300 007264 023737 001124 001126 ;* CMP SGDDAT,@SBDDAT ;COMPARE
1301 007272 001401 ;* BEQ #1 ;BR IF EQUAL
1302 007274 104002 ;* ERROR 2 ;INPUT REGISTER FAILED TO LATCH DATA
1303
1304 007276 1$:

```

```

1305
1306
1307 007306 053737 001532 001544 ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1308
1309
1310 007324 006337 001532 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1311 007330 001315 ;* ASL BRLEV3 ;CHANGE PATTERN
1312 ;* BNE 2$ ;BR UNTIL DONE
1313 ;*****
1314 ;*TEST 27 TEST FOR SLOW INPUT GATES WITH #125252
1315 007332 000004 ;*****
1316 ;*ST27: SCOPE
1317 007334 012737 125252 001124 MOV #125252,$GDDAT ;LOAD EXPECTED
1318 007342 043737 001510 001124 BIC NOTLCH,$GDDAT ;CONVERT
1319 007350 043737 001534 001124 BIC ODDJMP,$GDDAT ;MASK
1320 007356 013700 001124 MOV $GDDAT,RO ;LOAD PATTERN
1321 007362 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
1322
1323
1324 007400 012737 177777 001544 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1325
1326 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1327
1328
1329 ;* MOV RO,@GRDIO ;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1330 007426 010037 001544 MOV RO,$TMDAT
1331
1332 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1333 007442 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
1334
1335 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1336
1337 ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1338 007470 013701 001544 MOV $TMDAT,R1
1339
1340 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1341 007504 005100 COM RO
1342
1343 ;* MOV RO,@GRDIO ;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1344 007516 010037 001544 MOV RO,$TMDAT
1345
1346 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1347 007532 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
1348
1349 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1350
1351 ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1352 007560 013701 001544 MOV $TMDAT,R1
1353
1354 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1355 007574 005100 COM RO
1356
1357 ;* MOV RO,@GRDIO ;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1358 007606 010037 001544 MOV RO,$TMDAT

```





1413					;* MOV	RO,GRDIO	;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1414	010146	010037	001544		MOV	RO,STMDAT	
1415					;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1416					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1417	010162	012737	000000	001544	;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1418					MOV	GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1419					;* MOV	STMDAT,R1	
1420	010210	013701	001544		MOV	STMDAT,GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1421					COM	RO	
1422	010224	005100			;* MOV	RO,GRDIO	;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1423					MOV	RO,STMDAT	
1424	010236	010037	001544		;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1425					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1426					;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1427	010252	012737	000000	001544	;* MOV	GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1428					MOV	STMDAT,R1	
1429	010300	013701	001544		;* MOV	STMDAT,GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1430					COM	RO	
1431	010314	005100			;* MOV	RO,GRDIO	;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1432					MOV	RO,STMDAT	
1433	010326	010037	001544		;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1434					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1435					;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1436	010342	012737	000000	001544	;* MOV	GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1437					MOV	STMDAT,R1	
1438	010370	013701	001544		;* MOV	STMDAT,GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1439					COM	RO	
1440	010404	005100			;* MOV	RO,GRDIO	;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1441					MOV	RO,STMDAT	
1442	010416	010037	001544		;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1443					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1444					;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1445	010432	012737	000000	001544	;* MOV	GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1446					MOV	STMDAT,R1	
1447	010460	013701	001544		;* MOV	STMDAT,GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1448					COM	RO	
1449	010416	010037	001544		;* MOV	RO,GRDIO	;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1450					MOV	RO,STMDAT	
1451	010432	012737	000000	001544	;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1452					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1453					;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1454	010460	013701	001544		;* MOV	GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1455					MOV	STMDAT,R1	
1456					;* MOV	STMDAT,GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1457					COM	RO	
1458					;* MOV	RO,GRDIO	;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1459					MOV	RO,STMDAT	
1460					;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1461					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1462					;* MOV	STMDAT,GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1463	010460	013701	001544		;* MOV	GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1464					MOV	STMDAT,R1	
1465					;* MOV	STMDAT,GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1466					COM	RO	



```

1467 010474 005100          COM      RO
1468
1469          ;*      MOV      RO,@GRDIO      ;/ PUT DATA FROM RO TO DEVICE REG GRDIO
1470 010506 010037 001544          MOV      RO,$TMDAT
1471
1472          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1473 010522 012737 000000 001544          MOV      @0,$TMDAT      ;CLEAR OUTPUT REGISTER
1474
1475          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1476
1477          ;*      MOV      @GRDAI,$TMDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1478 010550 013701 001544          MOV      $TMDAT,R1
1479
1480          ;*      MOV      $TMDAT,@GRDAI      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1481 010564 005100          COM      RO
1482
1483 010566 010137 001126          MOV      R1,$BDDAT      ;LOAD READ
1484 010572 000240          NOP
1485 010574 000240          NOP
1486 010576 000240          NOP
1487 010600 023737 001124 001126          CMP      $GDDAT,$BDDAT      ;COMPARE
1488 010606 001401          BEQ      TST30      ;;BR IF EQUAL
1489 010610 104002          ERROR      2      ;INPUT GATE SLOW
1490
1491          ;*****
1492          ;*TEST 30      TEST FOR SLOW INPUT GATES WITH #52525
1493          ;*****
1494 010612 000004          TST30:  SCOPE
1495
1496 010614 012737 052525 001124          MOV      #52525,$GDDAT      ;SETUP EXPECTED
1497 010622 043737 001534 001124          BIC      ODDJMP,$GDDAT      ;MASK ODD JUMPER BITS
1498 010630 043737 001510 001124          BIC      NOTLCH,$GDDAT      ;CONVERT
1499 010636 013700 001124          MOV      $GDDAT,RO
1500 010642 012737 000000 001544          MOV      @0,$TMDAT      ;CLEAR OUTPUT REGISTER
1501
1502          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1503 010660 012737 177777 001544          MOV      @-1,$TMDAT
1504
1505          ;*      MOV      $TMDAT,@GRDAI      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1506
1507 010676 010037 001544          MOV      RO,$TMDAT
1508
1509          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1510 010712 012737 000000 001544          MOV      @0,$TMDAT      ;CLEAR OUTPUT REGISTER
1511
1512          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1513
1514          ;*      MOV      @GRDAI,$TMDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1515 010740 013701 001544          MOV      $TMDAT,R1
1516
1517          ;*      MOV      $TMDAT,@GRDAI      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1518 010754 005100          COM      RO
1519 010756 010037 001544          MOV      RO,$TMDAT
1520

```

```

1521          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1522 010772 012737 000000 001544      MOV      #0, STMDAT    ;CLEAR OUTPUT REGISTER
1523
1524          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1525
1526          ;*      MOV      @GRDAI, STMDAT    ;/ READ DEVICE REG GRDAI, PUT DATA IN STMDAT.
1527 011020 013701 001544      MOV      STMDAT, R1
1528
1529          ;*      MOV      STMDAT, @GRDAI    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1530 011034 005100      COM      RO
1531 011036 010037 001544      MOV      RO, STMDAT
1532
1533          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1534 011052 012737 000000 001544      MOV      #0, STMDAT    ;CLEAR OUTPUT REGISTER
1535
1536          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1537
1538          ;*      MOV      @GRDAI, STMDAT    ;/ READ DEVICE REG GRDAI, PUT DATA IN STMDAT.
1539 011100 013701 001544      MOV      STMDAT, R1
1540
1541          ;*      MOV      STMDAT, @GRDAI    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1542 011114 005100      COM      RO
1543 011116 010037 001544      MOV      RO, STMDAT
1544
1545          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1546 011132 012737 000000 001544      MOV      #0, STMDAT    ;CLEAR OUTPUT REGISTER
1547
1548          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1549
1550          ;*      MOV      @GRDAI, STMDAT    ;/ READ DEVICE REG GRDAI, PUT DATA IN STMDAT.
1551 011160 013701 001544      MOV      STMDAT, R1
1552
1553          ;*      MOV      STMDAT, @GRDAI    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1554 011174 005100      COM      RO
1555 011176 010037 001544      MOV      RO, STMDAT
1556
1557          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1558 011212 012737 000000 001544      MOV      #0, STMDAT    ;CLEAR OUTPUT REGISTER
1559
1560          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1561
1562          ;*      MOV      @GRDAI, STMDAT    ;/ READ DEVICE REG GRDAI, PUT DATA IN STMDAT.
1563 011240 013701 001544      MOV      STMDAT, R1
1564
1565          ;*      MOV      STMDAT, @GRDAI    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1566 011254 005100      COM      RO
1567 011256 010037 001544      MOV      RO, STMDAT
1568
1569          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1570 011272 012737 000000 001544      MOV      #0, STMDAT    ;CLEAR OUTPUT REGISTER
1571
1572          ;*      MOV      STMDAT, @GRDIO    ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1573
1574          ;*      MOV      @GRDAI, STMDAT    ;/ READ DEVICE REG GRDAI, PUT DATA IN STMDAT.

```



1575	011320	013701	001544		MOV	STMDAT,R1	
1576							
1577					;	*	
1578	011334	005100			MOV	STMDAT,@GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1579	011336	010037	001544		COM	RO	
1580					MOV	RO,STMDAT	
1581					;	*	
1582	011352	012737	000000	001544	MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1583					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1584					;	*	
1585					MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1586					;	*	
1587	011400	013701	001544		MOV	@GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1588					MOV	STMDAT,R1	
1589					;	*	
1590	011414	005100			MOV	STMDAT,@GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1591	011416	010037	001544		COM	RO	
1592					MOV	RO,STMDAT	
1593					;	*	
1594	011432	012737	000000	001544	MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1595					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1596					;	*	
1597					MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1598					;	*	
1599	011460	013701	001544		MOV	@GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1600					MOV	STMDAT,R1	
1601					;	*	
1602	011474	005100			MOV	STMDAT,@GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1603	011476	010037	001544		COM	RO	
1604					MOV	RO,STMDAT	
1605					;	*	
1606	011512	012737	000000	001544	MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1607					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1608					;	*	
1609					MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1610					;	*	
1611	011540	013701	001544		MOV	@GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1612					MOV	STMDAT,R1	
1613					;	*	
1614	011554	005100			MOV	STMDAT,@GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1615	011556	010037	001544		COM	RO	
1616					MOV	RO,STMDAT	
1617					;	*	
1618	011572	012737	000000	001544	MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1619					MOV	#0,STMDAT	;CLEAR OUTPUT REGISTER
1620					;	*	
1621					MOV	STMDAT,@GRDIO	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1622					;	*	
1623	011620	013701	001544		MOV	@GRDAI,STMDAT	;/READ DEVICE REG GRDAI,PUT DATA IN STMDAT.
1624					MOV	STMDAT,R1	
1625					;	*	
1626	011634	005100			MOV	STMDAT,@GRDAI	;/ PUT DATA FROM STMDAT TO DEVICE REG GRDAI
1627	011636	010037	001544		COM	RO	
1628					MOV	RO,STMDAT	

```

1629          ;*      MOV      $TMDAT, @GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1630 011652 012737 000000 001544      MOV      #0, $TMDAT    ;CLEAR OUTPUT REGISTER
1631
1632          ;*      MOV      $TMDAT, @GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1633
1634          ;*      MOV      @GRDAI, $TMDAT    ;/READ DEVICE REG GRDAI, PUT DATA IN $TMDAT.
1635 011700 013701 001544      MOV      $TMDAT, R1
1636
1637          ;*      MOV      $TMDAT, @GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1638 011714 005100      COM      RO
1639
1640 011716 010137 001126      MOV      R1, $BDDAT    ;LOAD VALUE READ
1641 011722 000240      NOP
1642 011724 000240      NOP
1643 011726 000240      NOP
1644 011730 023737 001124 001126      CMP      $GDDAT, $BDDAT    ;COMPARE
1645 011736 001401      BEQ      TST31    ;;BR IF EQUAL
1646 011740 104002      ERROR    2
1647          ;:*****
1648          ;*TEST 31      TEST THAT RESET CLEARS INPUT REGISTER BITS
1649          ;:*****
1650 011742 000004      TST31:  SCOPE
1651 011744 012737 000002 001160      MOV      #2, $TIMES    ;;DO 2 ITERATIONS
1652 011752 012737 000000 001544      MOV      #0, $TMDAT    ;CLEAR OUTPUT REGISTER
1653
1654          ;*      MOV      $TMDAT, @GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1655 011770 012737 177777 001544      MOV      #-1, $TMDAT
1656
1657          ;*      MOV      $TMDAT, @GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1658 012006 012737 177777 001544      MOV      #-1, $TMDAT    ;LOAD OUTPUT
1659
1660          ;*      MOV      $TMDAT, @GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1661 012024 005037 001124      CLR      $GDDAT    ;LOAD EXPECTED
1662 012030 004737 024354      JSR      PC, $RESET
1663
1664          ;*      MOV      @GRDAI, $BDDAT    ;/READ DEVICE REG GRDAI, PUT DATA IN $BDDAT.
1665 012044 043737 001534 001126      BIC      ODDJMP, $BDDAT    ;MASK ODD JUMPERS
1666 012052 005737 001126      TST      $BDDAT
1667 012056 001401      BEQ      TST32    ;;BR IF ALL BITS CLEARED
1668 012060 104002      ERROR    2    ;INPUT REG. FAILED TO CLEAR UPON RESET INST.
1669
1670          ;:*****
1671          ;*TEST 32      TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
1672          ;:*****
1673 012062 000004      TST32:  SCOPE
1674 012064 012737 000200 001124      MOV      #BIT7, $GDDAT    ;LOAD EXPECTED
1675 012072 012737 000000 001544      MOV      #0, $TMDAT
1676
1677          ;*      MOV      $TMDAT, @GRSTAT    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1678
1679          ;*      MOV      $TMDAT, @GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1680 012120 022727 000000 000000      CMP      #0, #0    ;DELAY
1681
1682          ;*      MOV      @GRSTAT, $BDDAT    ;/READ DEVICE REG GRSTAT, PUT DATA IN $BDDAT.

```



```

1683 012136 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1684 012144 001401                      BEQ      TST33              ;;BR IF EQUAL
1685 012146 104001                      ERROR    1                  ;INPUT DATA READY FLAG FAILED TO SET
1686
1687 ;TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
1688
1689 ;*****
1690 ;*TEST 33 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
1691 ;*****
1692 012150 000004                      TST33: SCOPE
1693 012152 012737 100000 001124      MOV      #BIT15,$GDDAT      ;LOAD EXPECTED
1694 012160 012737 000000 001544      MOV      #0,$TMDAT
1695
1696 ;* MOV      $TMDAT,$GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1697 ;* MOV      $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1698 012206 022727 000000 000000      CMP      #0,#0
1699
1700 ;* MOV      $GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1701 012224 013700 001544              MOV      $TMDAT,$R0
1702
1703 ;* MOV      $GRSTAT,$BDDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
1704 012240 005737 001126              TST      $BDDAT
1705 012244 100401                      BMI     TST34              ;;BR IF SET
1706 012246 104001                      ERROR    1                  ;INPUT DATA READY FLAG FAILED TO SET
1707
1708
1709
1710 012250
1711
1712 DRT21:
1713 ;*****
1714 ;*TEST 34 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
1715 ;*****
1716 012250 000004                      TST34: SCOPE
1717 012252 032777 002000 166660      BIT      #BIT10,$SWR        ;TEST CABLE SWITCH
1718 012260 001165                      BNE     TST35              ;;BYPASS IF NO I/O CABLE
1719
1720 012262 012737 012276 001110      MOV      #15,$LPERR        ;LOAD ERROR SCOPE RETURN
1721 012270 012737 000001 001532      MOV      #BIT0,$RLEV3      ;LOAD INTERRUPT BIT
1722 012276 005037 001126 15:      CLR      $BDDAT           ;CLEAR BAD DATA
1723 012302 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD GOOD DATA
1724
1725 012310 033737 001532 001512      BIT      $RLEV3,INTBIT     ;TEST IF THIS BIT WILL INTERRUPT
1726 012316 001543                      BEQ     3$                 ;;NO TRY NEXT BIT
1727 012320 012737 000000 001544      MOV      #0,$TMDAT        ;CLEAR OUTPUT REGISTER
1728
1729 ;* MOV      $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1730 ;* MOV      $TMDAT,$GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1731 012354 012737 000000 001544      MOV      #0,$TMDAT        ;CLEAR STATUS
1732
1733 ;* MOV      $TMDAT,$GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1734 ;* MOV      $GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1735 012402 053737 001532 001544      BIS     $RLEV3,$TMDAT
1736

```

```

1737
1738 ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1739 ;* MOV @GRDIO, $TMDAT ;/ READ DEVICE REG GRDIO, PUT DATA IN $TMDAT.
1740 012430 043737 001532 001544 ;* BIC BRLEV3, $TMDAT
1741 ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1742 ;* MOV @GRDIO, $TMDAT ;/ READ DEVICE REG GRDIO, PUT DATA IN $TMDAT.
1743 012456 053737 001532 001544 ;* BIS BRLEV3, $TMDAT
1744 ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1745 012474 005077 167000 ;* CLR @GRSTAT ; CLEAR FLAG FROM DATA READY
1746 ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1747 ;* MOV @GRSTAT, $TMDAT ;/ READ DEVICE REG GRSTAT, PUT DATA IN $TMDAT.
1748 ;* MOV $TMDAT, @GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1749 ;* IF INTERRUPT INPUT SWITCH IS ON
1750 012520 105737 001544 ;* TSTB $TMDAT
1751 012524 100401 ;* BMI 2$ ; BR IF SET
1752 012526 104010 ;* ERROR 10 ; INPUT INTERRUPT BIT FAILED TO SET INPUT READY
1753 ;* ; ?? DID OPERATOR GIVE CORRECT
1754 ;* ; INPUT INTERRUPT BITS ??
1755
1756 012530 2$:
1757 ;* MOV @GRDIO, $TMDAT ;/ READ DEVICE REG GRDIO, PUT DATA IN $TMDAT.
1758 012540 043737 001532 001544 ;* BIC BRLEV3, $TMDAT
1759 ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1760 ;* MOV @GRDAI, $TMDAT ;/ READ DEVICE REG GRDAI, PUT DATA IN $TMDAT.
1761 012566 053737 001532 001544 ;* BIS BRLEV3, $TMDAT
1762 ;* MOV $TMDAT, @GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1763 012604 005037 001124 ;* CLR $GDDAT ; CLEAR EXPECTED
1764 012610 005077 166664 ;* CLR @GRSTAT ; CLEAR STATUS
1765 012614 117737 166660 001126 ;* MOV @GRSTAT, $BDDAT ; READ STATUS
1766 012622 100001 ;* BPL 3$ ; BR IF CLEARED
1767 012624 104001 ;* ERROR 1 ; INPUT READY FAILED TO CLEAR
1768
1769 012626 006337 001532 3$: ASL BRLEV3 ; CHANGE BIT
1770 012632 001221 ;* BNE 1$ ; BR IF NOT DONE
1771 ;* *****
1772 ;* TEST 35 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
1773 ;* *****
1774 ;* ST35: SCOPE
1775 012634 000004 ;* BIT #BIT10, @SWR ; TEST CABLE SWITCH
1776 012636 032777 002000 166274 ;* BNE TST36 ; BYPASS IF NO I/O CABLE
1777 012644 001127
1778
1779 012646 012737 012662 001110 ;* MOV #1$, $LPERR ; LOAD ERROR SCOPE RETURN
1780 012654 012737 000001 001532 ;* MOV #BIT0, BRLEV3 ; LOAD NON-INTERRUPT BIT
1781 012662 012737 000200 001126 1$: ;* MOV #200, $BDDAT ; LOAD BAD DATA

```



J04

```

1791 012670 005037 001124 CLR $GDDAT ;CLEAR GOOD DATA
1792
1793 012674 033737 001532 001512 BIT BRLEV3,INTBIT ;TEST IF THIS BIT WILL INTERRUPT
1794 012702 001105 BNE 3$ ;;YES SKIP AND TRY NEXT BIT
1795 012704 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
1796
1797 ;* MOV $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1798 012722 012737 177777 001544 MOV #-1,$TMDAT
1799
1800 ;* MOV $TMDAT,$GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1801 012740 012737 000000 001544 MOV #0,$TMDAT ;CLEAR STATUS
1802
1803 ;* MOV $TMDAT,$GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1804
1805 ;* MOV $GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1806 012766 053737 001532 001544 BIS BRLEV3,$TMDAT
1807
1808 ;* MOV $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1809 013004 043737 001532 001544 BIC BRLEV3,$TMDAT ;CLEAR OUTPUT
1810
1811 ;* MOV $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1812
1813 ;* MOV $GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1814 013032 053737 001532 001544 BIS BRLEV3,$TMDAT
1815
1816 ;* MOV $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1817
1818 ;* MOV $TMDAT,$GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1819 013060 012737 000000 001544 MOV #0,$TMDAT ;CLEAR FLAG FROM DATA READY
1820
1821 ;* MOV $TMDAT,$ ;/ PUT DATA FROM $TMDAT TO DEVICE REG
1822 ;SHOULD REMAIN SET VIA DIRECT SET SIDE
1823
1824 ;* MOV $GRSTAT,$TMDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
1825 013106 105737 001544 TSTB $TMDAT
1826 013112 100001 BPL 3$ ;;BR IF CLEAR
1827 013114 104011 ERROR 11 ;INPUT NON-INTERRUPT BIT SET INPUT READY
;?? DID OPERATOR GIVE CORRECT
;INPUT INTERRUPT BITS ??
1828
1829
1830
1831
1832 013116 006337 001532 3$: ASL BRLEV3 ;CHANGE BIT
1833 013122 001257 BNE 1$ ;BR IF NOT DONE
1834 ;*****
1835 ;*TEST 36 DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
1836 ;*****
1837 013124 000004 †ST36: SCOPE
1838 013126 012737 000001 001160 MOV #1,$TIMES ;;DO 1 ITERATION
1839 013134 005737 001464 BYPASS: TST NBEXT ;TEST IF ANY
1840 013140 001415 BEQ 1$ ;BR IF NONE
1841 013142 032777 010000 165770 BIT #SW12,$SWR ;TEST BIT12 OF SWR
1842 013150 001024 BNE BYPASS ;INHIBIT TESTING NEXT DR11-K
1843 013152 162737 000010 001466 SUB #10,$DRADD ;UPDATE DEVICE ADDRESS
1844 013160 062737 000010 001470 ADD #10,$DRIV ;UPDATE DEVICE VECTOR

```

```

1845 013166 005337 001464          DEC      NBEXT          ; ANOTHER ONE ?
1846 013172 000413                BR        BYPAS1       ; BR IF ANOTHER
1847 013174 013737 001452 001466 1S:  MOV      BASEBA,DRADD ; RELOAD ADDRESS
1848 013202 013737 001454 001470  MOV      BASEIV,DRIV   ; RELOAD VECTOR
1849 013210 013737 001462 001464  MOV      NMBEXT,NBEXT  ; RELOAD NUMBER
1850 013216 000137 013232          JMP      SEOP          ; DONE
1851 013222 012700 177777          BYPAS1: MOV     #-1,RO  ;
1852 013226 000137 003360          JMP      RBEG2         ; TEST ANOTHER UNIT

```

.SBTTL END OF PASS ROUTINE

```

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

```

```

1863 013232                SEOP:
1864 013232 000004          CLR      SCOPE
1865 013234 005037 001102          CLR      $TSTNM      ;; ZERO THE TEST NUMBER
1866 013240 005037 001160          CLR      $TIMES      ;; ZERO THE NUMBER OF ITERATIONS
1867 013244 005237 001176          INC      $PASS       ;; INCREMENT THE PASS NUMBER
1868 013250 042737 100000 001176  BIC      #100000,$PASS ; DON'T ALLOW A NEG. NUMBER
1869 013256 005327          DEC      (PC)+       ;; LOOP?
1870 013260 000001          SEOPCT: .WORD 1
1871 013262 003022          BGT      SDOAGN      ;; YES
1872 013264 012737          MOV      (PC)+,a(PC)+ ; RESTORE COUNTER
1873 013266 000001          SENDCT: .WORD 1
1874 013270 013260          SEOPCT
1875 013272 104401 013337          TYPE    SENDMG      ;; TYPE "END PASS #"
1876 013276 013746 001176          MOV     $PASS,-(SP)  ; SAVE $PASS FOR TYPEOUT
1877 013302 104405          TYPDS   ;; GO TYPE--DECIMAL ASCII WITH SIGN
1878 013304 104401 013334          TYPE    $NULL      ;; TYPE A NULL CHARACTER
1879 013310 013700 000042          SGET42: MOV     a#42,RO ; GET MONITOR ADDRESS
1880 013314 001405          BEQ     SDOAGN      ;; BRANCH IF NO MONITOR
1881 013316 000005          RESET  ;; CLEAR THE WORLD
1882 J13320 004710          SENDAD: JSR    PC,(RO) ; GO TO MONITOR
1883 013322 000240          NOP    ;; SAVE ROOM
1884 013324 000240          NOP    ;; FOR
1885 013326 000240          NOP    ;; ACT11
1886 013330
1887 013330 000137          SDOAGN: JMP     a(PC)+       ;; RETURN
1888 013332 013222          SRTNAD: .WORD  BYPAS1
1889 013334 377 377 000          SENULL: .BYTE  -1,-1,0 ; NULL CHARACTER STRING
1890 013337 015 042412 042116          SENDMG: .ASCIZ <15><12>/END PASS #/
1891 013344 050040 051501 020123
1892 013352 000043

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

;*****
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED

```

1893  
1894  
1895  
1896  
1897  
1898



```

1899          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1900          ;*REPLACED WITH SPACES.
1901          ;*CALL:
1902          ;*      MOV      NUM,-(SP)          ;:PUT THE BINARY NUMBER ON THE STACK
1903          ;*      TYPDS          ;:GO TO THE ROUTINE
1904
1905          STYPDS:
1906          MOV      R0,-(SP)          ;:PUSH R0 ON STACK
1907          MOV      R1,-(SP)          ;:PUSH R1 ON STACK
1908          MOV      R2,-(SP)          ;:PUSH R2 ON STACK
1909          MOV      R3,-(SP)          ;:PUSH R3 ON STACK
1910          MOV      R5,-(SP)          ;:PUSH R5 ON STACK
1911          MOV      #20200,-(SP)      ;:SET BLANK SWITCH AND SIGN
1912          MOV      20(SP),R5        ;:GET THE INPUT NUMBER
1913          BPL      1$                ;:BR IF INPUT IS POS.
1914          NEG      R5                ;:MAKE THE BINARY NUMBER POS.
1915          MOV      #'-',1(SP)        ;:MAKE THE ASCII NUMBER NEG.
1916          CLR      R0                ;:ZERO THE CONSTANTS INDEX
1917          MOV      #SDBLK,R3        ;:SETUP THE OUTPUT POINTER
1918          MOV      #'',(R3)+        ;:SET THE FIRST CHARACTER TO A BLANK
1919          CLR      R2                ;:CLEAR THE BCD NUMBER
1920          MOV      $DTBL(R0),R1     ;:GET THE CONSTANT
1921          SUB      R1,R5             ;:FORM THIS BCD DIGIT
1922          BLT      4$                ;:BR IF DONE
1923          INC      R2                ;:INCREASE THE BCD DIGIT BY 1
1924          BR      3$
1925          ADD      R1,R5             ;:ADD BACK THE CONSTANT
1926          TST      R2                ;:CHECK IF BCD DIGIT=0
1927          BNE      5$                ;:FALL THROUGH IF 0
1928          TSTB    (SP)              ;:STILL DOING LEADING 0'S?
1929          BMI      7$                ;:BR IF YES
1930          ASLB    (SP)              ;:MSD?
1931          BCC      6$                ;:BR IF NO
1932          MOV      1(SP),-1(R3)      ;:YES--SET THE SIGN
1933          BIS      #'0,R2           ;:MAKE THE BCD DIGIT ASCII
1934          BIS      #' ,R2           ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
1935          MOV      R2,(R3)+        ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
1936          TST      (R0)+            ;:JUST INCREMENTING
1937          CMP      R0,#10           ;:CHECK THE TABLE INDEX
1938          BLT      2$                ;:GO DO THE NEXT DIGIT
1939          BGT      8$                ;:GO TO EXIT
1940          MOV      R5,R2            ;:GET THE LSD
1941          BR      6$                ;:GO CHANGE TO ASCII
1942          TSTB    (SP)+            ;:WAS THE LSD THE FIRST NON-ZERO?
1943          BPL      9$                ;:BR IF NO
1944          MOV      -1(SP),-2(R3)     ;:YES--SET THE SIGN FOR TYPING
1945          CLRB    (R3)              ;:SET THE TERMINATOR
1946          MOV      (SP)+,R5         ;:POP STACK INTO R5
1947          MOV      (SP)+,R3         ;:POP STACK INTO R3
1948          MOV      (SP)+,R2         ;:POP STACK INTO R2
1949          MOV      (SP)+,R1         ;:POP STACK INTO R1
1950          MOV      (SP)+,R0         ;:POP STACK INTO R0
1951          TYPE    $SDBLK           ;:NOW TYPE THE NUMBER
1952          MOV      2(SP),4(SP)      ;:ADJUST THE STACK
    
```

```

1953 013554 012616          MOV      (SP)+,(SP)
1954 013556 000002          RTI
1955 013560 023420          SDTBL:  10000.          ;;RETURN TO USER
1956 013562 001750          1000.
1957 013564 000144          100.
1958 013566 000012          10.
1959 013570 000004          SDBLK:  .BLKW  4
1960
1961          .SBTTL  MISC. EXTERNAL LOGIC TEST
1962 013600 012706 001100          EXTTST: MOV      #STACK,SP
1963 013604 004737 003424          JSR      PC,SETADD      ;SET UP ADDRESS AND VECTOR
1964 013610 012737 040000 013760 2S:  MOV      #BIT14,EXTCNT ;LOAD COUNT
1965 013616 012737 000000 001544 1S:  MOV      #0,STMDAT      ;CLEAR OUTPUT REGISTER
1966 013616 012737 000000 001544
1967
1968          ;*  MOV      STMDAT,@GRDIO ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1969 013634 012737 125252 001544          MOV      #125252,STMDAT ;LOAD OUTPUT
1970
1971          ;*  MOV      STMDAT,@GRDIO ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1972
1973          ;*  MOV      @GRDAI,EXTTMP ;/READ DEVICE REG GRDAI,PUT DATA IN EXTTMP.
1974
1975          ;*  MOV      EXTTMP,@GRDAI ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
1976 013672 012737 000000 001544          MOV      #0,STMDAT      ;CLEAR OUTPUT REGISTER
1977
1978          ;*  MOV      STMDAT,@GRDIO ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1979 013710 012737 052525 001544          MOV      #52525,STMDAT ;LOAD OUTPUT
1980
1981          ;*  MOV      STMDAT,@GRDIO ;/ PUT DATA FROM STMDAT TO DEVICE REG GRDIO
1982
1983          ;*  MOV      @GRDAI,EXTTMP ;/READ DEVICE REG GRDAI,PUT DATA IN EXTTMP.
1984
1985          ;*  MOV      EXTTMP,@GRDAI ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
1986 013746 005337 013760          DEC      EXTCNT      ;FINISHED COUNT
1987 013752 001321          BNE     1S
1988 013754 000715          BR      2S            ;LOOP
1989
1990          EXTTMP: 0
1991          EXTCNT: 0
1992
1993          ;SUBROUTINE TO LOAD TRAP AND POWER FAIL VECTORS
1994 013762 012737 022440 000034  SETTRP: MOV      #STRAP,@TRAPVEC ;LOAD TRAP VECTOR
1995 013770 012737 000340 000036          MOV      #340,@TRAPVEC+2
1996 013776 012737 020464 000024          MOV      #SPWRDN,@PWRVEC ;LOAD POWER FAIL VECTOR
1997 014004 012737 000340 000026          MOV      #340,@PWRVEC+2
1998 014012 012737 014022 000004          MOV      #1S,@#4
1999 014020 000207          RTS     PC            ;SET UP FOR UNEXPECTED TRAP
2000 014022 022626          1S:  CMP      (SP)+,(SP)+ ;EXIT
2001 014024 104401 015647          TYPE   ,BUSTRP
2002 014030 000000          HALT
2003 014032 000137 001546          JMP
2004          .SBTTL  BEGIN COULTER INTERFACE TEST
2005 014036 012706 001100          COULTR: MOV      #STACK,SP
2006 014042 004737 013762          JSR      PC,SETTRP      ;LOAD TRAP AND POWER FAIL VECTORS

```



```

2007 014046 004737 003424 JSR PC,SETADD ;SETUP BUS ADDRESS AND VECTOR
2008 014052 012737 000000 001544 MOV #0,$TMDAT ;CLEAR INPUT STATUR REG
2009
2010 ;* MOV $TMDAT,$GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
2011 ;LOAD INPUT VECTOR
2012 014070 104401 015710 1$: TYPE,RUNMSG ;TELL OPERATOR TO "RUN SAMPLE"
2013 014074 012701 022522 MOV #BUFFER,R1 ;LOAD POINTER FOR RESULTS
2014 ;WAIT FOR OPERATOR
2015 014100 4$:
2016
2017 ;* MOV $GRSTAT,$TMDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
2018 014110 105737 001544 TSTB $TMDAT
2019 014114 100371 BPL 4$
2020 014116 012737 000106 013756 MOV #70,$EXTTMP ;LOAD # OF 1 MSEC DELAYS
2021 014124 013700 001460 6$: MOV CPU,R0 ;LOAD 1 MSEC. DELAY WEIGHT ;240 FOR 11/05 ;620 FOR 11/40
2022 014130 005300 3$: DEC R0 ;DELAY
2023 014132 001376 BNE 3$
2024 014134 005337 013756 DEC EXTTMP ;FINISHED ALL MSEC. DELAY ?
2025 014140 001371 BNE 6$ ;BR IF NOT
2026
2027 ;* MOV $GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
2028 014152 013700 001544 MOV $TMDAT,R0
2029 014156 010021 MOV R0,(R1)+ ;SAVE IN BUFFER
2030 014160 012737 177777 001544 MOV #-1,$TMDAT ;CLEAR INPUT
2031
2032 ;* MOV $TMDAT,$GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
2033 014176 042700 007777 BIC #7777,R0 ;MASK
2034 014202 001336 BNE 4$ ;BR IF NOT LAST SAMPLE
2035 014204 010103 MOV R1,R3 ;COPY R1
2036 014206 014300 5$: MOV -(R3),R0 ;GET RESULTS
2037 014210 004737 014224 JSR PC,40$ ;CONVERT AND TYPE
2038 014214 022703 022522 CMP #BUFFER,R3 ;FINISHED ?
2039 014220 001723 BEQ 1$ ;BR IF YES
2040 014222 000771 BR 5$ ;CONT. TYPING UNTIL DONE
2041
2042 ;SUBROUTINE FOR THE COULTER TEST
2043 014224 012737 000055 015732 40$: MOV #55,MSGRUS+2 ;FIX ASCII MESSAGE
2044 014232 012702 015737 MOV #MSPNT1,R2 ;LOAD DEST. POINTER
2045 014236 012737 000004 013756 MOV #4,EXTTMP ;LOAD COUNT
2046 014244 000404 BR 12$
2047 014246 006000 10$: ROR R0
2048 014250 006000 ROR R0
2049 014252 006000 ROR R0
2050 014254 006000 ROR R0
2051 014256 010001 12$: MOV R0,R1 ;LOAD R1
2052 014260 042701 177760 BIC #177760,R1 ;MASK
2053 014264 052701 000060 BIS #60,R1 ;MAKE DIGIT
2054 014270 110142 MOVB R1,-(R2) ;SAVE DIGIT
2055 014272 005337 013756 DEC EXTTMP ;FINISHED ?
2056 014276 001363 BNE 10$ ;BR IF NOT
2057 014300 000337 015732 SWAB MSGRUS+2 ;ADJUST MESSAGE
2058 014304 104401 015730 TYPE,MSGRUS
2059 014310 000207 RTS PC ;EXIT
2060 014312 012706 001100 LOCBOX: MOV #STACK,SP
    
```

```

2061 014316 004737 013762 JSR PC,SETTRP
2062 014322 104401 015470 TYPE LOCHDR ; TELL OPERATOR WHAT THIS IS
2063 014326 004537 014432 JSR R5,INADR ; GET THE ADDRESSES
2064 014332 001472 014616 LOC1,LOC1Y
2065 014336 000412 BR LOCA ; BR IF DONE
2066 014340 004537 014432 JSR R5,INADR ; GET NEXT ADDRESS
2067 014344 001474 014620 LOC2,LOC2Y
2068 014350 000405 BR LOCA ; BR IF DONE
2069 014352 004537 014432 JSR R5,INADR ; GET NEXT ADDRESS
2070 014356 001476 014622 LOC3,LOC3Y
2071 014362 000400 BR LOCA
2072
2073 014364 005737 014616 LOCA: TST LOC1Y ; TEST IF SELECTED
2074 014370 001403 BEQ 1$ ; BR IF NOT
2075 014372 004537 014544 JSR R5,LOUP1 ; TEST FOR SWITCHES AND LOAD LAMPS
2076 014376 001472 LOC1
2077 014400 005737 014620 1$: TST LOC2Y ; TEST IF SELECTED
2078 014404 001403 BEQ 2$ ; BR IF NOT
2079 014406 004537 014544 JSR R5,LOUP1 ; LOAD NEXT SET
2080 014412 001474 LOC2
2081 014414 005737 014622 2$: TST LOC3Y ; TEST IF SELECTED
2082 014420 001403 BEQ 3$ ; BR IF NOT
2083 014422 004537 014544 JSR R5,LOUP1 ; LOAD NEXT SET
2084 014426 001476 LOC3
2085 014430 000755 BR LOCA ; LOOP BACK
2086
2087 014432 013537 014536 INADR: MOV @ (R5)+,10$ ; GET BUS ADDRESS
2088 014436 012537 014542 MOV (R5)+,12$ ; GET INDICATOR
2089 014442 005077 000074 CLR @12$ ; CLEAR FLAG
2090 014446 104401 015531 1$: TYPE, INADRH ; ASK FOR INPUT
2091 014452 013746 014536 MOV 10$,-(SP) ; TYPE CURRENT ADDRESS
2092 014456 104402 TYPOC
2093 014460 104401 015574 TYPE INADRH ; ADD YES NO
2094 014464 104411 RDLIN ; READ CHAR. AND ECHO
2095 014466 000240 NOP
2096 014470 000240 NOP
2097 014472 013637 014540 MOV @ (SP)+,11$ ; GET THE ANSWER
2098 014476 042737 177640 014540 BIC #177640,11$ ; MASK OFF BITS
2099 014504 001413 BEQ 3$ ; BR IF CR
2100 014506 022737 000116 014540 CMP #'N,11$ ; TEST IF NO
2101 014514 001406 BEQ 2$ ; BR IF NO
2102 014516 022737 000131 014540 CMP #'Y,11$ ; TEST IF YES
2103 014524 001350 BNE 1$ ; BR IF OTHER
2104 014526 005277 000010 INC @12$ ; SET YES FLAG
2105 014532 005725 2$: TST (R5)+
2106 014534 000205 3$: RTS R5 ; EXIT
2107 014536 000000 10$: 0
2108 014540 000000 11$: 0
2109 014542 000000 12$: 0
2110
2111 ; SUBROUTINE TO LOAD THE LAMPS FROM THE LOC BOX SWITCHES
2112 014544 013537 014612 LOUP1: MOV @ (R5)+,40$ ; GET ARG.
2113 014550 001001 BNE 1$ ; BR IF YES
2114 014552 000205 RTS R5 ; EXIT
    
```



```

2115 014554 012702 014612 1S: MOV 40S,R2
2116 014560 062702 000002 ADD #2,R2
2117 014564 010203 MOV R2,R3
2118 014566 062703 000002 ADD #2,R3
2119 014572 011200 MOV (R2),R0
2120 014574 011301 MOV (R3),R1
2121 014576 040100 BIC R1,R0 ;MASK OFF BITS
2122 014600 041213 BIC (R2),(R3) ;LOAD OUTPUT BITS
2123 014602 060013 ADD R0,(R3) ;LOAD
2124 014604 012712 177777 MOV #-1,(R2) ;CLEAR INPUT
2125 014610 000205 RTS ;EXIT

2126
2127 014612 000000 40S: 0
2128 014614 000000 41S: 0
2129 014616 000000 LOC1Y: 0 ;YES/NO ANSWER TO 10C BOX #1
2130 014620 000000 LOC2Y: 0
2131 014622 000000 LOC3Y: 0

2132
2133
2134 014624 012706 001100 XLO1AD: .SBTTL XLO1 ADJUSTMENT ROUTINE
2135 014630 004737 013762 MOV #STACK,SP
2136 014634 104401 015612 JSR PC,SETTRP
2137 014640 004537 014432 TYPE XLOHDR ;TELL OPERATOR
2138 014644 001472 014616 JSR R5,INADR ;GET BUS ADDRESS
2139 014650 000412 BR XLO10
2140 014652 004537 014432 JSR R5,INADR ;
2141 014656 001474 014620 LOC2,LOC2Y
2142 014662 000405 BR XLO10
2143 014664 004537 014432 JSR R5,INADR
2144 014670 001476 014622 LOC3,LOC3Y
2145 014674 000400 BR XLO10

2146
2147 014676 005737 014616 XLO10: TST LOC1Y ;TEST IF SELECTED
2148 014702 001431 BEQ 1S ;BR IF NOT SELECTED
2149 014704 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2150 014710 000 000 .BYTE 0,0 ;LOAD CH# AND SHIFT COUNT
2151 014712 000007 ? ;LOAD DATA MASK
2152 014714 001472 LOC1 ;DR11K BUS ADDRESS
2153 014716 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2154 014722 001 003 .BYTE 1,3 ;LOAD CH# AND SHIFT COUNT
2155 014724 000070 70 ;LOAD DATA MASK
2156 014726 001472 LOC1 ;DR11K BUS ADDRESS
2157 014730 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2158 014734 002 006 .BYTE 2,6 ;LOAD CH# AND SHIFT COUNT
2159 014736 000700 700 ;LOAD DATA MASK
2160 014740 001472 LOC1 ;DR11K BUS ADDRESS
2161 014742 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2162 014746 003 011 .BYTE 3,11 ;LOAD CH# AND SHIFT COUNT
2163 014750 007000 7000 ;LOAD DATA MASK
2164 014752 001472 LOC1 ;DR11K BUS ADDRESS
2165 014754 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2166 014760 004 014 .BYTE 4,14 ;LOAD CH# AND SHIFT COUNT
2167 014762 070000 70000 ;LOAD DATA MASK
2168 014764 001472 LOC1 ;DR11K BUS ADDRESS

```

```

2169 014766 005737 014620 1S: TST LOC2Y ;TEST IF SELECTED
2170 014772 001431 BEQ 2S ;BR IF NOT
2171 014774 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2172 015000 005 000 .BYTE 5,0 ;LOAD CH# AND SHIFT COUNT
2173 015002 000007 ? ;LOAD DATA MASK
2174 015004 001474 LOC2 ;DRIK BUS ADDRESS
2175 015006 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2176 015012 006 003 .BYTE 6,3 ;LOAD CH# AND SHIFT COUNT
2177 015014 000070 ? ;LOAD DATA MASK
2178 015016 001474 LOC2 ;DRIK BUS ADDRESS
2179 015020 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2180 015024 007 006 .BYTE 7,6 ;LOAD CH# AND SHIFT COUNT
2181 015026 000700 ? ;LOAD DATA MASK
2182 015030 001474 LOC2 ;DRIK BUS ADDRESS
2183 015032 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2184 015036 010 011 .BYTE 10,11 ;LOAD CH# AND SHIFT COUNT
2185 015040 007000 ? ;LOAD DATA MASK
2186 015042 001474 LOC2 ;DRIK BUS ADDRESS
2187 015044 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2188 015050 011 014 .BYTE 11,14 ;LOAD CH# AND SHIFT COUNT
2189 015052 070000 ? ;LOAD DATA MASK
2190 015054 001474 LOC2 ;DRIK BUS ADDRESS
2191 015056 005737 014622 2S: TST LOC3Y ;TEST IF SELECTED
2192 015062 001431 BEQ 3S ;BR IF NOT
2193 015064 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2194 015070 012 000 .BYTE 12,0 ;LOAD CH# AND SHIFT COUNT
2195 015072 000007 ? ;LOAD DATA MASK
2196 015074 001476 LOC3 ;DRIK BUS ADDRESS
2197 015076 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2198 015102 013 003 .BYTE 13,3 ;LOAD CH# AND SHIFT COUNT
2199 015104 000070 ? ;LOAD DATA MASK
2200 015106 001476 LOC3 ;DRIK BUS ADDRESS
2201 015110 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2202 015114 014 006 .BYTE 14,6 ;LOAD CH# AND SHIFT COUNT
2203 015116 000700 ? ;LOAD DATA MASK
2204 015120 001476 LOC3 ;DRIK BUS ADDRESS
2205 015122 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2206 015126 015 011 .BYTE 15,11 ;LOAD CH# AND SHIFT COUNT
2207 015130 007000 ? ;LOAD DATA MASK
2208 015132 001476 LOC3 ;DRIK BUS ADDRESS
2209 015134 004537 015152 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2210 015140 016 014 .BYTE 16,14 ;LOAD CH# AND SHIFT COUNT
2211 015142 070000 ? ;LOAD DATA MASK
2212 015144 001476 LOC3 ;DRIK BUS ADDRESS
2213 015146 000137 014676 3S: JMP XLO10
2214 ;ANALOG CONVERSION AND LOC BOX DRIVER
2215 XLOADJ:
2216
2217 ;* MOV @ADCS1,$TMDAT ;/READ DEVICE REG ADCS1,PUT DATA IN $TMDAT.
2218 015162 112537 001544 MOVB (R5)+,$TMDAT
2219
2220 ;* MOV $TMDAT,@ADCS1 ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS1
2221
2222 ;* MOV @ADCS,$TMDAT ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.

```



```

2223 015206 052737 020000 001544      BIS      #BIT13,$TMDAT
2224
2225      ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2226 015224 112537 015454      ;*      MOV      (R5)+,40$      ;GET SHIFT COUNT
2227 015230 012537 015460      ;*      MOV      (R5)+,42$      ;GET DATA MASK
2228 015234 013537 015456      ;*      MOV      @($R5)+,41$      ;GET DR11K BUS ADDRESS
2229 015240 001504      ;*      BEQ      70$      ;BR IF NONE
2230
2231      ;*      MOV      $ADCS,$TMDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2232 015252 105237 001544      ;*      INCB     $TMDAT
2233
2234      ;*      MOV      $TMDAT,$ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2235 015266 005001      ;*      CLR      R1
2236 015270 062737 000004 015456      ;*      ADD      #4,41$      ;GET LAMP ADDRESS
2237 015276      ;*      1$:
2238
2239      ;*      MOV      $ADCS,$TMDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2240 015306 105737 001544      ;*      TSTB    $TMDAT
2241 015312 100371      ;*      BPL      1$
2242
2243      ;*      MOV      $ADBR,$TMDAT      ;/READ DEVICE REG ADBR,PUT DATA IN $TMDAT.
2244 015324 013700 001544      ;*      MOV      $TMDAT,R0
2245 015330 162700 000004      ;*      SUB      #4,R0      ;TEST IF AT UPPER END
2246 015334 100432      ;*      BMI      7$      ;BR IF YES
2247 015336 162700 000014      ;*      SUB      #14,R0      ;TEST IF NEAR UPPER END
2248 015342 100425      ;*      BMI      6$      ;BR IF YES
2249 015344 162700 000060      ;*      SUB      #60,R0      ;TEST IF GETTING NEAR UPPER END
2250 015350 100420      ;*      BMI      5$      ;BR IF YES
2251 015352 162700 001530      ;*      SUB      #1530,R0      ;TEST IF GETTING NEAR LOWER END
2252 015356 100407      ;*      BMI      2$      ;BR IF YES
2253 015360 162700 000060      ;*      SUB      #60,R0      ;TEST IF NEAR LOWER END
2254 015364 100406      ;*      BMI      3$      ;BR IF YES
2255 015366 162700 000014      ;*      SUB      #14,R0      ;TEST IF AT LOWER END
2256 015372 100405      ;*      BMI      4$      ;BR IF YES
2257 015374 000412      ;*      BR       7$      ;BR IF AT STOP
2258 015376 012701 000004      ;*      MOV      #4,R1      ;LOAD #
2259 015402 062701 000002      ;*      ADD      #2,R1
2260 015406 005201      ;*      INC      R1
2261 015410 000404      ;*      BR       7$      ;BR AND LOAD LAMP'S
2262 015412 012701 000002      ;*      MOV      #2,R1      ;LOAD #
2263 015416 062701 000004      ;*      ADD      #4,R1
2264 015422 013700 015454      ;*      MOV      40$,R0      ;LOAD SHIFT COUNT
2265 015426 006301      ;*      ASL      R1      ;SHUFFEL THE DATA
2266 015430 005300      ;*      DEC      R0
2267 015432 100375      ;*      BPL      10$      ;BR IF NOT DONE
2268 015434 006201      ;*      ASR      R1      ;RE ADJUST
2269 015436 043777 015460 000012      ;*      BIC      42$,@41$      ;CLEAR OLD DATA IN LAMPS
2270 015444 050177 000006      ;*      BIS      R1,@41$      ;LOAD THE LAMPS
2271 015450 005001      ;*      CLR      R1
2272 015452 000205      ;*      RTS      R5      ;EXIT
2273
2274 015454 000000      ;*      40$: 0
2275 015456 000000      ;*      41$: 0
2276 015460 000000      ;*      42$: 0
    
```

2277	015462	170400			ADCS:	170400	
2278	015464	170401			ADCS1:	170401	
2279	015466	170402			ADBR:	170402	
2280	015470	005015	047514	026503	LOCHDR:	.ASCIZ	<15><12>/LOC-BOX SWITCH AND LAMP LOOP/<15><12>
2281	015476	047502	020130	053523			
2282	015504	052111	044103	040440			
2283	015512	042116	046040	046501			
2284	015520	020120	047514	050117			
2285	015526	005015	000				
2286	015531	015	052412	042523	INADRH:	.ASCIZ	<15><12>/USE LOC-BOX <DR11K AT BUS ADDR. /
2287	015536	046040	041517	041055			
2288	015544	054117	036040	051104			
2289	015552	030461	020113	052101			
2290	015560	041040	051525	040440			
2291	015566	042104	027122	000040			
2292	015574	037040	020040	020131	INDADR:	.ASCIZ	/ > Y OR N ? /
2293	015602	051117	047040	037440			
2294	015610	000040					
2295	015612	005015	046130	030460	XLOHDR:	.ASCIZ	<15><12>/XLO1 POT ADJUSTMENT LOOP/<15><12>
2296	015620	050040	052117	040440			
2297	015626	045104	051525	046524			
2298	015634	047105	020124	047514			
2299	015642	050117	005015	000			
2300	015647	015	041012	051525	BUSTRP:	.ASCIZ	<15><12>/BUS TIME-OUT ON SELECTED DR11K/
2301	015654	052040	046511	026505			
2302	015662	052517	020124	047117			
2303	015670	051440	046105	041505			
2304	015676	042524	020104	051104			
2305	015704	030461	000113				
2306	015710	005015	051012	047125	RUNMSG:	.ASCIZ	<15><12><12>/RUN SAMPLE/<15><12>
2307	015716	051440	046501	046120			
2308	015724	006505	000012				
2309	015730	005015	047055	047116	MSGRUS:	.ASCII	<15><12>/-NNNN/
2310	015736	116					
2311	015737	000			MSPNT1:	.BYTE	0
2312						.EVEN	
2313							
2314							
2315	015740	042523	020124	053523	SWNLB:	.ASCIZ	/SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING INPUT BITS/
2316	015746	052111	044103	051040			
2317	015754	043505	051511	042524			
2318	015762	020122	044502	051524			
2319	015770	042440	052521	046101			
2320	015776	052040	020117	044124			
2321	016004	020105	047516	026516			
2322	016012	040514	041524	044510			
2323	016020	043516	044440	050116			
2324	016026	052125	041040	052111			
2325	016034	000123					
2326	016036	042523	020124	053523	SWINTB:	.ASCIZ	/SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING INPUT BITS/
2327	016044	052111	044103	051040			
2328	016052	043505	051511	042524			
2329	016060	020122	044502	051524			
2330	016066	042440	052521	046101			



```

2331 016074 052040 020117 044124
2332 016102 020105 047111 042524
2333 016110 051122 050125 044524
2334 016116 043516 044440 050116
2335 016124 052125 041040 052111
2336 016132 000123
2337 016134 042523 020124 053523
2338 016142 052111 044103 051040
2339 016150 043505 051511 042524
2340 016156 020122 044502 051524
2341 016164 030440 026465 031061
2342 016172 042440 052521 046101
2343 016200 052040 020117 047520
2344 016206 044523 044524 042526
2345 016214 044440 050116 052125
2346 016222 041040 052111 000123
2347 016230 042523 020124 053523
2348 016236 052111 044103 051040
2349 016244 043505 051511 042524
2350 016252 020122 044502 051524
2351 016260 030440 026465 031061
2352 016268 042440 052521 046101
2353 016274 052040 020117 051124
2354 016302 047101 044523 044524
2355 016310 047117 044440 050116
2356 016316 052125 041040 052111
2357 016324 000123
2358 016326 042523 020124 053523
2359 016334 052111 044103 051040
2360 016342 043505 051511 042524
2361 016350 020122 044527 044124
2362 016356 052040 042510 042040
2363 016364 051505 051111 042105
2364 016372 050040 047522 051107
2365 016400 046501 047440 052120
2366 016406 047511 051516 000
2367 016413 015 042012 050105
2368 016420 042522 051523 041440
2369 016426 047117 027124 005015
2370 016434 000
2371 016435 123 040524 052524
2372 016442 020123 042522 044507
2373 016450 052123 051105 044440
2374 016456 020116 051105 047522
2375 016464 000122
2376 016466 047111 052520 020124
2377 016474 042522 044507 052123
2378 016502 051105 044440 020116
2379 016510 051105 047522 000122
2380 016516 052517 050124 052125
2381 016524 051040 043505 051511
2382 016532 042524 020122 047111
2383 016540 042440 051122 051117
2384 016546 000

```

SWPOSB: .ASCIZ /SET SWITCH REGISTER BITS 15-12 EQUAL TO POSITIVE INPUT BITS/

SWTRAB: .ASCIZ /SET SWITCH REGISTER BITS 15-12 EQUAL TO TRANSITION INPUT BITS/

SWDPOB: .ASCIZ /SET SWITCH REGISTER WITH THE DESIRED PROGRAM OPTIONS/

DEPCNT: .ASCIZ <15><12>/DEPRESS CONT./<15><12>

EM1: .ASCIZ /STATUS REGISTER IN ERROR/

EM2: .ASCIZ /INPUT REGISTER IN ERROR/

EM3: .ASCIZ /OUTPUT REGISTER IN ERROR/

2385	016547	111	050116	052125	EM4:	.ASCIZ	/INPUT FAILED TO INTERRUPT/
2386	016554	043040	044501	042514			
2387	016562	020104	047524	044440			
2388	016570	052116	051105	052522			
2389	016576	052120	000				
2390	016601	117	052125	052520	EM5:	.ASCIZ	/OUTPUT FAILED TO INTERRUPT/
2391	016606	020124	040506	046111			
2392	016614	042105	052040	020117			
2393	016622	047111	042524	051122			
2394	016630	050125	000124				
2395	016634	047125	054105	042520	EM6:	.ASCIZ	/UNEXPECTED INTERRUPT/
2396	016642	052103	042105	044440			
2397	016650	052116	051105	052522			
2398	016656	052120	000				
2399	016661	117	042520	040522	EM7:	.ASCIZ	/OPERATOR INTERVENTION ERROR/
2400	016666	047524	020122	047111			
2401	016674	042524	053122	047105			
2402	016702	044524	047117	042440			
2403	016710	051122	051117	000			
2404	016715	111	052116	051105	EM10:	.ASCIZ	/INTERRUPT INPUT BIT FAILED TO SET INPUT READY FLAG/
2405	016722	052522	052120	044440			
2406	016730	050116	052125	041040			
2407	016736	052111	043040	044501			
2408	016744	042514	020104	047524			
2409	016752	051440	052105	044440			
2410	016760	050116	052125	051040			
2411	016766	040506	054504	043040			
2412	016774	040514	000107				
2413	017000	047516	026516	047111	EM11:	.ASCIZ	/NON-INTERRUPT INPUT BIT SET INPUT READY FLAG/
2414	017006	042524	051122	050125			
2415	017014	020124	047111	052520			
2416	017022	020124	044502	020124			
2417	017030	042523	020124	047111			
2418	017036	052520	020124	042522			
2419	017044	042101	020131	046106			
2420	017052	043501	000				
2421	017055	105	051122	041520	DH1:	.ASCIZ	/ERRPC DRADD TSTNUM STATUS EXPECTED/
2422	017062	020040	042040	040522			
2423	017070	042104	052011	052123			
2424	017076	052516	020115	020040			
2425	017104	052123	052101	051525			
2426	017112	020040	054105	042520			
2427	017120	052103	042105	000			
2428	017125	105	051122	041520	DH2:	.ASCIZ	/ERRPC DRADD TSTNUM INPUT EXPECTED/
2429	017132	020040	042040	040522			
2430	017140	042104	052011	052123			
2431	017146	052516	004515	047111			
2432	017154	052520	020124	020040			
2433	017162	054105	042520	052103			
2434	017170	042105	000				
2435	017173	105	051122	041520	DH3:	.ASCIZ	/ERRPC DRADD TSTNUM OUTPUT EXPECTED/
2436	017200	020040	042040	040522			
2437	017206	042104	052011	052123			



```

2439 017214 052516 004515 052517
2440 017222 050124 052125 020040
2441 017230 054105 042520 052103
2442 017236 042105 000
2443 017241 105 051122 041520
2444 017246 020040 042040 040522
2445 017254 042104 052011 052123
2446 017262 052516 000115
2447 017266 051105 050122 020103
2448 017274 020040 051104 042101
2449 017302 004504 051524 047124
2450 017310 046525 051411 040524
2451 017316 052524 020123 042440
2452 017324 050130 041505 020124
2453 017332 044440 050116 052125
2454 017340 041040 052111 000
2455 017346 001116 001466 017426
2456 017354 001126 001124 000000
2457 017362 001116 001466 017426
2458 017370 000000
2459 017372 001116 001466 017426
2460 017400 001126 001124 001532
2461 017406 000000
2462 017410 000000 000000 000000
2463 017416 000000 000000 000000
2464 017424 000000
2465 017426 000000

```

```

DH4: .ASCIZ /ERRPC DRADD TSTNUM/
DH10: .ASCIZ /ERRPC DRADD TSTNUM STATUS EXPECT INPUT BIT/
DT1: $ERRPC, DRADD, TSTNUM, $BDDAT, $GDDAT, 0
DT4: $ERRPC, DRADD, TSTNUM, 0
DT10: $ERRPC, DRADD, TSTNUM, $BDDAT, $GDDAT, $RLEV3, 0
DF0: 0,0,0,0,0,0,0
TSTNUM: 0

```

.SBTTL SCOPE HANDLER ROUTINE

```

;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($TSTNUM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1 LOOP ON TEST
;SW11=1 INHIBIT ITERATIONS
;SW09=1 LOOP ON ERROR
;SW08=1 LOOP ON TEST IN SWR<7:0>
;CALL SCOPE ;;SCOPE=IOT
;SCOPE:
1$: CKSWR ;TEST FOR CHANGE IN SOFT-SWR
BIT #BIT14,$SWR ;LOOP ON PRESENT TEST?
BNE $OVER ;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$ ;IF RUNNING ON THE "XOR" TESTER CHANGE
;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV @#ERRVEC, -(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5$, @#ERRVEC ;SET FOR TIMEOUT
TST @#177060 ;TIME OUT ON XOR?
MOV (SP)+, @#ERRVEC ;RESTORE THE ERROR VECTOR

```

```

2481 017430
2482 017430 104407
2483 017432 032777 040000 161500
2484 017440 001114
2485 017442 000416
2486
2487
2488
2489 017444 013746 000004
2490 017450 012737 017470 000004
2491 017456 005737 177060
2492 017462 012637 000004

```

```

2493 017466 000463          BR      $SVLAD          ;; GO TO THE NEXT TEST!
2494 017470 022626          SS:    CMP      (SP)+,(SP)+      ;; CLEAR THE STACK AFTER A TIME OUT
2495 017472 012637 000004    MOV      (SP)+,@ERRVEC      ;; RESTORE THE ERROR VECTOR
2496 017476 000423          BR      7$              ;; LOOP ON THE PRESENT TEST!
2497 017500          6$:;#####END OF CODE FOR THE XOR TESTER#####
2498 017500 032777 000400 161432    BIT      @BIT08,@SWR        ;; LOOP ON SPEC. TEST?
2499 017506 001404          BEQ      2$              ;; BR IF NO
2500 017510 127737 161424 001102    CMPB    @SWR,$STSTNM        ;; ON THE RIGHT TEST? SWR<7:0>
2501 017516 001465          BEQ      $OVER           ;; BR IF YES
2502 017520 105737 001103          2$:    TSTB    $ERFLG        ;; HAS AN ERROR OCCURRED?
2503 017524 001421          BEQ      3$              ;; BR IF NO
2504 017526 123737 001115 001103    CMPB    $ERMAX,$ERFLG      ;; MAX. ERRORS FOR THIS TEST OCCURRED?
2505 017534 101015          BHI      3$              ;; BR IF NO
2506 017536 032777 001000 161374    BIT      @BIT09,@SWR        ;; LOOP ON ERROR?
2507 017544 001404          BEQ      4$              ;; BR IF NO
2508 017546 013737 001110 001106    7$:    MOV      $LPERR,$LPADR    ;; SET LOOP ADDRESS TO LAST SCOPE
2509 017554 000446          BR      $OVER           ;;
2510 017556 105037 001103          4$:    CLRB    $ERFLG        ;; ZERO THE ERROR FLAG
2511 017562 005037 001160          CLR      $TIMES          ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
2512 017564 000415          BR      1$              ;; ESCAPE TO THE NEXT TEST
2513 017570 032777 004000 161342    3$:    BIT      @BIT11,@SWR    ;; INHIBIT ITERATIONS?
2514 017576 001011          BNE      1$              ;; BR IF YES
2515 017600 005737 001176          TST     $PASS           ;; IF FIRST PASS OF PROGRAM
2516 017604 001406          BEQ      1$              ;; INHIBIT ITERATIONS
2517 017606 005237 001104          INC     $ICNT           ;; INCREMENT ITERATION COUNT
2518 017612 023737 001160 001104    CMP     $TIMES,$ICNT      ;; CHECK THE NUMBER OF ITERATIONS MADE
2519 017620 002024          BGE     $OVER           ;; BR IF MORE ITERATION REQUIRED
2520 017622 012737 000001 001104    1$:    MOV      @1,$ICNT      ;; REINITIALIZE THE ITERATION COUNTER
2521 017630 013737 017706 001160    $SVLAD: MOV      $SMXCNT,$TIMES   ;; SET NUMBER OF ITERATIONS TO DO
2522 017636 105237 001102          INCB    $STSTNM         ;; COUNT TEST NUMBERS
2523 017642 113737 001102 001174    MOVB   $STSTNM,$STSTNM    ;; SET TEST NUMBER IN APT MAILBOX
2524 017650 011637 001106          MOV     (SP),$LPADR      ;; SAVE SCOPE LOOP ADDRESS
2525 017654 011637 001110          MOV     (SP),$LPERR      ;; SAVE ERROR LOOP ADDRESS
2526 017660 005037 001162          CLR     $ESCAPE         ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2527 017664 112737 000001 001115    $OVER: MOVB   @1,$ERMAX    ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2528 017672 013777 001102 161242    MOV     $STSTNM,@DISPLAY  ;; DISPLAY TEST NUMBER
2529 017700 013716 001106          MOV     $LPADR,(SP)     ;; FUDGE RETURN ADDRESS
2530 017704 000002          RTI                    ;; FIXES PS
2531 017706 000010          $SMXCNT: 8              ;; MAX. NUMBER OF ITERATIONS
2532          .SBTTL  ERROR HANDLER ROUTINE
2533
2534          ;; *****
2535          ;; *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2536          ;; *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2537          ;; *AND GO TO SER,TYP ON ERROR
2538          ;; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2539          ;; *SW15=1      HALT ON ERROR
2540          ;; *SW13=1      INHIBIT ERROR TYPEOUTS
2541          ;; *SW09=1     LOOP ON ERROR
2542          ;; *CALL
2543          ;; *      ERROR      N      ;; ERROR=EMT AND N=ERROR ITEM NUMBER
2544
2545 017710          $ERROR:  CKSWR          ;; TEST FOR CHANGE IN SOFT-SWR
2546 017710 104407

```



```

DR11-K LOGIC TEST MAINDEC-11-DRLPF-A MACY11 27(654) 14-DEC-77 20:26 PAGE 51
DRLPF.P11 ERROR HANDLER ROUTINE

2547 017712 113737 001102 017426      MOVB   $STNM,TSTNUM
2548 017720 105237 001103      7$:   INCB   $ERFLG      ;; SET THE ERROR FLAG
2549 017724 001775      BEQ    7$             ;; DON'T LET THE FLAG GO TO ZERO
2550 017726 013777 001102 161206      MOV    $STNM,$DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
2551 017734 005237 001112      INC    $ERTTL        ;; INC THE ERROR COUNT
2552 017740 011637 001116      MOV    (SP),$ERRPC    ;; GET ADDRESS OF ERROR INSTRUCTION
2553 017744 162737 000002 001116      SUB    #2,$ERRPC
2554 017752 117737 161140 001114      MOVB   $ERRPC,$ITEMB  ;; STRIP AND SAVE THE ERROR ITEM CODE
2555 017760 032777 020000 161152      BIT    #BIT13,$SWR    ;; SKIP TYPEOUT IF SET
2556 017766 001004      BNE   20$           ;; SKIP TYPEOUTS
2557 017770 004737 020102      JSR   PC,$ERRTYP     ;; GO TO USER ERROR ROUTINE
2558 017774 104401 001165      TYPE  , $CRLF
2559 020000
2560 020000 122737 000001 001210      20$:  CMPB   #APTENV,$ENV  ;; RUNNING IN APT MODE
2561 020006 001007      BNE   2$            ;; NO SKIP APT ERROR REPORT
2562 020010 113737 001114 020022      MOVB   $ITEMB,21$    ;; SET ITEM NUMBER AS ERROR NUMBER
2563 020016 004737 022210      JSR   PC,$SATY4     ;; REPORT FATAL ERROR TO APT
2564 020022      000
2565 020023      000
2566 020024 000777      21$:  .BYTE  0
2567 020026 005777 161106      22$:  BR     22$          ;; APT ERROR LOOP
2568 020032 100002      2$:   TST    $SWR      ;; HALT ON ERROR
2569 020034 000000      BPL   3$            ;; SKIP IF CONTINUE
2570 020036 104407      HALT
2571 020040 032777 001000 161072      3$:   CKSWR
2572 020046 001402      BIT    #BIT09,$SWR  ;; TEST FOR CHANGE IN SOFT-SWR
2573 020050 013716 001110      BEQ    4$           ;; LOOP ON ERROR SWITCH SET?
2574 020054 005737 001162      MOV    $LPERR,(SP)  ;; BR IF NO
2575 020060 001402      TST   $ESCAPE      ;; FUDGE RETURN FOR LOOPING
2576 020062 013716 001162      BEQ    5$           ;; CHECK FOR AN ESCAPE ADDRESS
2577 020066      MOV    $ESCAPE,(SP) ;; BR IF NONE
2578 020066 022737 013320 000042      5$:   CMP    #SENDAD,$#42 ;; ACT-11 AUTO-ACCEPT?
2579 020074 001001      BNE   6$           ;; BRANCH IF NO
2580 020076 000000      HALT
2581 020100      6$:   RTI            ;; YES
2582 020100 000002      ;; RETURN
2583
2584      .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
2585
2586      ;; *****
2587      ;; *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
2588      ;; *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
2589      ;; *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
2590
2591      $ERRTYP:
2592 020102 104401 001165      TYPE  $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
2593 020106 010046      MOV   R0,-(SP)     ;; SAVE R0
2594 020110 005000      CLR  R0            ;; PICKUP THE ITEM INDEX
2595 020112 153700 001114      BISB #,$ITEMB,R0
2596 020116 001004      BNE  1$            ;; IF ITEM NUMBER IS ZERO, JUST
2597      1$:   TYPE  PC OF THE ERROR
2598 020120 013746 001116      MOV   $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
2599      2$:   ERROR ADDRESS
2600 020124 104402      TYPE  $CRLF        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)

```

DR11-K LOGIC TEST MAINDEC-11-DRLPF-A MACY11 27(654) 14-DEC-77 20:26 PAGE 52  
 DRLPF.P11 ERROR MESSAGE TYPEOUT ROUTINE

```

2601 020126 000426
2602 020130 005300
2603 020132 006300
2604 020134 006300
2605 020136 006300
2606 020140 062700 001252
2607 020144 012037 020154
2608 020150 001404
2609 020152 104401
2610 020154 000000
2611 020156 104401 001165
2612 020162 012037 020172
2613 020166 001404
2614 020170 104401
2615 020172 000000
2616 020174 104401 001165
2617 020200 011000
2618 020202 001004
2619 020204 012600
2620 020206 104401 001165
2621 020212 000207
2622 020214
2623 020214 013046
2624 020216 104402
2625 020220 005710
2626 020222 001770
2627 020224 104401 020232
2628 020230 000771
2629 020232 020040 000
2630 020236
  
```

```

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

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
* MOV NUM,-(SP) ;; NUMBER TO BE TYPED
* TYPOS ;; CALL FOR TYPEOUT
* .BYTE N ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
* .BYTE M ;; M=1 OR 0
* ;; 1=TYPE LEADING ZEROS
* ;; 0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
* MOV NUM,-(SP) ;; NUMBER TO BE TYPED
* TYPON ;; CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
* MOV NUM,-(SP) ;; NUMBER TO BE TYPED
  
```

```

2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
  
```





2709  
2710  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729  
2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753  
2754  
2755  
2756  
2757  
2758  
2759  
2760  
2761  
2762

.SBTTL POWER DOWN AND UP ROUTINES

\*\*\*\*\*

:POWER DOWN ROUTINE

```

$PWRDN: MOV $SILLUP, @PWRVEC ;; SET FOR FAST UP
        MOV #340, @PWRVEC+2 ;; Prio:7
        MOV RO, -(SP) ;; PUSH RO ON STACK
        MOV R1, -(SP) ;; PUSH R1 ON STACK
        MOV R2, -(SP) ;; PUSH R2 ON STACK
        MOV R3, -(SP) ;; PUSH R3 ON STACK
        MOV R4, -(SP) ;; PUSH R4 ON STACK
        MOV R5, -(SP) ;; PUSH R5 ON STACK
        MOV @SWR, -(SP) ;; PUSH @SWR ON STACK
        MOV SP, $SAVR6 ;; SAVE SP
        MOV $PWRUP, @PWRVEC ;; SET UP VECTOR
        HALT
        BR -.2 ;; HANG UP

```

\*\*\*\*\*

:POWER UP ROUTINE

```

$PWRUP: MOV $SILLUP, @PWRVEC ;; SET FOR FAST DOWN
        MOV $SAVR6, SP ;; GET SP
        CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
        IS: INC $SAVR6 ;; WAIT FOR THE INC
        BNE IS OF WORD
        MOV (SP)+, @SWR ;; POP STACK INTO @SWR
        MOV (SP)+, R5 ;; POP STACK INTO R5
        MOV (SP)+, R4 ;; POP STACK INTO R4
        MOV (SP)+, R3 ;; POP STACK INTO R3
        MOV (SP)+, R2 ;; POP STACK INTO R2
        MOV (SP)+, R1 ;; POP STACK INTO R1
        MOV (SP)+, R0 ;; POP STACK INTO R0
        MOV $PWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
        MOV #340, @PWRVEC+2 ;; Prio:7
        TYPE PWRMSG ;; REPORT THE POWER FAILURE
        SPWRMG: .WORD PWRMSG ;; POWER FAIL MESSAGE POINTER
        MOV (PC)+, (SP) ;; RESTART AT IOTEST
        SPWRAD: .WORD IOTEST ;; RESTART ADDRESS
        RTI
        $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
        BR -.2 ;; BEFORE THE POWER DOWN WAS COMPLETE
        $SAVR6: 0 ;; PUT THE SP HERE
        PWRMSG: .ASCIIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>

```

.EVEN

.SBTTL TYPE ROUTINE



```

2763 *****
2764 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2765 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2766 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2767 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2768 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2769 *
2770 *CALL:
2771 *1) USING A TRAP INSTRUCTION
2772 *      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2773 *OR
2774 *      TYPE
2775 *      MESADR
2776 *
2777
2778 020704 105737 001157 STYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
2779 020710 100002      BPL 1$          ;; BR IF YES
2780 020712 000000      HALT      ;; HALT HERE IF NO TERMINAL
2781 020714 000430      BR 3$          ;; LEAVE
2782 020716 010046      1$: MOV RO, -(SP)      ;; SAVE RO
2783 020720 017600 000002      MOV 32(SP), RO      ;; GET ADDRESS OF ASCIZ STRING
2784 020724 122737 000001 001210      CMPB #APTENV, $ENV      ;; RUNNING IN APT MODE
2785 020732 001011      BNE 62$          ;; NO GO CHECK FOR APT CONSOLE
2786 020734 132737 000100 001211      BITB #APTSPool, $ENVm      ;; SPOOL MESSAGE TO APT
2787 020742 001405      BEQ 62$          ;; NO GO CHECK FOR CONSOLE
2788 020744 010037 020754      MOV RO, 61$      ;; SETUP MESSAGE ADDRESS FOR APT
2789 020750 004737 022200      JSR PC, $ATY3      ;; SPOOL MESSAGE TO APT
2790 020754 000000      61$: .WORD 0      ;; MESSAGE ADDRESS
2791 020756 132737 000040 001211      62$: BITB #APTCSUP, $ENVm      ;; APT CONSOLE SUPPRESSED
2792 020764 001003      BNE 60$          ;; YES, SKIP TYPE OUT
2793 020766 112046      2$: MOVB (RO)+, -(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2794 020770 001005      BNE 4$          ;; BR IF IT ISN'T THE TERMINATOR
2795 020772 005726      TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
2796 020774 012600      60$: MOV (SP)+, RO      ;; RESTORE RO
2797 020776 062716 000002      3$: ADD #2, (SP)      ;; ADJUST RETURN PC
2798 021002 000002      RTI      ;; RETURN
2799 021004 122716 000011      4$: CMPB #HT, (SP)      ;; BRANCH IF <HT>
2800 021010 001430      BEQ 8$          ;;
2801 021012 122716 000200      CMPB #CRLF, (SP)      ;; BRANCH IF NOT <CRLF>
2802 021016 001006      BNE 5$          ;;
2803 021020 005726      TST (SP)+      ;; POP <CR><LF> EQUIV
2804 021022 104401      TYPE      ;; TYPE A CR AND LF
2805 021024 001165      $CRLF
2806 021026 105037 021162      CLRB $CHARCNT      ;; CLEAR CHARACTER COUNT
2807 021032 000755      BR 2$          ;; GET NEXT CHARACTER
2808 021034 004737 021116      5$: JSR PC, $TYPEC      ;; GO TYPE THIS CHARACTER
2809 021040 123726 001156      6$: CMPB $FILLC, (SP)+      ;; IS IT TIME FOR FILLER CHARS.?
2810 021044 001350      BNE 2$          ;; IF NO GO GET NEXT CHAR.
2811 021046 013746 001154      MOV $NULL, -(SP)      ;; GET # OF FILLER CHARS. NEEDED
2812      AND THE NULL CHAR.
2813 021052 105366 000001      7$: DECB 1(SP)      ;; DOES A NULL NEED TO BE TYPED?
2814 021056 002770      BLT 6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
2815 021060 004737 021116      JSR PC, $TYPEC      ;; GO TYPE A NULL
2816 021064 105337 021162      DECB $CHARCNT      ;; DO NOT COUNT AS A COUNT

```

```

2817 021070 000770          BR      7$          ;;LOOP
2818
2819          ;HORIZONTAL TAB PROCESSOR
2820
2821 021072 112716 000040      8$:   MOVB   8' (SP)          ;; REPLACE TAB WITH SPACE
2822 021076 004737 021116      9$:   JSR    PC,$TYPEC          ;; TYPE A SPACE
2823 021102 132737 000007 021162      BITB   87,$SCHARCNT          ;; BRANCH IF NOT AT
2824 021110 001372          BNE    9$          ;; TAB STOP
2825 021112 005726          TST    (SP)+          ;; POP SPACE OFF STACK
2826 021114 000724          BR     2$          ;; GET NEXT CHARACTER
2827 021116 105777 160026      $TYPEC: TSTB   2$STPS          ;; WAIT UNTIL PRINTER IS READY
2828 021122 100375          BPL    $TYPEC
2829 021124 116677 000002 160020      MOVB   2(SP),2$STPB          ;; LOAD CHAR TO BE TYPED INTO DATA REG.
2830 021132 122766 000015 000002      CMPB   8CR,2(SP)          ;; IS CHARACTER A CARRIAGE RETURN?
2831 021140 001003          BNE    1$          ;; BRANCH IF NO
2832 021142 105037 021162      CLRB   $SCHARCNT          ;; YES--CLEAR CHARACTER COUNT
2833 021146 000406          BR     $TYPEX          ;; EXIT
2834 021150 122766 000012 000002 1$:   CMPB   8LF,2(SP)          ;; IS CHARACTER A LINE FEED?
2835 021156 001402          BEQ    $TYPEX          ;; BRANCH IF YES
2836 021160 105227          INCB   (PC)+          ;; COUNT THE CHARACTER
2837 021162 000000          $SCHARCNT: .WORD 0          ;; CHARACTER COUNT STORAGE
2838 021164 000207          $TYPEX: RTS    PC
2839
2840          .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2841
2842          ;*****
2843          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2844          ;*CHANGE IT TO BINARY.
2845          ;*CALL:
2846          ;*   RDOCT
2847          ;*   RETURN HERE
2848          ;*
2849          ;*   READ AN OCTAL NUMBER
2850          ;*   LOW ORDER BITS ARE ON TOP OF THE STACK
2851          ;*   HIGH ORDER BITS ARE IN $SHIOCT
2852
2853 021166 011646 000004 000002  $RDOCT: MOV    (SP),-(SP)          ;; PROVIDE SPACE FOR THE
2854 021170 016666          MOV    4(SP),2(SP)          ;; INPUT NUMBER
2855 021176 010046          MOV    R0,-(SP)          ;; PUSH R0 ON STACK
2856 021200 010146          MOV    R1,-(SP)          ;; PUSH R1 ON STACK
2857 021202 010246          MOV    R2,-(SP)          ;; PUSH R2 ON STACK
2858 021204 104411          1$:   RDLIN          ;; READ AN ASCII LINE
2859 021206 012600          MOV    (SP)+,R0          ;; GET ADDRESS OF 1ST CHARACTER
2860 021210 005001          CLR    R1          ;; CLEAR DATA WORD
2861 021212 005002          CLR    R2
2862 021214 112046          2$:   MOVB   (R0)+,-(SP)          ;; PICKUP THIS CHARACTER
2863 021216 001412          BEQ    3$          ;; IF ZERO GET OUT
2864 021220 006301          ASL    R1          ;; *2
2865 021222 006102          ROL    R2
2866 021224 006301          ASL    R1          ;; *4
2867 021226 006102          ROL    R2
2868 021230 006301          ASL    R1          ;; *8
2869 021232 006102          ROL    R2
2870 021234 042716 177770          BIC    8C7,(SP)          ;; STRIP THE ASCII JUNK
2871 021240 062601          ADD    (SP)+,R1          ;; ADD IN THIS DIGIT
2872 021242 000764          BR     2$          ;; LOOP
2873 021244 005726          3$:   TST    (SP)+          ;; CLEAN TERMINATOR FROM STACK

```



```

2871 021246 010166 000012          MOV      R1,12(SP)          ;;SAVE THE RESULT
2872 021252 010237 021266          MOV      R2,$HIOCT
2873 021256 012602                MOV      (SP)+,R2          ;;POP STACK INTO R2
2874 021260 012601                MOV      (SP)+,R1          ;;POP STACK INTO R1
2875 021262 012600                MOV      (SP)+,R0          ;;POP STACK INTO R0
2876 021264 000002                RTI                          ;;RETURN
2877 021266 000000                $HIOCT: WORD 0              ;;HIGH ORDER BITS GO HERE
2878                                     .SBTTL  TTY INPUT ROUTINE
2879
2880                                     ;;*****
2881                                     .ENABL  LSB
2882
2883                                     ;;*****
2884                                     *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2885                                     *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2886                                     *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2887                                     *WHEN OPERATING IN TTY FLAG MODE.
2888 021270 022737 000176 001140 $CKSWR: CMP      $SWREG,SWR          ;;IS THE SOFT-SWR SELECTED?
2889 021276 001074                    BNE     15$                  ;;BRANCH IF NO
2890 021300 105777 157640                    TSTB   $STKS                 ;;CHAR THERE?
2891 021304 100071                    BPL     15$                  ;;IF NO, DON'T WAIT AROUND
2892 021306 117746 157634                    MOVB   $STKB,-(SP)           ;;SAVE THE CHAR
2893 021312 042716 177600                    BIC    $C177,(SP)           ;;STRIP-OFF THE ASCII
2894 021316 022726 000007                    CMP    $7,(SP)+             ;;IS IT A CONTROL G?
2895 021322 001062                    BNE     15$                  ;;NO, RETURN TO USER
2896 021324 123727 001134 000001                    CMPB   $AUTOB,#1           ;;ARE WE RUNNING IN AUTO-MODE?
2897 021332 001456                    BEQ    15$                  ;;BRANCH IF YES
2898
2899 021334 104401 022142          $GTSWR: TYPE    ,SCNTLG          ;;ECHO THE CONTROL-G (↑G)
2900 021340 104401 022147          MOV      $MSWR              ;;TYPE CURRENT CONTENTS
2901 021344 013746 000176          MOV      $WREG,-(SP)         ;;SAVE SWREG FOR TYPEOUT
2902 021350 104402                    TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2903 021352 104401 022160          TYPE    ,SMNEW              ;;PROMPT FOR NEW SWR
2904 021356 005046                    CLR    -(SP)                ;;CLEAR COUNTER
2905 021360 005046                    CLR    -(SP)                ;;THE NEW SWR
2906 021362 105777 157556          7$:    TSTB   $STKS           ;;CHAR THERE?
2907 021366 100375                    BPL    7$                   ;;IF NOT TRY AGAIN
2908
2909 021370 117746 157552          MOVB   $STKB,-(SP)           ;;PICK UP CHAR
2910 021374 042716 177600          BIC    $C177,(SP)           ;;MAKE IT 7-BIT ASCII
2911
2912
2913
2914 021400 021627 000025          9$:    CMP    (SP),#25         ;;IS IT A CONTROL-U?
2915 021404 001005                    BNE    10$                  ;;BRANCH IF NOT
2916 021406 104401 022135          TYPE    ,SCNTLU             ;;YES, ECHO CONTROL-U (↑U)
2917 021412 062706 000006          20$:   ADD    #6,SP           ;;IGNORE PREVIOUS INPUT
2918 021416 000757                    BR     19$                  ;;LET'S TRY IT AGAIN
2919
2920
2921 021420 021627 000015          10$:   CMP    (SP),#15        ;;IS IT A <CR>?
2922 021424 001022                    BNE    16$                  ;;BRANCH IF NO
2923 021426 005766 000004          TST    4(SP)                ;;YES, IS IT THE FIRST CHAR?
2924 021432 001403                    BEQ    11$                  ;;BRANCH IF YES

```

2925	021434	016677	000002	157476		MOV	2(SP),@SWR	:: SAVE NEW SWR
2926	021442	062706	000006		11\$:	ADD	#6,SP	:: CLEAR UP STACK
2927	021446	104401	001165		14\$:	TYPE	\$CRLF	:: ECHO <CR> AND <LF>
2928	021452	123727	001135	000001		CMPB	\$INTAG,#1	:: RE-ENABLE TTY KBD INTERRUPTS?
2929	021460	001003				BNE	15\$	:: BRANCH IF NOT
2930	021462	012777	000100	157454		MOV	#100,@STKS	:: RE-ENABLE TTY KBD INTERRUPTS
2931	021470	000002			15\$:	RTI		:: RETURN
2932	021472	004737	021116		16\$:	JSR	PC,\$TYPEC	:: ECHO CHAR
2933	021476	021627	000060			CMP	(SP),#60	:: CHAR < 0?
2934	021502	002420				BLT	18\$	:: BRANCH IF YES
2935	021504	021627	000067			CMP	(SP),#67	:: CHAR > 7?
2936	021510	003015				BGT	18\$	:: BRANCH IF YES
2937	021512	042726	000060			BIC	#60,(SP)+	:: STRIP-OFF ASCII
2938	021516	005766	000002			TST	2(SP)	:: IS THIS THE FIRST CHAR
2939	021522	001403				BEQ	17\$	:: BRANCH IF YES
2940	021524	006316				ASL	(SP)	:: NO, SHIFT PRESENT
2941	021526	006316				ASL	(SP)	:: CHAR OVER TO MAKE
2942	021530	006316				ASL	(SP)	:: ROOM FOR NEW ONE.
2943	021532	005266	000002		17\$:	INC	2(SP)	:: KEEP COUNT OF CHAR
2944	021536	056616	177776			BIS	-2(SP),(SP)	:: SET IN NEW CHAR
2945	021542	000707				BR	7\$	:: GET THE NEXT ONE
2946	021544	104401	001164		18\$:	TYPE	\$QUES	:: TYPE ?<CR><LF>
2947	021550	000720				BR	20\$	:: SIMULATE CONTROL-U
2948						.DSABL	LSB	

\*\*\*\*\*  
: THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

: \*CALL:  
: \* RDCHR : INPUT A SINGLE CHARACTER FROM THE TTY  
: \* RETURN HERE : CHARACTER IS ON THE STACK  
: \* : WITH PARITY BIT STRIPPED OFF

2959	021552	011646			\$RDCHR:	MOV	(SP),-(SP)	:: PUSH DOWN THE PC
2960	021554	016666	000004	000002		MOV	4(SP),2(SP)	:: SAVE THE PS
2961	021562	105777	157356		1\$:	TSTB	@STKS	:: WAIT FOR
2962	021566	100375				BPL	1\$	:: A CHARACTER
2963	021570	117766	157352	000004		MOVB	@STKB,4(SP)	:: READ THE TTY
2964	021576	042766	177600	000004		BIC	#1C<177>,4(SP)	:: GET RID OF JUNK IF ANY
2965	021604	026627	000004	000023		CMP	4(SP),#23	:: IS IT A CONTROL-S?
2966	021612	001013				BNE	3\$	:: BRANCH IF NO
2967	021614	105777	157324		2\$:	TSTB	@STKS	:: WAIT FOR A CHARACTER
2968	021620	100375				BPL	2\$	:: LOOP UNTIL ITS THERE
2969	021622	117746	157320			MOVB	@STKB,-(SP)	:: GET CHARACTER
2970	021626	042716	177600			BIC	#1C177,(SP)	:: MAKE IT 7-BIT ASCII
2971	021632	022627	000021			CMP	(SP)+,#21	:: IS IT A CONTROL-Q?
2972	021636	001366				BNE	2\$	:: IF NOT DISCARD IT
2973	021640	000750				BR	1\$	:: YES, RESUME
2974	021642	026627	000004	000140	3\$:	CMP	4(SP),#140	:: IS IT UPPER CASE?
2975	021650	002407				BLT	4\$	:: BRANCH IF YES
2976	021652	026627	000004	000175		CMP	4(SP),#175	:: IS IT A SPECIAL CHAR?
2977	021660	003003				BGT	4\$	:: BRANCH IF YES
2978	021662	042766	000040	000004		BIC	#40,4(SP)	:: MAKE IT UPPER CASE



2979	021670	000002		4\$:	RTI	::GO BACK TO USER
2980					*****	*****
2981					*THIS ROUTINE WILL INPUT A STRING FROM THE TTY	
2982					*CALL:	
2983				*	RDLIN	::INPUT A STRING FROM THE TTY
2984				*	RETURN HERE	::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2985				*		::TERMINATOR WILL BE A BYTE OF ALL 0'S
2986						
2987	021672	010346		\$RDLIN:	MOV R3, -(SP)	::SAVE R3
2988	021674	005046			CLR -(SP)	::CLEAR THE RUBOUT KEY
2989	021676	012703	022126	1\$:	MOV #STTYIN, R3	::GET ADDRESS
2990	021702	022703	022135	2\$:	CMP #STTYIN+7, R3	::BUFFER FULL?
2991	021706	101456			4\$	::BR IF YES
2992	021710	104410			BLOS RDCHR	::GO READ ONE CHARACTER FROM THE TTY
2993	021712	112613			MOV (SP)+, (R3)	::GET CHARACTER
2994	021714	122713	000177	10\$:	CMPB #177, (R3)	::IS IT A RUBOUT
2995	021720	001022			BNE 5\$	::BR IF NO
2996	021722	005716			TST (SP)	::IS THIS THE FIRST RUBOUT?
2997	021724	001007			BNE 6\$	::BR IF NO
2998	021726	112737	000134 022124		MOVB #' \, 9\$	::TYPE A BACK SLASH
2999	021734	104401	022124		TYPE 9\$	
3000	021740	012716	177777		MOV #-1, (SP)	::SET THE RUBOUT KEY
3001	021744	005303		6\$:	DEC R3	::BACKUP BY ONE
3002	021746	020327	022126		CMP R3, #STTYIN	::STACK EMPTY?
3003	021752	103434			4\$	::BR IF YES
3004	021754	111337	022124		MOVB (R3), 9\$	::SETUP TO TYPEOUT THE DELETED CHAR.
3005	021760	104401	022124		TYPE 9\$	::GO TYPE
3006	021764	000746			BR 2\$	::GO READ ANOTHER CHAR.
3007	021766	005716		5\$:	TST (SP)	::RUBOUT KEY SET?
3008	021770	001406			BEG 7\$	::BR IF NO
3009	021772	112737	000134 022124		MOVB #' \, 9\$	::TYPE A BACK SLASH
3010	022000	104401	022124		TYPE 9\$	
3011	022004	005016			CLR (SP)	::CLEAR THE RUBOUT KEY
3012	022006	122713	000025	7\$:	CMPB #25, (R3)	::IS CHARACTER A CTRL U?
3013	022012	001003			BNE 8\$	::BR IF NO
3014	022014	104401	022135		TYPE \$CNTLU	::TYPE A CONTROL "U"
3015	022020	000726			BR 1\$	::GO START OVER
3016	022022	122713	000022	8\$:	CMPB #22, (R3)	::IS CHARACTER A "↑R"?
3017	022026	001011			BNE 3\$	::BRANCH IF NO
3018	022030	105013			CLRB (R3)	::CLEAR THE CHARACTER
3019	022032	104401	001165		TYPE , \$CRLF	::TYPE A "CR" & "LF"
3020	022036	104401	022126		TYPE \$TTYIN	::TYPE THE INPUT STRING
3021	022042	000717			BR 2\$	::GO PICKUP ANOTHER CHACTER
3022	022044	104401	001164	4\$:	TYPE \$QUES	::TYPE A '?'
3023	022050	000712			BR 1\$	::CLEAR THE BUFFER AND LOOP
3024	022052	111337	022124	3\$:	MOVB (R3), 9\$	::ECHO THE CHARACTER
3025	022056	104401	022124		TYPE 9\$	
3026	022062	122723	000015		CMPB #15, (R3)+	::CHECK FOR RETURN
3027	022066	001305			BNE 2\$	::LOOP IF NOT RETURN
3028	022070	105063	177777		CLRB -1(R3)	::CLEAR RETURN (THE 15)
3029	022074	104401	001166		TYPE \$LF	::TYPE A LINE FEED
3030	022100	005726			TST (SP)+	::CLEAN RUBOUT KEY FROM THE STACK
3031	022102	012603			MOV (SP)+, R3	::RESTORE R3
3032	022104	011646			MOV (SP), -(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE

```

3033 022106 016666 000004 000002      MOV      4(SP),2(SP)      ;; FIRST ASCII CHARACTER ON IT
3034 022114 012766 022126 000004      MOV      #STTYIN,4(SP)  ;;
3035 022122 000002      RTI                      ;; RETURN
3036 022124 000      9$: .BYTE 0                ;; STORAGE FOR ASCII CHAR. TO TYPE
3037 022125 000      .BYTE 0                ;; TERMINATOR
3038 022126 000007      $TTYIN: .BLKB 7         ;; RESERVE 7 BYTES FOR TTY INPUT
3039 022135 136      006525 000012      $CNTLU: .ASCIZ /↑U/<15><12>  ;; CONTROL "U"
3040 022142 043536 005015 000      $CNTLG: .ASCIZ /↑G/<15><12>  ;; CONTROL "G"
3041 022147 015      051412 051127      $MSWR: .ASCIZ <15><12>/SWR = /
3042 022154 036440 000040      $MNEW: .ASCIZ / NEW = /
3043 022160 020040 042516 020127
3044 022166 020075 000
3045 022172
3046
3047
3048 .SBTTL APT COMMUNICATIONS ROUTINE
3049
3050 ..*****
3051 022172 112737 000001 022436 $ATY1: MOVB #1,$FFLG      ;; TO REPORT FATAL ERROR
3052 022200 112737 000001 022434 $ATY3: MOVB #1,$MFLG      ;; TO TYPE A MESSAGE
3053 022206 000403
3054 022210 112737 000001 022436 $ATY4: MOVB #1,$FFLG      ;; TO ONLY REPORT FATAL ERROR
3055 022216
3056 022216 010046      MOV      R0,-(SP)        ;; PUSH R0 ON STACK
3057 022220 010146      MOV      R1,-(SP)        ;; PUSH R1 ON STACK
3058 022222 105737 022434      TSTB    $MFLG            ;; SHOULD TYPE A MESSAGE?
3059 022226 001450      BEQ     5$               ;; IF NOT: BR
3060 022230 122737 000001 001210      CMPB    #APTENV,$ENV     ;; OPERATING UNDER APT?
3061 022236 001031      BNE     3$               ;; IF NOT: BR
3062 022240 132737 000100 001211      BITB    #APTPOOL,$ENVM   ;; SHOULD SPOOL MESSAGES?
3063 022246 001425      BEQ     3$               ;; IF NOT: BR
3064 022250 017600 000004      MOV      24(SP),R0        ;; GET MESSAGE ADDR.
3065 022254 062766 000002 000004      ADD     #2,4(SP)         ;; BUMP RETURN ADDR.
3066 022262 005737 001170      1$: TST  $MSGTYPE        ;; SEE IF DONE W/ LAST XMISSION?
3067 022266 001375      BNE     1$               ;; IF NOT: WAIT
3068 022270 010037 001204      MOV     R0,$MSGAD        ;; PUT ADDR IN MAILBOX
3069 022274 105720      2$: TSTB (R0)+           ;; FIND END OF MESSAGE
3070 022276 001376      BNE     2$
3071 022300 163700 001204      SUB     $MSGAD,R0        ;; SUB START OF MESSAGE
3072 022304 006200      ASR    R0                ;; GET MESSAGE LNTH IN WORDS
3073 022306 010037 001206      MOV     R0,$MSGGLT       ;; PUT LENGTH IN MAILBOX
3074 022312 012737 000004 001170      MOV     #4,$MSGTYPE      ;; TELL APT TO TAKE MSG.
3075 022320 000413      BR     5$
3076 022322 017637 000004 022346 3$: MOV  24(SP),4$          ;; PUT MSG ADDR IN JSR LINAGE
3077 022330 062766 000002 000004      ADD     #2,4(SP)         ;; BUMP RETURN ADDRESS
3078 022336 013746 177776      MOV     177776,-(SP)     ;; PUSH 177776 ON STACK
3079 022342 004737 020704      JSR    PC,$TYPE         ;; CALL TYPE MACRO
3080 022346 000000      4$: .WORD 0
3081 022350      5$:
3082 022350 105737 022436      10$: TSTB $FFLG          ;; SHOULD REPORT FATAL ERROR?
3083 022354 001416      BEQ    12$              ;; IF NOT: BR
3084 022356 005737 001210      TST    $ENV             ;; RUNNING UNDER APT?
3085 022362 001413      BEQ    12$              ;; IF NOT: BR
3086 022364 005737 001170      11$: TST  $MSGTYPE        ;; FINISHED LAST MESSAGE?

```



```

3087 022370 001375          BNE      115          ;; IF NOT: WAIT
3088 022372 017637 000004 001172  MOV     24(SP), $FATAL ;; GET ERROR #
3089 022400 062766 000002 000004  BDD     24(SP)      ;; BUMP RETURN ADDR.
3090 022406 005237 001170          INC     $MSGTYPE    ;; TELL APT TO TAKE ERROR
3091 022412 105037 022436          12$:   CLR   $FFLG      ;; CLEAR FATAL FLAG
3092 022416 105037 022435          CLR   $LFLG      ;; CLEAR LOG FLAG
3093 022422 105037 022434          CLR   $MFLG      ;; CLEAR MESSAGE FLAG
3094 022426 012601          MOV     (SP)+, R1  ;; POP STACK INTO R1
3095 022430 012600          MOV     (SP)+, R0  ;; POP STACK INTO R0
3096 022432 000207          RTS     PC         ;; RETURN
3097 022434          000          $MFLG: .BYTE 0    ;; MESSG. FLAG
3098 022435          000          $LFLG: .BYTE 0    ;; LOG FLAG
3099 022436          000          $FFLG: .BYTE 0    ;; FATAL FLAG
3100          022440          .EVEN
3101          000200  APTSIZE=200
3102          000001  APTENV=001
3103          000100  APTSPool=100
3104          000040  APTCSUP=040
3105          .SBTTL TRAP DEC0DER
3106
3107          ;; *****
3108          ;; *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3109          ;; *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3110          ;; *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3111          ;; *GO TO THAT ROUTINE.
3112
3113 022440 010046          STRAP:  MOV     RO, -(SP)      ;; SAVE RO
3114 022442 016600 000002  MOV     2(SP), RO          ;; GET TRAP ADDRESS
3115 022446 005740          TST     -(RO)            ;; BACKUP BY 2
3116 022450 111000          MOV     (RO), RO         ;; GET RIGHT BYTE OF TRAP
3117 022452 006300          ASL     RO               ;; POSITION FOR INDEXING
3118 022454 016000 022474  MOV     $TRAPAD(RO), RO  ;; INDEX TO TABLE
3119 022460 000200          RTS     RO              ;; GO TO ROUTINE
3120
3121
3122          ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
3123
3124 022462 011646          STRAP2: MOV     (SP), -(SP)  ;; MOVE THE PC DOWN
3125 022464 016666 000004 000002  MOV     4(SP), 2(SP)     ;; MOVE THE PSW DOWN
3126 022472 000002          RTI                    ;; RESTORE THE PSW
3127
3128          .SBTTL TRAP TABLE
3129
3130          ;; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3131          ;; *BY THE "TRAP" INSTRUCTION.
3132
3133          ;          ROUTINE
3134          ;          -----
3135 022474 022462          $TRPAD: .WORD  $STRAP2          TRAP+1(104401)  TTY TYPEOUT ROUTINE
3136 022476 020704          $TYPE   ;; CALL=TYPE          TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3137 022500 020262          $TYPOC  ;; CALL=TYPOC         TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3138 022502 020236          $TYPOS  ;; CALL=TYPOS         TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3139 022504 020276          $TYPON  ;; CALL=TYPON          TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
3140 022506 013354          $TYPDS  ;; CALL=TYPDS

```

```

3141
3142 022510 021340          SGTSWR  ;;CALL=GTSWR   TRAP+6(104406)  GET SOFT-SWR SETTING
3143
3144 022512 021270          SCKSWR  ;;CALL=CKSWR   TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
3145 022514 021552          $RDCHR  ;;CALL=RDCHR   TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
3146 022516 021672          $RDLIN  ;;CALL=RDLIN   TRAP+11(104411) TTY TYPEIN STRING ROUTINE
3147 022520 021166          $RDOCT  ;;CALL=RDOCT   TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3148
3149 022522 000000          BUFFER: 0                ;10 WORD BUFFER FOR THE COULTER INTERFACE TEST
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167 022524
3168 022524 013746 000004          SLPAI:  MOV      4,-(SP)
3169
3170 022530 000413          BR      31$
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183 022532 012737 022556 000004          MOV      #30$,4
3184 022540 005237 170000          INC      170000
3185 022544 104401 022552          TYPE    65$
3186 022550 000401          BR      64$
3187
3188 022554
3189 022554 000401          ;;65$: .ASCIZ <?>##
3190 022556 022626          64$: BR      31$
3191 022560 012637 000004          30$: CMP      (SP)+,(SP)+
3192 022564 005037 023402          31$: MOV      (SP)+,4
3193 022570 004537 023404          CLR      $AERR
3194 022574 000000G          JSR      R5,$LOAD
          .WORD DRLPX2
          ;LOAD MICRO-CODE.
          ;FILE "DRLPX2.OBJ"

;THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
;FIRST WE WILL LOAD MICROCODE INTO KMC-11
;NEXT WE WILL INIT BOTH UPROCESSORS
;THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
;THE ORDER OF LOAD IS DETERMINED BY THE USER.
;
;CALL= JSR 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.

;FIELD DOES NOT HAVE A BUS SWITCH TO
;WORRY ABOUT, SO WE WILL UNCONDITIONALLY
;BRANCH AROUND THE NEXT CODE THAT
;WORKS BASED ON A BUS SWITCH.
;CODE LEFT IN HERE FOR IN HOUSE
;PERSONAL WHO MAY PATCH THIS BRANCH
;INSTRUCTION TO A <NOP> OCTAL <240>
;IN ORDER TO RUN PROGRAM WITH A SWITCH.

;NOTE THIS "SWITCH" IS A PIECE OF INHOUSE
;TEST EQUIPMENT ONLY IT CONNECTS
;THE UNIBUS TO THE I/O BUS FOR
;CERTAIN TESTING.

;;TYPE ASCIZ STRING
;;GET OVER THE ASCIZ

;ALL THIS JUNK MUST BE REMOVED!!

```



```

3195
3196 022576 052777 040000 156556      BIS      #BIT14,AKMADO      ;ISSUE KMC+DMC INIT.
3197
3198 022604                                1$:
3199
3200 022604 010146      MOV      R1,-(SP)      ;"HANGS" HERE THEN KMC-11 ERROR.
3201 022606 005001      CLR      R1
3202 022610 005201      INC      R1            ;STALL FOR DMC-UP
3203 022612 001376      BNE     2$
3204 022614 012777 104000 156540      MOV      #BIT15!BIT11,AKMADO      ;SET RUN, AND ENABLE ARBITRATION.
3205 022622 105201      25$:  INCB   R1
3206 022624 001376      BNE     25$
3207
3208 022626 032777 000040 156526      BIT      #BITS,AKMADO      ;SLAVE READY? (READING IPBM SR)
3209 022634 001401      BEQ     3$
3210
3211 022636 104000      ERROR      ;FATAL LPA-11 ERROR SLAVE NOT READY.
3212
3213 022640 012777 000004 156520      3$:  MOV      #4,AKMAD2      ;READ FAST PATH
3214 022646                                4$:
3215 022646 004537 024314      JSR     R5,$TOUT      ;-TOUT-CHECK FOR TIMEOUT
3216
3217 022652 104000      ERROR      ;/TIME-OUT ERROR
3218
3219
3220
3221
3222
3223 022654 000774      BR              4$
3224
3225
3226 022656 122777 000377 156502      CMPB   #377,AKMAD2      ;/RETURNS HERE-FROM-TIMED OUT.
3227 022664 001370      BNE     4$            ;WAIT TILL KMC DONE COMMAND.
3228 022666 122777 000377 156476      CMPB   #377,AKMAD4      ;IF FAST PATH=377 THEN ERROR.
3229 022674 001001      BNE     35$
3230 022676 104000      ERROR      ;IPBM ERROR (SLAVE SIDE)
3231
3232
3233 022700 122777 000004 156464      35$:  CMPB   #4,AKMAD4      ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
3234 022706 001543      BEQ     5$            ;YES-CONTINUE.
3235 022710 005227 177777      INC     #-1
3236 022714 001140      BNE     5$
3237 022716 005227 177777      INC     #-1
3238 022722 001135      BNE     5$
3239 022724 104401 022732      TYPE   ,67$          ;;TYPE ASCIZ STRING
3240 022730 000440      BR      66$          ;;GET OVER THE ASCIZ
3241
3242                                ;;67$: .ASCIZ <200>"W A R N I N G THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
3243 023032 104401 023040      66$:  TYPE   ,69$          ;;TYPE ASCIZ STRING
3244 023036 000430      BR      68$          ;;GET OVER THE ASCIZ
3245
3246                                ;;69$: .ASCIZ <200>"MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED."
3247 023120 104401 023126      68$:  TYPE   ,71$          ;;TYPE ASCIZ STRING
3248 023124 000434      BR      70$          ;;GET OVER THE ASCIZ

```

```

3249          ;:71$: .ASCIZ <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
3250          70$:
3251 023216
3252 023216 112737 177777 023350 5$:   MOVB   #0-1,11$   ;DAC CODE FOR SLAVE.
3253 023224 012501          5$:   MOV    (5)+,R1    ;GET NEXT DEVICE ADDR.
3254 023226 021127 000000 6$:   CMP    (R1),#0    ;TERM REACHED?
3255 023232 001444          6$:   BEQ    10$
3256 023234 105237 023350          INCB   11$
3257 023240 113777 023350 156124 7$:   MOVB   11$,@KMA4  ;FIFO DATA
3258 023246 004737 023352          JSR    PC,20$     ;ISSUE SEND
3259 023252 112177 156114          MOVB   (R1)+,@KMA4 ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
3260 023256 004737 023352          JSR    PC,20$     ;ISSUE SEND
3261 023262 112177 156104          MOVB   (R1)+,@KMA4 ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
3262 023266 004737 023352          JSR    PC,20$
3263
3264 023272 032777 000002 156062 7$:   BIT    #BIT1,@KMA0 ;WAIT FOR FIFO DATA
3265 023300 001374          7$:   BNE    7$        ;=1 NO DATA. =0 DATA.
3266 023302 112777 000002 156056 8$:   MOVB   #2,@KMA2  ;READ FIFO.
3267
3268 023310
3269 023310 004537 024314          8$:   JSR    R5,$TOUT  ; -TOUT-CHECK FOR TIMEOUT
3270
3271 023314 104000          ERROR ; /TIME-OUT ERROR
3272          ; /WE FAILED TO COMPLETE
3273          ; /CURRENT OPERATION.
3274          ; /CONTINUES IN THIS LOOP
3275          ; /WOULD MAKE US "HANG" HERE
3276
3277 023316 000774          BR     8$
3278
3279
3280 023320 122777 000377 156040 9$:   CMPB   #377,@KMA2 ; /RETURNS HERE-FROM-TIMED OUT.
3281 023326 001370          9$:   BNE    8$        ; WAIT FOR READ.
3282 023330 105777 156036          9$:   TSTB   @KMA4    ; WAS A ZERO RETURNED?
3283 023334 001734          9$:   BEQ    6$        ; YES GET NEXT ADDR.
3284          ; SLAVE WILL RETURN CODE 0 IF
3285 023336 005237 023402          INC    $AERR     ; DEV PRESENT. ELSE
3286          ; EXIT $AERR=1 IF SLAVE GIVES ERROR.
3287 023342 005041          10$:  CLR    -(1)      ; GET RID OF REFERENCE TO BAD ADDR.
3288 023344 012601          10$:  MOV    (SP)+,R1
3289 023346 000205          10$:  RTS    R5        ; RETURN ALL ADDR. CHECKED.
3290
3291 023350 000000          11$:  .WORD  0        ; HOLDS DAC CODE PLUS OFFSET
3292          ; TO SLAVES ADDR. TABLE.
3293
3294 023352 112777 000003 156006 20$:  MOVB   #3,@KMA2  ; ISSUE FIFO WRITE
3295 023360
3296 023360 004537 024314          21$:  JSR    R5,$TOUT  ; -TOUT-CHECK FOR TIMEOUT
3297
3298 023364 104000          ERROR ; /TIME-OUT ERROR
3299          ; /WE FAILED TO COMPLETE
3300          ; /CURRENT OPERATION.
3301          ; /CONTINUES IN THIS LOOP
3302          ; /WOULD MAKE US "HANG" HERE

```



```

3303
3304 023366 000774 BR 21$
3305
3306
3307 023370 122777 000377 155770 CMPB #377, @KMA02 ;/RETURNS HERE-FROM-TIMED OUT.
3308 023376 001370 BNE 21$ ;KMC CODE WILL RETURN A "377"
3309 023400 000207 RTS PC ;WHEN DONE COMMAND.
3310
3311 023402 000000 $AERR: .WORD 0 ;=0 IF ADDR. LIST OK,=1 IF BAD.
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321 023404 010446 $LOAD: MOV R4, -(SP) ;SAVE R4.
3322 023406 010046 MOV RO, -(SP) ;SAVE RO.
3323 023410 012500 1$: MOV (5)+, RO ;GET PROG. ADDR.
3324 023412 005077 155744 CLR @KMA00 ;CLEAR CSR
3325 023416 005077 155750 CLR @KMA04 ;CLEAR CRAM ADDR.
3326 023422 052777 002000 155732 2$: BIS #2000, @KMA00 ;SELECT CRAM.
3327 023430 012077 155742 MOV (0)+, @KMA06 ;WRITE DATA.
3328 023434 052777 020000 155720 BIS #20000, @KMA00 ;SET CRAM WRITE
3329 023442 005077 155714 CLR @KMA00 ;DISABLE CRAM.
3330 023446 005277 155720 INC @KMA04 ;UPDATE CRAM ADDR.
3331 023452 021027 177777 CMP (0), #-1 ;ALL DONE?
3332 023456 001361 BNE 2$ ;NO LOOP.
3333 023460 005077 155706 CLR @KMA04 ;CLEAR CRAM ADDR.
3334 023464 016500 177776 MOV -2(5), RO ;GET MICRO CODE ADDR.
3335
3336 023470 052777 002000 155664 3$: BIS #2000, @KMA00 ;SELECT CRAM
3337 023476 022077 155674 CMP (RO)+, @KMA06 ;DATA OK?
3338 023502 001013 BNE 5$ ;NO - REPORT AN ERROR.
3339 023504 021027 177777 CMP (0), #-1 ;ALL DONE?
3340 023510 001405 BEQ 4$ ;YES - EXIT
3341 023512 005077 155644 CLR @KMA00 ;NO - DESELECT CRAM.
3342 023516 005277 155650 INC @KMA04 ;UPDATE CRAM ADDR.
3343 023522 000762 BR 3$
3344
3345 023524 012600 4$: MOV (SP)+, RO ;RESTORE RO
3346 023526 012604 MOV (SP)+, R4 ;RESTORE R4
3347 023530 000205 RTS R5 ;EXIT
3348
3349
3350 5$: TST -(5) ;COME HERE ON LOAD ERROR
3351 023534 105204 INCB R4 ;UPDATE ERROR COUNTER.
3352 023536 100324 BPL 1$ ;IF NOT TOO MANY, TRY AGAIN.
3353 023540 000000 HALT ;MICRO CODE LOAD ERROR.
3354
3355 023542 000722 BR 1$ ;KMC-11 FAULT. YOU COULD TRY
3356 ;TO PRESS CONTINUE TO GIVE IT
;ANOTHER CHANCE, BUT I DOUBT

```

```

3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369 023544 010046
3370 023546 012500
3371 023550 052700 000340
3372 023554 004737 024026
3373 023560 010037 023652
3374 023564 010077 155602
3375 023570 112777 000005 155570
3376 023576 004737 024026
3377 023602 011537 023654
3378 023606 112577 155560
3379
3380 023612 112777 000005 155546
3381 023620 004737 024026
3382 023624 111537 023656
3383 023630 112577 155536
3384 023634 112777 000005 155524
3385 023642 004737 024026
3386 023646 012600
3387 023650 000205
3388 023652 000000
3389 023654 000000
3390 023656 000000
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401 023660 010046
3402 023662 012500
3403 023664 052700 000300
3404 023670 004737 024026
3405 023674 110077 155472
3406 023700 112777 000005 155460
3407 023706 004737 024026
3408 023712 010037 024022
3409 023716
3410 023716 004537 024314

```

```

; THAT THAT WOULD WORK. SINCE I'VE
; ALREADY GIVEN IT 177 (OCTAL) CHANCES.
; TRY RUNNING THE KMC-11 DIAGNOSTIC.

; *THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
; *
; *      CALL = JSR      R5,STLKW
; *              .WORD   0          ; OFFSET OF DEVICE ADDR.
; *              .WORD   0          ; DATA TO BE WRITTEN
; *
STLKW: MOV      RD, -(SP)          ; SAVE RD
        MOV      (5)+, RD        ; GET DEVICE OFFSET
        BIS      #340, RD        ; ADD WRITE CODE
        JSR      PC, $LPW        ; WAIT FOR FAST PATH READY
        MOV      RD, W1
        MOV      RD, @KMD4
        MOV      #5, @KMD2        ; ISSUE FAST PATH WRITE
        JSR      PC, $LPW        ; WAIT FOR RDY
        MOV      (5), W2
        MOV      (5)+, @KMD4      ; WRITE LOW BYTE DATA.
        MOV      #5, @KMD2        ; FP WRITE
        JSR      PC, $LPW
        MOV      (5), W3
        MOV      #5, @KMD4        ; WRITE HIGH BYTE
        JSR      PC, $LPW
        MOV      (SP)+, RD
        RTS      R5              ; EXIT DONE.
W1:    0
W2:    0
W3:    0

; *THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
; *
; *      CALL = JSR      R5,STLKR
; *              .WORD   0          ; OFFSET OF DEVICE
; *              .WORD   0          ; RETURNS HERE
; * *DATA IN WORD $DATR
; *
STLKR: MOV      RD, -(SP)          ; SAVE RD
        MOV      (5)+, RD        ; GET OFFSET
        BIS      #300, RD        ; ADD READ CODE
        JSR      PC, $LPW        ; WAIT TILL READY
        MOV      RD, @KMD4
        MOV      #5, @KMD2        ; ISSUE WRITE FP
        JSR      PC, $LPW
        MOV      RD, RD1
        JSR      R5, $TOUT        ; -TOUT-CHECK FOR TIMEOUT

```



```

3411 023722 104000 ERROR ;/TIME-OUT ERROR
3412 ;/WE FAILED TO COMPLETE
3413 ;/CURRENT OPERATION.
3414 ;/CONTINUES IN THIS LOOP
3415 ;/WOULD MAKE US "HANG" HERE
3416
3417
3418 023724 000774 BR 1$
3419
3420 ;/RETURNS HERE-FROM-TIMED OUT.
3421 023726 032777 000040 155426 BIT #BITS,@KMADO ;FAST PATH GOT DATA?
3422 023734 001370 BNE 1$
3423 023736 112777 000004 155422 MOVB #4,@KMAD2 ;ISSUE FAST PATH READ
3424 023744 004737 024026 JSR PC,$LPW
3425 023750 117737 155416 024024 MOVB @KMAD4,$DATR ;GET LOW BYTE
3426 2$:
3427 023756 004537 024314 JSR R5,$STOUT ;-TOU1-CHECK FOR TIMEOUT
3428
3429 023762 104000 ERROR ;/TIME-OUT ERROR
3430 ;/WE FAILED TO COMPLETE
3431 ;/CURRENT OPERATION.
3432 ;/CONTINUES IN THIS LOOP
3433 ;/WOULD MAKE US "HANG" HERE
3434
3435 023764 000774 BR 2$
3436
3437 ;/RETURNS HERE-FROM-TIMED OUT.
3438 023766 032777 000040 155366 BIT #BITS,@KMADO ;FAST PATH READY?
3439 023774 001370 BNE 2$
3440 023776 112777 000004 155362 MOVB #4,@KMAD2 ;ISSUE FAST PATH READ
3441 024004 004737 024026 JSR PC,$LPW
3442 024010 117737 155356 024025 MOVB @KMAD4,$DATR+1 ;SAVE HIGH BYTE
3443 024016 012600 MOV (SP)+,R0
3444 024020 000205 RTS R5
3445 024022 000000 RD1: 0
3446 024024 000000 $DATR: .WORD 0
3447
3448 ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
3449 ;AS FAST PATH TO BE READ.
3450
3451 ;CALL = JSR PC,$LPW
3452
3453 ;IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
3454 ;THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
3455
3456
3457 024026 010146 SLPW: MOV R1,-(SP) ;SAVE R1
3458 024030 005001 CLR R1
3459 024032 122777 000377 155326 1$: CMPB #377,@KMAD2 ;FINISHED INSTRUCTION?
3460 024040 001403 BEQ 2$
3461 024042 005201 INC R1 ;TIME OUT?
3462 024044 001372 BNE 1$
3463 024046 000411 BR 10$
3464

```

```

3465 024050 032777 000020 155304 2$: BIT #BIT4,AKMADO ;FAST PATH READ?
3466 024056 001403 BEQ 3$ ;
3467 024060 005201 INC R1 ;NO - TIME OUT?
3468 024062 001372 BNE 2$ ;
3469 024064 000402 BR 10$ ;YES - REPORT AN ERROR
3470
3471 024066 012601 3$: MOV (SP)+,R1 ;RESTORE R1
3472 024070 000207 RTS PC ;EXIT
3473
3474 024072 10$:
3475 024072 104401 024100 TYPE 65$ ;:TYPE ASCIZ STRING
3476 024076 000407 BR 64$ ;:GET OVER THE ASCIZ
3477 ;:65$: .ASCIZ <200>#LPA-11 FAULT#
3478 64$:
3479
3480 024116 000000 11$: HALT ;LPA-11 FAULT RUN LPA-11
3481 024120 000776 BR 11$ ;DIAGNOSTICS.
3482
3483
3484
3485
3486 ;*
3487 ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
3488 ;*A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.
3489 ;*
3490 ;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED
3491 ;* BEFORE. IF NOT WE HAVE TO INITIALIZE THE LPA WITH
3492 ;* THAT ADDRESS.
3493 ;* WHEN THE ADDR. IS KNOWN BY THE LPA, DO THE OUTPUT BY
3494 ;* STLKW
3495 ;*
3496 024122 010046 $OUTLP: MOV R0,-(SP) ;SAVE R0
3497 024124 010146 MOV R1,-(SP) ;SAVE R1
3498
3499 024126 012700 001410 MOV #.DVLS,R0 ;PROGRAM DEFINED LIST.
3500 024132 005001 CLR R1
3501 024134 005710 1$: TST (0) ;TERMINATOR REACHED?
3502 024136 001421 BEQ 10$ ;YES NEXT STEP.
3503 024140 027520 000000 CMP @ (5), (0)+ ;MATCH WITH ADDR IN LIST?
3504 024144 001402 BEQ 2$
3505 024146 005201 INC R1
3506 024150 000771 BR 1$
3507
3508 024152 010137 024170 2$: MOV R1,3$ ;SAVE OFFSET, DEVICE KNOWN.
3509 024156 005725 TST (5)+
3510 024160 013537 024172 MOV @ (5)+,4$ ;GET DATA TO BE WRITTEN
3511 024164 004537 023544 JSR R5,STLKW ;DO WRITE
3512 024170 000000 3$: .WORD 0 ;DEVICE OFFSET
3513 024172 000000 4$: .WORD 0 ;DATA TO BE WRITTEN.
3514 024174 012601 MOV (SP)+,R1
3515 024176 012600 MOV (SP)+,R0
3516 024200 000205 RTS R5
3517 024202 017520 000000 10$: MOV @ (5), (0)+ ;SAVE ADDR.
3518 024206 005010 CLR (0)

```



```

3519 024210 004537 022524 JSR R5,$LPAI
3520 024214 001410 .WORD .DVL5
3521 024216 000755 BR 2$
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538 024220 010046 $INLP: MOV R0,-(SP) ;SAVE R0
3539 024222 010146 MOV R1,-(SP) ;SAVE R1
3540
3541 024224 012700 001410 MOV #.DVL5,R0 ;PROG DEFINED ADDR. LIST.
3542 024230 005001 CLR R1
3543 024232 005710 1$: TST (0) ;EOL REACHED?
3544 024234 001420 BEQ 10$ ;YES - DEFINE NEW ADDR.
3545
3546 024236 027520 000000 CMP @($), (0)+ ;ADDR. MATCH?
3547 024242 001402 BEQ 2$
3548 024244 005201 INC R1
3549 024246 000771 BR 1$
3550
3551 024250 010137 024262 2$: MOV R1,3$ ;SAVE LIST OFFSET
3552 024254 005725 TST ($)+
3553 024256 004537 023660 JSR R5,$TLK r ;GO READ DEVICE
3554 024262 000000 $OFS=. 3$: .WORD 0 ;OFFSET OF DEVICE
3555
3556 024264 013735 024024 MOV $DATA,@($)+ ;STORE DATA.
3557 024270 012601 MOV (SP)+,R1 ;RESTORE R1
3558 024272 012600 MOV (SP)+,R0 ;RESTORE R2
3559 024274 000205 RTS R5 ;EXIT
3560
3561
3562 024276 017520 000000 10$: MOV @($), (0)+
3563 024302 005010 CLR (0)
3564 024304 004537 022524 JSR R5,$LPAI
3565 024310 001410 .WORD .DVL5
3566 024312 000755 BR 2$
3567
3568
3569
3570
3571
3572

```

;\* 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  
 ;\* USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA  
 ;\* WITH THE NEW ADDR.  
 ;\* WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH  
 ;\* \$TLKR  
 ;\* CALL THROUGH MOVEI DATA, ADDR.  
 ;\* WHICH EQUALS:  
 ;\* JSR R5,\$INLP  
 ;\* .WORD XX ADDR OF DEVICE  
 ;\* .WORD YY ADDR TO STORE READ DATA.

;\* \$STOUT ROUTINE USED TO WATCH IF  
 ;\* WE'RE IN A LOOP TOO-LONG  
 ;\* CALL= JSR R5, \$STOUT  
 ;\* ERROR X ; RETURNS HERE ON TIMEOUT  
 ;\* BR





DR11-K LOGIC TEST MAINDEC-11-DRLPF-A  
 DRLPF.P11 TRAP TABLE

3627	024412		
3628	024412	005737	024474
3629	024416	100016	
3630	024420	012737	000002 024464
3631	024426	052777	000115 000040
3632	024434	005037	177776
3633	024440	005737	024464
3634	024444	001375	
3635	024446	005077	000022
3636			
3637	024452	000207	
3638	024454	105237	024464
3639	024460	001375	
3640	024462	000207	
3641			
3642	024464	000000	
3643			
3644	024466	005337	024464
3645	024472	000002	
3646	024474	000000	
3647			
3648			
3649			
3650			
3651			
3652			
3653			
3654			
3655			
3656			
3657			
3658	024476		
3659	024476	005037	001410
3660	024502		
3661	024502	104401	024510
3662	024506	000405	
3663			
3664	024522		
3665	024522	105777	154416
3666	024526	100375	
3667	024530	117737	154412 024652
3668	024536	104401	024652
3669	024542	142737	000240 024652
3670	024550	104412	
3671	024552	012637	024650
3672	024556	123727	024652 000104
3673	024564	001411	
3674			
3675	024566	004537	024220
3676	024572	024650	
3677	024574	024606	
3678			
3679	024576	013746	024606
3680	024602	104402	

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

10\$: RTS PC  
 INCB TIME  
 BNE 10\$  
 RTS PC

TIME: .WORD 0

CLKINT: DEC TIME  
 RTI

RTCCSR: .WORD 0 ;CLOCK CSR IF USED.

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

SUTK: CLR .DVLS

21\$: TYPE 65\$ ;:TYPE ASCIZ STRING  
 BR 64\$ ;:GET OVER THE ASCIZ  
 ;:65\$: .ASCIZ <200>#E OR D?#  
 ;:64\$:

1\$: TSTB STKS  
 BPL 1\$  
 MOVB STKB,20\$ ;GET INPUT  
 TYPE, 20\$ ;ECHO NEXT MESSAGE.  
 BICB #240,20\$ ;STRIP PARITY, LC  
 RDOCT ;GET ADDR.  
 MOV (SP)+,14\$  
 CMPB 20\$,#D ;DEPOSIT?  
 BEQ 10\$

2\$: JSR R5,\$INLP ;GET DATA  
 .WORD 14\$  
 .WORD 5\$

MOV 5\$,-(SP) ;:SAVE 5\$ FOR TYPEOUT  
 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)

```

3681 024604 000736          BR      21$          ;LOOP.
3682 024606 000000          5$:      .WORD      0
3683
3684 024610          10$:     TYPE      ,67$          ;;TYPE ASCIZ STRING
3685 024610 104401 024616          BR      66$          ;;GET OVER THE ASCIZ
3686 024614 000404          ;;67$:   .ASCIZ   <200>#DATA= #
3687
3688 024626          66$:     RDOCT
3689 024626 104412          MOV      (SP)+,13$
3690 024630 012637 024646
3691
3692 024634 004537 024122          11$:     JSR      R5,$OUTLP          ;OUTPUT ROUTINE.
3693 024640 024650          12$:     .WORD      14$          ;DEVICE ADDR.
3694 024642 024646          .WORD      13$          ;DATA
3695 024644 000716          BR      21$
3696
3697 024646 000000          13$:     .WORD      0
3698 024650 000000          14$:     .WORD      0
3699 024652 100001 042504 044526          20$:     .ASCIZ   <1><200>#DEVICE ADDR= #
3700 024660 042503 040440 042104
3701 024666 036522 000040
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726 024672 012537 024702          .EVEN
3727 024676 004537 024220
3728 024702 000000
3729 024704 025000
3730 024706 113777 024262 154462
3731 024714 113777 024262 154456
3732 024722 013737 024702 024742
3733 024730 062737 000002 024742
3734 024736 004537 024220

```

```

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

```

```

SPUTS: MOV      (5)+,1$          ;GET ADDR OF ADDR. OF A/D
1$:     JSR      R5,$INLP
      .WORD      0
      .WORD      10$
      MOVVB     $OFS,$KMA6
      MOVVB     $OFS,$KMA7
      MOV      1$,2$
      ADD      #2,2$
      JSR      R5,$INLP

```



3735	024742	000000			2S:	.WORD	0
3736	024744	025000				.WORD	10S
3737	024746	113777	024262	154414		MOVB	\$0FS, @KMAD3
3738	024754	152777	000340	154414		BISB	#340, @KMAD6
3739	024762	152777	000300	154410		BISB	#300, @KMAD7
3740	024770	152777	000300	154372		BISB	#300, @KMAD3
3741	024776	000205				RTS	R5
3742	025000	000000			10S:	.WORD	0
3743							
3744		000001			.END		

ABASE = 167770	60#	282	323			
ACDW1 = 000000	282	325				
ACDW2 = 000000	282					
ACPUOP = 000000	282	297				
ADBR = 015466	2244	2279#				
ADCS = 015462	2223	2226	2232	2235	2240	2277#
ADCS1 = 015464	2218	2221	2278#			
ADDW0 = 000000	282					
ADDW1 = 000000	282					
ADDW10 = 000000	282					
ADDW11 = 000000	282					
ADDW12 = 000000	282					
ADDW13 = 000000	282					
ADDW14 = 000000	282					
ADDW15 = 000000	282					
ADDW2 = 000000	282					
ADDW3 = 000000	282					
ADDW4 = 000000	282					
ADDW5 = 000000	282					
ADDW6 = 000000	282					
ADDW7 = 000000	282					
ADDW8 = 000000	282					
ADDW9 = 000000	282					
ADEVCT = 000000	282	288				
ADEVN = 000000	282	324				
AREN = 000000	282	293				
AREN = 000000	282	294				
AFATAL = 000000	282	285				
AMADR1 = 000000	282	310				
AMADR2 = 000000	282	314				
AMADR3 = 000000	282	317				
AMADR4 = 000000	282	320				
AMAMS1 = 000000	282	304				
AMAMS2 = 000000	282	312				
AMAMS3 = 000000	282	315				
AMAMS4 = 000000	282	318				
AMSGAD = 000000	282	290				
AMSGLC = 000000	282	291				
AMSGTY = 000000	282	284				
AMTYP1 = 000000	282	305				
AMTYP2 = 000000	282	313				
AMTYP3 = 000000	282	316				
AMTYP4 = 000000	282	319				
APASS = 000000	282	287				
APRIOR = 000000	282					
APTCSU = 000040	2791	3104#				
APTENV = 000001	2560	2784	3060	3102#		
APTSIZ = 000200	520	3101#				
APTSP0 = 000100	2786	3062	3103#			
ASWREG = 000000	282	295				
AATESTN = 000000	282	286				
AUNIT = 000000	282	289				
AUSWR = 000000	282	296				
AVECT1 = 100300	61#	282	321	423	424	









GRSTAT	001500	447#	726*	758	999	1001	1012	1015	1017	1028	1030	1041	1044	1046
		1678	1683	1697	1705	1734	1750*	1753	1756	1774*	1775	1804	1825	2011
		2018												
GTSWR =	104406	596	3142#											
HT =	000011	70#	2799	2840										
H322J	001570	200	476#											
INADR	014432	2063	2066	2069	2087#	2137	2140	2143						
INADRH	015531	2090	2286#											
INDADR	015574	2093	2292#											
INTBIT	001512	452#	613*	620*	629	1723	1793							
IOTEST	003332	576	616	623	653	710#	2747							
IOTST1	003412	723#												
IOTTS1	003552	725	741#											
IOTVEC=	000020	165#	489*	490*										
JUMPER	001542	464#	471*	473*	476*	610	617							
KMADO	001362	406#	530	3196*	3204*	3208	3264	3324*	3326*	3328*	3329*	3336*	3341*	3421
		3438	3465											
KMAD1	001364	409#	531											
KMAD2	001366	411#	3213*	3226	3266*	3280	3294*	3307	3375*	3380*	3384*	3406*	3423*	3440*
		3459												
KMAD3	001370	413#	3737*	3740*										
KMAD4	001372	415#	3228	3233	3257*	3259*	3261*	3282	3325*	3330*	3333*	3342*	3374*	3378*
		3383#	3405*	3425	3442									
KMAD5	001374	417#												
KMAD6	001376	419#	3327*	3337	3730*	3738*								
KMAD7	001400	421#	535	3731*	3739*									
LF =	000012	71#	2834	2840										
LOCA	014364	2065	2068	2071	2073#	2085								
LOCBOX	014312	201	2060#											
LOCHDR	015470	2062	2280#											
LOC1	001472	444#	2064	2076	2138	2152	2156	2160	2164	2168				
LOC1Y	014616	2064	2073	2129#	2138	2147								
LOC2	001474	445#	2067	2080	2141	2174	2178	2182	2186	2190				
LOC2Y	014620	2067	2077	2130#	2141	2169								
LOC3	001476	446#	2070	2084	2144	2196	2200	2204	2208	2212				
LOC3Y	014622	2070	2081	2131#	2144	2191								
LOUP1	014544	2075	2079	2083	2112#									
LPADH	001374	416#												
LPADL	001372	414#												
LPCI	001362	405#												
LPCO	001366	410#												
LPMR	001364	408#												
LPMS1	001376	418#												
LPMS2	001400	420#												
LPSO	001370	412#												
MINSIN	001536	462#	614*	621*	632	639	743	1169						
MSGRUS	015730	2042#	2056*	2057	2309#									
MSPNT1	015737	2043	2311#											
NBEXT	001464	438#	713*	1839	1845*	1849*								
NMBEXT	001462	437#	568*	603	713	1849								
NOTLCH	001510	451#	612*	619*	626	704	1150	1195	1200	1246	1284	1290	1318	1498
ODDJMP	001534	461#	742*	743*	744*	1073	1089	1094	1110	1115	1131	1136	1145	1148
		1198	1248	1286	1319	1497	1665							
PC =	%000007	91#	571*	710*	714*	723*	724*	740*	785*	1662*	1869*	1872*	1882*	1887

	1963*	1999*	2006*	2007*	2037*	2058*	2061*	2135*	2557*	2563*	2621*	2746	2789*
	2808*	2815*	2822*	2836*	2838*	2932*	3079*	3096*	3258*	3260*	3262*	3309*	3372*
	3376*	3381*	3385*	3404*	3407*	3424*	3441*	3472*	3610*	3637*	3640*		
PIRQ = 177772	77#												
PIRQVE = 000240	171#												
PRO = 000000	94#												
PR1 = 000040	95#												
PR2 = 000100	96#												
PR3 = 000140	97#												
PR4 = 000200	98#												
PR5 = 000240	99#												
PR6 = 000300	100#												
PR7 = 000340	101#												
PS = 177776	74#	75	3632*										
PSW = 177776	75#												
PWRMSG 020636	2745	2752#											
PWRVEC = 000024	166#	495*	496*	1996*	1997*	2714*	2715*	2724*	2730*	2742*	2743*		
RBEG 001606	472	475	478	480#									
RBEG1 002300	571#												
RBEG2 003360	684	714#	1852										
RDCHR = 104410	2992	3145#											
RDL IN = 104411	2094	2855	3146#										
RDOCT = 104412	696	3147#	3670	3689									
RD1 024022	3408#	3445#											
RESVEC = 000010	161#												
RTCCSR 024474	3628	3631*	3635*	3646#									
RUNMSG 015710	2012	2306#											
RO = %000000	82#	470*	474*	477*	479*	528	530*	533*	534	539*	574	1320*	1330
	1341*	1344	1355*	1358	1369*	1372	1383*	1386	1397*	1400	1411*	1414	1425*
	1428	1439*	1442	1453*	1456	1467*	1470	1481*	1499*	1507	1518*	1519	1530*
	1531	1542*	1543	1554*	1555	1566*	1567	1578*	1579	1590*	1591	1602*	1603
	1614*	1615	1626*	1627	1638*	1702*	1851*	1879*	1882	1906	1916*	1920	1936
	1937	1950*	2021*	2022*	2028*	2029	2033*	2036*	2046*	2047*	2048*	2049*	2050
	2119*	2121*	2123	2244*	2245*	2247*	2249*	2251*	2253*	2255*	2264*	2266*	2593
	2594*	2595*	2602*	2603*	2604*	2605*	2606*	2607	2612	2617*	2619*	2623	2625
	2716	2741*	2782	2783*	2788	2793	2796*	2852	2856*	2859	2875*	3056	3064*
	3068	3069	3071*	3072*	3073	3095*	3113	3114*	3115	3116*	3117*	3118*	3119*
	3322	3323*	3334*	3337	3345*	3369	3370*	3371*	3373	3374	3386*	3401	3402*
	3403*	3405	3408	3443*	3496	3499*	3515*	3538	3541*	3559*			
R1 = %000001	83#	529	531*	535	538*	715*	717*	718*	1338*	1352*	1366*	1380*	1394*
	1408*	1422*	1436*	1450*	1464*	1478*	1483	1515*	1527*	1539*	1551*	1563*	1575*
	1587*	1599*	1611*	1623*	1635*	1640	1907	1920*	1921	1925	1949*	2013*	2029*
	2035	2050*	2051*	2052*	2053	2120*	2121	2235*	2258*	2259*	2260*	2262*	2263*
	2265*	2268*	2270	2271*	22717	2740*	2853	2857*	2861*	2863*	2865*	2868*	2871
	2874*	3057	3094*	3200	3201*	3202*	3205*	3253*	3254	3259	3261	3288*	3457
	3458*	3461*	3467*	3471*	3497	3500*	3505*	3508	3514*	3539	3542*	3548*	3551
	3558*												
R2 = %000002	84#	546*	548	553*	716*	717	719	720	1908	1919*	1923*	1926	1933*
	1934*	1935	1940*	1948*	2043*	2053*	2115*	2116*	2117	2119	2122	2124*	2718
	2739*	2854	2858*	2862*	2864*	2866*	2872	2873*					
R3 = %000003	85#	547*	554*	556	567*	568	1909	1917*	1918*	1932*	1935*	1944*	1945*
	1947*	2035*	2036	2038	2117*	2118*	2120	2122*	2123*	2665	2674*	2680*	2681*
	2684*	2689*	2690*	2691	2700*	2719	2738*	2987	2989*	2990	2993*	2994	3001*
	3002	3004	3012	3016	3018*	3024	3026	3028*	3031*				





















	980	990*	993*	996	1009*	1012	1038*	1041	1061*	1064*	1067	1068*	1071	
	1082*	1085*	1088	1103*	1106*	1109	1124*	1127*	1130	1154*	1157	1159*	1162	
	1171*	1174	1176*	1179	1218*	1221	1239*	1242*	1245	1253*	1256	1265*	1268	
	1277*	1280*	1283	1294*	1297	1307*	1310	1321*	1324*	1327	1330*	1333*	1336	
	1338	1341	1344*	1347*	1350	1352	1355	1358*	1361*	1364	1366	1369	1372*	
	1375*	1378	1380	1383	1386*	1389*	1392	1394	1397	1400*	1403*	1406	1408	
	1411	1414*	1417*	1420	1422	1425	1428*	1431*	1434	1436	1439	1442*	1445*	
	1448	1450	1453	1456*	1459*	1462	1464	1467	1470*	1473*	1476	1478	1481	
	1500*	1503*	1506	1507*	1510*	1513	1515	1518	1519*	1522*	1525	1527	1530	
	1531*	1534*	1537	1539	1542	1543*	1546*	1549	1551	1554	1555*	1558*	1561	
	1563	1566	1567*	1570*	1573	1575	1578	1579*	1582*	1585	1587	1590	1591*	
	1594*	1597	1599	1602	1603*	1606*	1609	1611	1614	1615*	1618*	1621	1623	
	1626	1627*	1630*	1633	1635	1638	1652*	1655*	1658*	1661	1675*	1678	1680	
	1694*	1697	1699	1702	1725*	1728*	1731*	1734	1736*	1739	1741*	1744	1746*	
	1749	1753	1756	1765*	1768	1770*	1773	1795*	1798*	1801*	1804	1806*	1809*	
	1812	1814*	1817	1819*	1822	1825	1966*	1969*	1972	1976*	1979*	1982	2008*	
	2011	2018	2028	2030*	2033	2218*	2221	2223*	2226	2232*	2235	2240	2244	
STN = 000037	30*	59	749	753*	763	767*	774	776	780*	789	792	796*	803	
	806	810*	817	820	824*	837	841*	858	862*	881	885*	931	933	
	937*	983	986	990*	1002	1005	1009*	1018	1021	1025*	1031	1034	1038*	
	1047	1052	1056*	1075	1078	1082*	1096	1099	1103*	1117	1120	1124*	1138	
	1140	1144*	1146	1188	1192*	1233	1237*	1271	1275*	1312	1316*	1488	1491	
	1495*	1645	1647	1651*	1667	1670	1674*	1684	1689	1693*	1706	1711	1715*	
	1716	1781	1785*	1786	1834	1838*								
STOUT	024314	3215	3269	3296	3410	3427								
STPB	001152	268*	2829*	2840										
STPFLG	001157	272*	2778	2840										
STPS	001150	267*	2827	2840										
STRAP	022440	493	1994	3113*										
STRAP2	022462	3124*	3135											
STRP =	000013	3128*	3137*	3138*	3139*	3140*	3141*	3142	3143*	3144	3145*	3146*	3147*	3148*
STRPAD	022474	3118	3135*											
STSTM	001004	232*												
STSNM	001102	245*	754*	1058*	1865*	2473	2500	2522*	2523	2528	2532	2547	2550	2583
STTYIN	022126	2989	2990	3002	3020	3034	3038*							
STY <sub>BN</sub> =	***** U	3141												
STY <sub>PDS</sub>	013354	1905*	3140											
STY <sub>PE</sub>	020704	2778*	3079	3128	3136									
STY <sub>PEC</sub>	021116	2808	2815	2822	2827*	2828	2932							
STY <sub>PEX</sub>	021164	2833	2835	2838*										
STY <sub>OC</sub>	020262	2662*	3137											
STY <sub>PON</sub>	020276	2661	2664*	3139										
STY <sub>POS</sub>	020236	2657*	3138											
SUNIT	001202	289*												
SUNITM	001010	234*												
SUSWR	001214	296*												
SUTK	024476	3658*												
SVECT1	001240	321*	541	543										
SVECT2	001242	322*												
SXTSTR	017442	2487*												
SSGET4=	000000	1881*												
SOFILL	020461	2658*	2662*	2672	2707*									
S4OCAT=	***** U	2484	2557											
.	= 025002	187*	191*	207	208*	210*	212*	213*	219	220*	222*	224*	242*	278





DRLPF.P11 CROSS REFERENCE TABLE

ADJXL	2147# 2209	2149	2153	2157	2161	2165	2171	2175	2179	2183	2187	2193	2197	2201	2205
CLERIN	1051#	1064	1085	1106	1127	1280	1324	1503	1655	1728	1798				
CLEROT	1050# 1417 1618	1060 1431 1630	1082 1445 1652	1103 1459 1725	1124 1473 1795	1253 1500 1965	1277 1510 1976	1294 1522	1321 1534	1333 1546	1347 1558	1361 1570	1375 1582	1389 1594	1403 1606
CLRVCT	467#														
COMMEN	172#														
ENDCOM	172#														
ERROR	66# 1048 1707	643 1076 1758	707 1097 1777	775 1118 1827	790 1139 3211	804 1166 3217	818 1183 3230	834 1210 3271	854 1227 3298	876 1260 3412	932 1302 3429	984 1489	1003 1646	1019 1668	1032 1685
ESCAPE	172#														
GETPRI	172#														
GETSWR	30#	172#	589#												
MOVEI	20# 918 1044 1304 1549 1754 3599	549# 923 1071 1336 1561 1762	771# 928 1092 1350 1573 1768	786 940 1113 1364 1585 1804	800 945 1134 1378 1597 1812	814 950 1157 1392 1609 1823	830 955 1162 1406 1621 1972	850 960 1174 1420 1633 1982	871 965 1179 1434 1663 2015	888 970 1205 1448 1681 2026	893 975 1215 1462 1700 2215	898 980 1221 1476 1703 2221	903 999 1256 1513 1734 2230	908 1015 1262 1525 1739 2237	913 1028 1278 1537 1744 2242
MOVEM	19# 896 978 1104 1251 1345 1398 1451 1523 1583 1656 1796 2031	756 901 991 1107 1254 1348 1401 1454 1528 1588 1659 1799 2219	758 906 994 1111 1266 1353 1404 1457 1532 1592 1676 1802 2224	760 911 997 1125 1278 1356 1409 1460 1535 1595 1678 1807 2233	769 916 1010 1128 1281 1359 1412 1465 1540 1600 1695 1810	783 921 1013 1132 1292 1362 1415 1468 1544 1604 1697 1815	798 926 1026 1152 1295 1367 1418 1471 1547 1607 1726 1817	812 938 1039 1155 1308 1370 1423 1474 1552 1612 1729 1820	827 943 1042 1160 1322 1373 1426 1479 1556 1616 1732 1967	845 948 1062 1172 1325 1376 1429 1479 1559 1619 1737 1970	848 953 1065 1177 1328 1381 1432 1501 1564 1624 1742 1974	866 958 1069 1203 1331 1384 1437 1508 1568 1628 1747 1977	869 963 1083 1219 1334 1387 1440 1511 1571 1631 1751 1980	886 968 1086 1240 1339 1390 1443 1516 1576 1636 1766 1984	891 973 1090 1243 1342 1395 1446 1520 1580 1653 1771 2009
MULT	172#														
NEWTST	172# 1052 1834	749 1078	763 1099	776 1120	792 1140	806 1188	820 1233	837 1271	858 1312	881 1491	932 1647	986 1670	1005 1689	1021 1710	1034 1781
POP	172#	1946	2735	2736	2873	3094	3095								
PUSH	172#	1905	2716	2722	2852	3055	3057	3078							
REPORT	172#														
SCOPE	67# 1055 1837	752 1081 1864	766 1102	779 1123	795 1143	809 1191	823 1236	840 1274	861 1315	884 1494	936 1650	989 1673	1008 1692	1024 1714	1037 1784
SETPRI	172#														
SETTRA	3128#	3137	3138	3139	3140	3142	3144	3145	3146	3147					
SETUP	172#	481													
SKIP	172# 1138 1786	774 1146 1794	789 1209 1826	803 1226	817 1259	833 1301	931 1488	983 1645	1002 1667	1018 1684	1031 1706	1047 1716	1075 1724	1096 1757	1117 1776
SLASH	172#														
SPACE	172#														
STARS	172#	205	216	218	225	238	278	281	749	751	763	765	776	778	792





DRLPF.P11 CROSS REFERENCE TABLE		
.SPOWE	30#	2710
.SRDOC	30#	2840
.SREAD	30#	2878
.SSAVE	30#	
.SSCOP	30#	2468
.SSPAC	30#	
.SSWDO	30#	
.STLKW	26#	3362
.STOUT	431#	3567
.STRAP	30#	3105
.STYPD	30#	1893
.STYPE	30#	2761
.STYPO	30#	2632

CROSS REFERENCE TABLE

ADD	728	730	732	735	737	739	1844	1925	2116	2118	2123	2236	2259	2263	2606	
ASL	2660	2670	2797	2868	2917	2926	3065	3077	3089	3583	3603	3733	2861	2863	2865	2940
ASLB	856	878	1230	1268	1310	1779	1832	2265	2603	2604	2605	2605	2861	2863	2865	2940
ASR	2941	2942	3117													
BCC	1930															
BEQ	2268	3072	1931													
BGE	648	1187	575	587	593	640	642	680	774	789	803	817	833	853	875	
BGT	521	562	1031	1047	1057	1075	1096	1117	1138	1146	1149	1165	1182	1201	1209	
BHI	931	983	1301	1468	1645	1667	1684	1724	1840	1880	2039	2074	2078	2082	2099	
BIC	1226	1259	2170	2192	2229	2499	2501	2503	2507	2516	2549	2572	2575	2608	2613	
BICB	2101	2148	2787	2800	2835	2860	2897	2924	2939	3008	3059	3063	3083	3085	3209	
BIS	2626	2687	3283	3340	3460	3466	3502	3504	3544	3547	3577	3673				
BISB	3234	3255														
BIT	2519	1939	2694	2936	2977											
BITB	1871	1939	2694	2936	2977											
BLO	661	2505														
BLOS	542	544	678	868	1073	1089	1094	1110	1115	1131	1136	1159	1207	1218	1223	
BLT	1290	1291	1318	1319	1497	1498	1665	1741	1765	1809	1868	2033	2051	2098	2121	
BMI	2122	2269	2684	2867	2893	2910	2937	2964	2970	2978						
BNE	3669															
BPL	743	744	847	1176	1265	1307	1736	1746	1770	1806	1814	1933	1934	2052	2223	
BRL	2270	2689	2690	2944	3196	3326	3328	3336	3371	3403	3631					
BRLB	2595	3738	3739	3740												
BRLS	558	561	639	641	666	679	1001	1017	1056	1148	1150	1169	1198	1200	1225	
BRLT	1246	1248	1284	1286	1715	1723	1785	1793	1841	2484	2498	2506	2513	2555	2571	
BRLB	3208	3264	3421	3438	3465											
BRLS	520	2786	2791	2823	3062											
BRLT	657	3003														
BRLB	2991															
BRLS	1922	1938	2695	2814	2934	2975										
BRLT	1706	1757	1929	2246	2248	2250	2252	2254	2256	3582						
BRLB	486	510	536	552	557	559	573	585	591	595	611	618	667	694	721	
BRLS	836	857	879	1002	1018	1151	1170	1199	1231	1247	1249	1269	1285	1287	1311	
BRLT	1716	1780	1786	1794	1833	1842	1927	1987	2023	2025	2034	2055	2103	2113	2485	
BRLB	2514	2556	2561	2579	2596	2618	2685	2734	2785	2792	2794	2802	2810	2824	2831	
BRLS	2889	2895	2915	2922	2929	2966	2972	2995	2997	3013	3017	3027	3061	3067	3070	
BRLT	3087	3203	3206	3227	3229	3236	3238	3265	3281	3308	3332	3338	3422	3439	3462	
BRLB	3468	3602	3634	3639												
BRLS	1776	1826	1913	1943	2019	2241	2267	2568	2683	2779	2828	2891	2907	2962	2968	
BRLT	3352	3629	3666													
BRLB	472	475	478	512	555	565	579	597	600	607	659	663	669	675	698	
BRLS	725	1846	1924	1941	1988	2040	2045	2065	2068	2071	2085	2139	2142	2145	2257	
BRLT	2261	2487	2493	2496	2509	2512	2566	2601	2628	2661	2676	2697	2726	2750	2781	
BRLB	2807	2817	2826	2833	2869	2918	2945	2947	2973	3006	3015	3021	3023	3053	3075	
BRLS	3170	3186	3189	3223	3240	3244	3248	3277	3304	3343	3355	3418	3435	3463	3469	
BRLT	3476	3481	3506	3521	3549	3566	3580	3608	3662	3681	3686	3695				
BRLB	470	471	474	477	484	498	499	519	537	547	612	614	615	621	742	
BRLS	781	826	1067	1224	1661	1720	1750	1773	1774	1791	1865	1866	1916	1919	2089	
BRLT	2235	2271	2511	2526	2594	2674	2732	2857	2858	2904	2905	2988	3011	3192	3201	
BRLB	3287	3324	3325	3329	3333	3341	3458	3500	3518	3542	3563	3579	3632	3635	3659	
BRLS	1945	2510	2806	2832	3018	3028	3091	3092	3093							
BRLT	485	509	535	586	594	610	617	656	660	682	693	719	720	773	802	
BRLB	816	832	852	874	930	982	1030	1046	1074	1095	1116	1137	1164	1181	1208	
BRLS	1258	1300	1487	1644	1680	1683	1699	1937	2000	2038	2100	2102	2494	2518	2578	



	2888	2894	2914	2921	2933	2935	2965	2971	2974	2976	2990	3002	3190	3254	3331
CMPB	3337	3339	3503	3546	3576										
COM	592	2500	2504	2560	2784	2799	2801	2809	2830	2834	2896	2928	2994	3012	3016
DEC	3026	3060	3226	3228	3233	3280	3307	3459	3672						
DECB	873	890	895	900	905	910	915	920	925	942	947	952	957	962	967
EMT	972	977	1196	1341	1355	1369	1383	1397	1411	1425	1439	1453	1467	1481	1518
HALT	1530	1542	1554	1566	1578	1590	1602	1614	1626	1638					
INC	567	1845	1869	1986	2022	2024	2054	2266	2602	3001	3644				
INCB	2682	2693	2813	2816											
IOT	66														
JMP	191	563	658	662	681	701	2002	2569	2580	2725	2749	2780	3353	3480	
JSR	533	554	584	835	1867	1923	2104	2260	2517	2551	2688	2696	2733	2943	3090
	3184	3202	3235	3237	3285	3330	3342	3461	3467	3505	3548	3581			
	2232	2522	2548	2836	3205	3256	3351	3638							
	67														
	195	196	197	198	199	200	201	202	576	616	623	645	653	684	1059
	1850	1852	1887	2003	2213										
	551	571	624	627	630	633	650	710	714	723	724	758	760	762	771
	773	785	788	800	802	814	816	830	832	847	850	852	868	871	873
	888	890	893	895	898	900	903	905	908	910	913	915	918	920	923
	925	928	930	940	942	945	947	950	952	955	957	960	962	965	967
	970	972	975	977	980	982	993	996	999	1001	1012	1015	1017	1028	1030
	1041	1044	1046	1064	1067	1071	1073	1085	1088	1092	1094	1106	1109	1113	1115
	1127	1130	1134	1136	1154	1157	1159	1162	1164	1174	1176	1179	1181	1205	1207
	1218	1221	1223	1242	1245	1253	1256	1258	1265	1268	1280	1283	1294	1297	1300
	1307	1310	1324	1327	1330	1333	1336	1338	1341	1344	1347	1350	1352	1355	1358
	1361	1364	1366	1369	1372	1375	1378	1380	1383	1386	1389	1392	1394	1397	1400
	1403	1406	1408	1411	1414	1417	1420	1422	1425	1428	1431	1434	1436	1439	1442
	1445	1448	1450	1453	1456	1459	1462	1464	1467	1470	1473	1476	1478	1481	1503
	1506	1510	1513	1515	1518	1522	1525	1527	1530	1534	1537	1539	1542	1546	1549
	1551	1554	1558	1561	1563	1566	1570	1573	1575	1578	1582	1585	1587	1590	1594
	1597	1599	1602	1606	1609	1611	1614	1618	1621	1623	1626	1630	1633	1635	1638
	1655	1658	1661	1662	1665	1678	1680	1683	1697	1699	1702	1705	1728	1731	1734
	1736	1739	1741	1744	1746	1749	1753	1756	1765	1768	1770	1773	1798	1801	1804
	1806	1809	1812	1814	1817	1819	1822	1825	1882	1963	1969	1972	1974	1976	1979
	1982	1984	1986	2006	2007	2011	2018	2028	2033	2037	2061	2063	2066	2069	2075
	2079	2083	2135	2137	2140	2143	2149	2153	2157	2161	2165	2171	2175	2179	2183
	2187	2193	2197	2201	2205	2209	2218	2221	2223	2226	2232	2235	2240	2244	2257
	2563	2789	2808	2815	2822	2932	3079	3193	3215	3258	3260	3262	3269	3296	3372
	3376	3381	3385	3404	3407	3410	3424	3427	3441	3511	3519	3553	3564	3601	3675
	3692	3727	3734												
MOV	473	476	479	483	487	489	490	491	492	493	494	495	496	497	501
	502	505	506	507	508	513	515	516	517	522	528	529	530	531	534
	538	539	540	541	545	546	548	568	569	570	603	613	619	620	622
	638	664	672	683	689	690	697	702	711	712	713	715	716	717	718
	726	727	729	731	733	734	736	738	741	745	746	747	748	753	754
	755	767	768	780	782	796	797	810	811	824	825	841	842	844	862
	863	865	885	937	990	993	996	1009	1012	1025	1038	1041	1061	1064	1068
	1082	1085	1088	1103	1106	1109	1124	1127	1130	1144	1147	1154	1171	1192	1194
	1195	1197	1237	1238	1239	1242	1253	1275	1276	1277	1280	1289	1294	1317	1320
	1321	1324	1330	1333	1338	1344	1347	1352	1358	1361	1366	1372	1375	1380	1386
	1389	1394	1400	1403	1408	1414	1417	1422	1428	1431	1436	1442	1445	1450	1456
	1459	1464	1470	1473	1478	1483	1496	1499	1500	1503	1507	1510	1515	1519	1522
	1527	1531	1534	1539	1543	1546	1551	1555	1558	1563	1567	1570	1575	1579	1582





CROSS REFERENCE TABLE

.ENDC	54	66	158	172	180	182	183	184	196	206	210	212	217	219	226
	239	243	245	273	274	275	279	282	304	312	315	318	321	322	323
	324	325	328	380	426	487	488	491	493	495	497	498	499	501	503
	524	540	581	586	588	594	600	602	609	671	677	750	751	752	753
	764	765	766	767	775	777	778	779	780	781	790	793	794	795	796
	804	807	808	809	810	818	821	822	823	824	825	834	838	839	840
	841	859	860	861	862	882	863	884	885	932	934	935	936	937	984
	987	988	989	990	1003	1006	1007	1008	1009	1019	1022	1023	1024	1025	1032
	1035	1036	1037	1038	1048	1053	1054	1055	1056	1076	1079	1080	1081	1082	1097
	1100	1101	1102	1103	1118	1121	1122	1123	1124	1139	1141	1142	1143	1144	1145
	1147	1189	1190	1191	1192	1210	1227	1234	1235	1236	1237	1260	1272	1273	1274
	1275	1302	1313	1314	1315	1316	1489	1492	1493	1494	1495	1646	1648	1649	1650
	1651	1652	1668	1671	1672	1673	1674	1685	1690	1691	1692	1693	1707	1712	1713
	1714	1715	1717	1725	1758	1777	1782	1783	1784	1785	1787	1795	1827	1835	1836
	1837	1838	1839	1857	1859	1860	1862	1865	1871	1874	1875	1879	1881	1887	1889
	1890	1893	1896	2471	2474	2479	2484	2486	2497	2500	2501	2502	2504	2506	2513
	2517	2522	2524	2528	2531	2532	2535	2538	2548	2552	2557	2558	2559	2567	2578
	2582	2583	2587	2602	2631	2635	2713	2722	2723	2729	2735	2736	2746	2748	2752
	2764	2793	2843	2845	2878	2881	2882	2884	2912	2948	2952	2980	2981	2989	2991
	2994	3022	3039	3045	3051	3052	3055	3082	3097	3108	3114	3117	3136	3137	3138
	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3188	3218	3225	3242	3246
	3250	3272	3279	3299	3306	3413	3420	3430	3437	3478	3648	3664	3688	3242	3246
.EQUIV	66	67	75	120	121	122	123	124	125	126	127	128	129	148	149
.EVEN	150	151	152	153	154	155	156	157	2630	2759	3045	3100	3188	3242	3246
	282	581	602	609	671	677	2312	2455	2630	2759	3045	3100	3188	3242	3246
.GLOBL	3250	3478	3664	3688	3702										
.IF	14														
	50	64	130	158	179	181	182	183	184	194	205	208	210	216	218
	225	229	242	244	273	274	275	278	279	281	304	312	315	318	321
	322	323	324	325	326	328	380	423	482	487	489	491	493	495	497
	498	499	501	519	540	580	585	586	587	589	592	601	608	670	676
	749	751	753	763	765	767	774	776	778	780	781	789	792	794	796
	803	806	808	810	817	820	822	824	825	833	837	839	841	858	860
	862	881	883	885	931	933	935	937	983	986	988	990	1002	1005	1007
	1009	1018	1021	1023	1025	1031	1034	1036	1038	1047	1052	1054	1056	1075	1078
	1080	1082	1096	1099	1101	1103	1117	1120	1122	1124	1138	1140	1142	1144	1145
	1146	1188	1190	1192	1209	1226	1233	1235	1237	1259	1271	1273	1275	1301	1312
	1314	1316	1488	1491	1493	1495	1645	1647	1649	1651	1652	1667	1670	1672	1674
	1684	1689	1691	1693	1706	1711	1713	1715	1716	1724	1757	1776	1781	1783	1785
	1786	1794	1826	1834	1836	1838	1839	1856	1857	1858	1859	1860	1861	1862	1864
	1870	1873	1875	1879	1881	1887	1889	1890	1895	2470	2473	2478	2484	2496	2498
	2499	2500	2502	2503	2504	2513	2515	2523	2525	2530	2531	2532	2534	2537	2547
	2551	2555	2557	2558	2560	2567	2571	2578	2582	2583	2586	2601	2617	2634	2712
	2722	2723	2728	2735	2736	2744	2746	2748	2752	2763	2784	2842	2845	2857	2880
	2882	2883	2884	2912	2951	2952	2980	2988	2990	2994	2995	3038	3039	3045	3050
	3052	3055	3082	3097	3107	3113	3117	3128	3137	3138	3139	3140	3141	3142	3144
	3145	3146	3147	3148	3187	3217	3223	3241	3245	3249	3271	3277	3298	3304	3412
	3418	3429	3435	3477	3648	3663	3687								
.IFF	64	179	181	183	184	206	210	212	217	219	226	239	242	245	273
	278	282	487	585	587	750	751	752	753	764	765	766	767	775	777
	778	779	780	790	793	794	795	796	804	807	808	809	810	818	821
	822	823	824	825	833	838	839	840	841	859	860	861	862	882	883
	884	885	932	934	935	936	937	984	987	988	989	990	1003	1006	1007
	1008	1009	1019	1022	1023	1024	1025	1032	1035	1036	1037	1038	1048	1053	1054

	1055	1056	1076	1079	1080	1081	1082	1097	1100	1101	1102	1103	1118	1121	1122
	1123	1124	1139	1141	1142	1143	1144	1145	1147	1189	1190	1191	1192	1209	1226
	1234	1235	1236	1237	1259	1272	1273	1274	1275	1301	1313	1314	1315	1316	1489
	1492	1493	1494	1495	1646	1648	1649	1650	1651	1668	1671	1672	1673	1674	1685
	1690	1691	1692	1693	1707	1712	1713	1714	1715	1717	1724	1757	1776	1782	1783
	1784	1785	1787	1794	1826	1835	1836	1837	1838	1839	1857	1861	1865	1870	1873
	1889	1896	2471	2497	2500	2501	2504	2531	2535	2537	2552	2578	2583	2587	2602
	2631	2635	2713	2729	2744	2764	2843	2881	2884	2952	2954	2959	2980	2981	2990
	3022	3038	3051	3108	3114	3217	3223	3271	3277	3298	3304	3412	3418	3429	3435
.IFT	581	602	609	671	677	2512	2558	2861	2877	2878	2954	2959	3188	3242	3246
.IFTF	3250	3478	3664	3688											
.IIF	581	602	609	671	677	2510	2557	2857	2861	2877	2899	2952	2955	3188	3242
	3246	3250	3478	3664	3688										
.IIF	49	54	59	176	177	178	180	183	184	191	278	282	488	491	497
	498	499	501	502	586	1859	1865	1866	1877	1889	1893	2474	2475	2476	2477
	2478	2479	2483	2511	2512	2528	2531	2532	2538	2539	2540	2541	2546	2570	2578
	2583	2599	2624	2840	2881	2902	3030	3039	3045	3136	3137	3138	3139	3140	3142
	3144	3145	3146	3147	3680										
.IRP	380	749	763	776	792	806	820	837	858	881	933	986	1005	1021	1034
	1052	1078	1099	1120	1140	1188	1233	1271	1312	1491	1647	1670	1689	1711	1781
.LIST	1834	1906	1946	2547	2716	2722	2735	2736	2852	2873	3056	3057	3078	3094	3095
	14	30	172	183	191	273	279	282	380	503	551	581	586	589	602
	609	671	677	749	753	758	760	762	763	767	771	773	776	780	785
	788	792	796	800	802	806	810	814	816	820	824	830	832	837	841
	847	850	852	858	862	868	871	873	881	885	888	890	893	895	898
	900	903	905	908	910	913	915	918	920	923	925	928	930	933	937
	940	942	945	947	950	952	955	957	960	962	965	967	970	972	975
	977	980	982	986	990	993	996	999	1001	1005	1009	1012	1015	1017	1021
	1025	1028	1030	1034	1038	1041	1044	1046	1052	1056	1064	1067	1071	1073	1078
	1082	1085	1088	1092	1094	1099	1103	1106	1109	1113	1115	1120	1124	1127	1130
	1134	1136	1140	1144	1154	1157	1159	1162	1164	1174	1176	1179	1181	1188	1192
	1205	1207	1218	1221	1223	1233	1237	1242	1245	1253	1256	1258	1265	1268	1271
	1275	1280	1283	1294	1297	1300	1307	1310	1312	1316	1324	1327	1330	1333	1336
	1338	1341	1344	1347	1350	1352	1355	1358	1361	1364	1366	1369	1372	1375	1378
	1380	1383	1386	1389	1392	1394	1397	1400	1403	1406	1408	1411	1414	1417	1420
	1422	1425	1428	1431	1434	1436	1439	1442	1445	1448	1450	1453	1456	1459	1462
	1464	1467	1470	1473	1476	1478	1481	1491	1495	1503	1506	1510	1513	1515	1518
	1522	1525	1527	1530	1534	1537	1539	1542	1546	1549	1551	1554	1558	1561	1563
	1566	1570	1573	1575	1578	1582	1585	1587	1590	1594	1597	1599	1602	1606	1609
	1611	1614	1618	1621	1623	1626	1630	1633	1635	1638	1647	1651	1655	1658	1661
	1665	1670	1674	1678	1680	1683	1689	1693	1697	1699	1702	1705	1711	1715	1728
	1731	1734	1736	1739	1741	1744	1746	1749	1753	1756	1765	1768	1770	1773	1781
	1785	1798	1801	1804	1806	1809	1812	1814	1817	1819	1822	1825	1834	1838	1865
	1881	1969	1972	1974	1976	1979	1982	1984	1986	2011	2018	2028	2033	2218	2221
	2223	2226	2232	2235	2240	2244	2478	2578	2980	3128	3136	3137	3138	3139	3140
	3141	3142	3143	3144	3145	3146	3147	3148	3188	3242	3246	3250	3478	3601	3664
	3688														
.MACRO	17	18	19	20	22	24	25	26	27	28	29	30	184	236	431
	467	519	1050	1051	2147	3128									
.MCALL	30	172	279	503	589										
.MEXIT	327														
.NLIST	14	30	172	183	191	273	279	282	380	503	551	581	586	589	602
	609	671	677	749	753	758	760	762	763	767	771	773	776	780	785
	788	792	796	800	802	806	810	814	816	820	824	830	832	837	841



847	850	852	858	862	868	871	873	881	885	888	890	893	895	898
900	903	905	908	910	913	915	918	920	923	925	928	930	933	937
940	942	945	947	950	952	955	957	960	962	965	967	970	972	975
977	980	982	986	990	993	996	999	1001	1005	1009	1012	1015	1017	1021
1025	1028	1030	1034	1038	1041	1044	1046	1052	1056	1064	1067	1071	1073	1078
1082	1085	1088	1092	1094	1099	1103	1106	1109	1113	1115	1120	1124	1127	1130
1134	1136	1140	1144	1154	1157	1159	1162	1164	1174	1176	1179	1181	1188	1192
1205	1207	1218	1221	1223	1233	1237	1242	1245	1253	1256	1258	1265	1268	1271
1275	1280	1283	1294	1297	1300	1307	1310	1312	1316	1324	1327	1330	1333	1336
1338	1341	1344	1347	1350	1352	1355	1358	1361	1364	1366	1369	1372	1375	1378
1380	1383	1386	1389	1392	1394	1397	1400	1403	1406	1408	1411	1414	1417	1420
1422	1425	1428	1431	1434	1436	1439	1442	1445	1448	1450	1453	1456	1459	1462
1464	1467	1470	1473	1476	1478	1481	1491	1495	1503	1506	1510	1513	1515	1518
1522	1525	1527	1530	1534	1537	1539	1542	1546	1549	1551	1554	1558	1561	1563
1566	1570	1573	1575	1578	1582	1585	1587	1590	1594	1597	1599	1602	1606	1609
1611	1614	1618	1621	1623	1626	1630	1633	1635	1638	1647	1651	1655	1658	1661
1665	1670	1674	1678	1680	1683	1689	1693	1697	1699	1702	1705	1711	1715	1728
1731	1734	1736	1739	1741	1744	1746	1749	1753	1756	1765	1768	1770	1773	1781
1785	1798	1801	1804	1806	1809	1812	1814	1817	1819	1822	1825	1834	1838	1865
1881	1969	1972	1974	1976	1979	1982	1984	1986	2011	2018	2028	2033	2218	2221
2223	2226	2232	2235	2240	2244	2478	2578	2980	3128	3136	3137	3138	3139	3140
3141	3142	3143	3144	3145	3146	3147	3148	3188	3242	3246	3250	3478	3601	3664
3688														
.PAGE	30	172	236	328										
.PSECT	14													
.REM	1	14	30											
.REPT	191	888	940	1328	1507									
.SBTTL	62	172	185	194	203	214	236	279	328	481	582	589	637	749
	776	792	806	820	837	858	881	933	986	1005	1021	1034	1052	1078
	1120	1140	1188	1233	1271	1312	1491	1647	1670	1689	1711	1781	1834	1854
	1961	2004	2059	2133	2468	2532	2584	2632	2710	2761	2840	2878	3048	3105
.TITLE	49													
.WORD	191	192	193	211	230	231	232	233	234	235	244	247	248	249
	253	254	255	256	257	258	259	262	263	264	284	285	286	287
	289	290	291	295	296	297	310	314	317	320	321	322	323	324
	406	409	411	413	415	417	419	421	423	424	426	428	551	605
	760	762	771	773	785	788	800	802	814	816	830	832	847	850
	868	871	873	888	890	893	895	898	900	903	905	908	910	913
	918	920	923	925	928	930	940	942	945	947	950	952	955	957
	962	965	967	970	972	975	977	980	982	993	996	999	1001	1012
	1017	1028	1030	1041	1044	1046	1064	1067	1071	1073	1085	1088	1092	1094
	1109	1113	1115	1127	1130	1134	1136	1154	1157	1159	1162	1164	1176	1179
	1181	1205	1207	1218	1221	1223	1242	1245	1253	1256	1258	1265	1268	1280
	1294	1297	1300	1307	1310	1324	1327	1330	1333	1336	1338	1341	1344	1347
	1352	1355	1358	1361	1364	1366	1369	1372	1375	1378	1380	1383	1386	1389
	1394	1397	1400	1403	1406	1408	1411	1414	1417	1420	1422	1425	1428	1431
	1436	1439	1442	1445	1448	1450	1453	1456	1459	1462	1464	1467	1470	1473
	1478	1481	1503	1506	1510	1513	1515	1518	1522	1525	1527	1530	1534	1537
	1542	1546	1549	1551	1554	1558	1561	1563	1566	1570	1573	1575	1578	1582
	1587	1590	1594	1597	1599	1602	1606	1609	1611	1614	1618	1621	1623	1626
	1633	1635	1638	1655	1658	1661	1665	1678	1680	1683	1697	1699	1702	1705
	1731	1734	1736	1739	1741	1744	1746	1749	1753	1756	1765	1768	1770	1773
	1801	1804	1806	1809	1812	1814	1817	1819	1822	1825	1870	1873	1888	1969
	1974	1976	1979	1982	1984	1986	2011	2018	2028	2033	2218	2223	2226	2232

2235	2240	2244	2610	2615	2708	2745	2747	2790	2837	2877	3080	3135	3194	3291
3311	3446	3512	3513	3520	3555	3565	3586	3587	3601	3611	3612	3642	3646	3676
3677	3682	3693	3694	3697	3698	3728	3729	3735	3736	3742				

000000

ERRORS DETECTED: 0



E09

DR11-K LOGIC TEST MAINDEC-11-DRLPF-A MACY11 27(654) 14-DEC-77 20:26 PAGE 97  
DRLPF.P11

SEQ 0108

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